IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 November 2023, accepted 23 November 2023, date of publication 27 November 2023,
date of current version 30 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3336912

RESEARCH ARTICLE

Performance Comparison of Machine Learning Models for Handwritten Devanagari Numerals Classification

AGASTYA GUMMARAJU, AJITHA K. B. SHENOY, AND SMITHA N. PAI, (Senior Member, IEEE)
Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka 576104, India

Corresponding authors: Ajitha K. B. Shenoy (ajith.shenoy@manipal.edu) and Smitha N. Pai (smitha.pai@manipal.edu)

ABSTRACT This work focuses on comparing the suitability of different machine learning models for the classification of handwritten digits in the Devanagari script. The models that will be compared in this study are: K-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Convolutional Neural Network (CNN), GoogLeNet (Inception v1), and ResNet-50. GoogLeNet and ResNet-50 are complex, deep neural networks. They possess a large number of hidden layers, and are generally used for more complex image classification tasks. The use of these models in this project is to gauge how well they perform on simpler image data. The foundation of this research is based on the ever increasing demand for accurate and efficient digit classification models in India, for purposes such as document scanning, ID card recognition, and the digitization of institutional records. The primary objective of this research project is to identify the most accurate and efficient digit classification model for numbers in the Devanagari script. Surprisingly, proposed simple CNN model outperforms the other complex GoogleNet and ResNet-50 models. Accuracy and Fl score of proposed CNN model is 99.522% and 0.9978 respectively. Also, the proposed CNN model used in this study outperforms other CNN model considered for Devanagari numerals classification.

INDEX TERMS Convolution neural networks, deep learning, Devanagari script, handwritten numerals classification, quality education, GoogLeNet, image classification, K-nearest neighbours, machine learning, ResNet-50, support vector machine.

I. INTRODUCTION

Handwriting recognition is the process by which a computer identifies and understands written text. It involves the use of machine learning algorithms and models to interpret and analyze written text. Handwriting recognition systems can be used in a variety of applications, such as character/digit recognition for OCR systems [1], signature verification [2], [3], and document scanning, Digit classification, or number recognition [4], [5], is a specific task within handwriting recognition that involves identifying and recognizing digits or numbers within an image or document. The goal is to accurately identify the digits present in an image or document. The most common approach to digit classification is to use machine learning techniques, such as neural networks, to train a model on a labelled dataset containing images of handwritten digits. The model learns to recognize the digits based on their visual features and is able to classify new images of handwritten digits.

At one point in time, all of our tax records, land ownership records, as well as personal identification records were stored in a physical format. This meant that they took up a lot of physical space, were susceptible to damage due to environmental factors, it took a lot of time to have to sift through thousands of pages worth of information in order to refer to a particular document, and editing these records was a very tedious process as well. 2 Ever since then, there has been a gradual digitization of records through handwriting recognition software. Today, handwriting recognition has evolved significantly, and is used in various domains. For

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

example, It is used in banking: for the automatic processing of cheques, extraction of information from physical forms, and the verification of signatures to prevent fraud; it is used in healthcare: for the digitization of patient records; and it is also used by the government: to extract information from images of PAN cards, Aadhar cards and Passports, and to enhance the accessibility of public records. While a lot of research has been done on computer vision tasks involving the standard number system as well as the English language [6], [7], [8], [9], [10], [11], we have not seen the same level of development when it comes to Hindi, or any of the native languages that are spoken in India. This project aims at furthering the research that is currently being carried out for the detection of Devanagari numbers, as the aforementioned task has many real world use cases. This work is a stepping stone to more complex projects, such as an optical character recognition system that can detect multiple numbers and characters, or detect numbers and characters in real time using a live video feed. The purpose of this project, however, is simply to compare and contrast the performance of different models in order to ascertain which system works best at classifying digits after being trained on an image database of hand-written numbers in a native language. One of the main challenges of optical character recognition systems that work with hand-written numbers and characters is dealing with variations in writing styles, and since there are very few publicly available databases of hand-written Devanagari numbers, the importance of our data pre-processing phase is magnified. During this phase, we introduce small variations to the images, such as rotation, zoom, and shifts in width/height.

There have been several studies that have compared the performance of different systems for handwriting recognition and digit classification. These studies have shown that CNNs generally perform better than other methods, such as K-Nearest Neighbours and Support Vector Machine. However, these results can differ on a case to case basis. Moreover, more complex models, such as GoogLeNet (Inception v1) and ResNet-50, are not commonly used for tasks such as number classification. This study will outline how such complex models perform when trained on our image dataset, and will establish whether or not they can, in fact, perform better than much simpler models, or whether they are too complex for this use-case. Handwriting Recognition has been a topic of research for several decades. Previously, researchers used structural and syntactic approaches for the analysis of handwriting. In the 1990s, researchers began to use statistical models, such as Hidden Markov Models (HMMs) and Support Vector Machines (SVMs) to recognize handwriting. With the advent of deep learning, convolutional neural networks (CNNs) have become the most popular method for handwriting recognition.

When it comes to the detection of Hindi/Devanagari characters, due to the complexity of the language as well as the lack of publicly available datasets, models have historically not performed as well as their standard counterparts. While work has been done in these fields in recent years, there is still room for development. There have been several studies that have compared the performance of different systems for handwriting recognition and digit classification. These studies have shown that CNNs generally perform better than other methods, such as KNN and SVM. However, these studies have mostly focused on the standard number system, and there is a need for more research on the comparison of different systems for handwriting recognition and digit classification for Devanagari digits.

Seng et al. [12] worked on handwritten digit classification on the MNIST dataset using different CNN architectures. The MNIST dataset contains 70,000 grayscale images, each of size 28×28. There are a total of 10 classes, representing the numbers 0-9. The authors modified the train-test distribution, which was 42,000 images for training and 28,000 images for testing in this case. They first implemented the ResNet-18 architecture and then modified its architecture using existing PyTorch architectures. The models that were implemented are: GoogLeNet, MobileNet v2, ResNet-50, ResNeXt-50 and Wide ResNet-50. Wide ResNet-50 had the lowest top 1% error, at 0.5278%. GoogLeNet was a close second, with a top 1% error rate of 0.5317%. MobileNet v2 achieved the third best 5 top 1% error rate of 0.5754% and the best top 5% error rate of 0.0079% (tied with Wide ResNet-50) despite being significantly smaller than Wide ResNet-50.

Reddy et al. [13] proposed a deep learning method for the recognition of handwritten Hindi digits. The proposed model, a Convolutional Neural Network (CNN) with root mean square propagation (RMSprop) optimization, was trained on a sample of 20,000 images of handwritten Hindi digits. The network consisted of 7 layers in total. The first and third were Convolutional layers, the second and fourth were Max Pooling layers, the fifth was a Flatten layer, and the last two were Fully Connected Dense layers. The authors were able to achieve a top 1% accuracy of 99.85%.

Ahamed et al. [14] proposed an SVM based real time handwritten digit recognition system. The model was tested on the MNIST dataset (which contains 60,000 training images and 10,000 testing images) and was able to achieve a testing accuracy of 97.83%. The model was also made capable of accurately classifying hand-written digits, by first converting the scanned image to gray-scale, performing binarization and thresholding, feature extraction, and feeding this processed data to the classifier.

Babu et al. [15] proposed a K-NN model for the classification of digits based on 4 structural features. The model was trained on 50,000 images, and was tested on 5,000 images belonging to the MNIST dataset. Euclidean distances were used to determine the nearest neighbours. The proposed model was able to achieve a testing accuracy of 96.94%.

Prashanth et al. [16] analysed performance of Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) on the dataset [17]. They are able to achieve testing

accuracy of 97.22% using CNN. The F1 score, precision and recall obtained are 0.9721, 0.9720 and 0.9724 respectively. On the same dataset, our proposed CNN architecture is able to achieve testing accuracy of 99.522%. The F1 score, precision and recall obtained are 0.997, 0.998 and 0.997 and respectively. This is the best known accuracy, F1 score, precision and recall for data set [17] as per our knowledge/literature studied.

Mhapsekar et al. [18] analyzed the performance of ResNet for Devanagari handwritten numerals classification on dataset [19]. They are able to achieve the accuracy of 99.35%. On the same dataset, the proposed CNN model in this paper able to achieve the better accuracy than ResNet model used by Mhapsekar et al. in [18]. The other literature on Devanagari numerals recognition can be found in [20], [21], [22], [23], and [24] etc.

II. MODELS CONSIDERED IN THIS STUDY

In this study we have used five machine learning models and compared their performance for Devanagari handwritten numerals classification. The models considered are:

1) K-Nearest Neighour (K-NN) [25], [26]
2) Support Vector Machine (SVM) [27], [28]
3) Convolutional Neural Network (CNN) [29]
4) VGG-16 [30], [31]
5) GoogLeNet (Inception V1) [32], [33]
6) ResNet-50 [34], [35]

The main intention of this study is to try and ascertain the level of model complexity required in order to achieve the best possible results for handwritten Devanagari digit classification. KNN, SVM and CNN are some of the most commonly used models for relatively simple image classification tasks such as in this case, Devanagari digit classification. Hence, the proposed CNN model is compared with the KNN and SVM model. To understand whether the increase in model complexity will give us better results, this study compares the proposed CNN model with the complex deep learning models VGG-16, ResNet-50 and GoogLeNet (Inseption V1). From this work it is evident that ceiling to be the CNN model, after which, an increase in model complexity did not imply an increase in accuracy, efficiency or speed.

A. K-NEAREST NEIGHBOURS

K-Nearest Neighbours (K-NN) is a simple, non-parametric, supervised machine learning algorithm that is commonly used to perform both classification and regression tasks. K-NN classifies an input image by finding the k-nearest training examples in the feature space and then assigning the input image to the class that is most common among the k-nearest training examples. The first step in using a K-NN network for digit classification is to extract features from the input images. These features can be the pixel values of the image, or more sophisticated features extracted using techniques such as edge detection or wavelet transform. Once the features have been extracted, the K-NN algorithm is trained using a labelled dataset of images of hand-written digits. For each image in the dataset, the algorithm stores the feature vector and the corresponding class label (i.e., the digit represented by the image). When a new image needs to be classified, the K-NN algorithm first extracts the features from the image. Then, it finds the k-nearest training examples in the feature space using a distance metric such as Euclidean or Manhattan distance. Finally, it assigns the input image to the class that is most common among the k-nearest training examples. The value of k, which denotes the number of nearest training samples to be taken into consideration during classification, needs to be experimentally determined as there isn't a concrete way of choosing it for any given dataset. If the value of k is very small, very few training samples are considered during classification, increasing the model's sensitivity towards noise and outliers. Since fewer training samples are considered, each sample has a larger effect on the outcome of the classification process (analogous to over-fitting). On the contrary, a very large value of k will reduce the model's sensitivity towards noise and outliers, making it more generalized (analogous to under-fitting). More about working of KNN model is given in [25] and [26].

B. SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a more recent supervised machine learning algorithm that can also be used to perform both classification and regression tasks. The basic idea behind SVM is to construct a hyperplane in a high-dimensional feature space that separates the different classes of data. Once the hyperplane is constructed, new data points can be classified by determining on which side of the hyperplane they fall. The SVM algorithm tries to find a hyperplane such that the distance between the nearest data points of each class to the hyperplane is maximum. This distance is known as the margin, and these nearest data points are known as support vectors. SVM is very useful when we are dealing with linearly non-separable data. This is because we can assess the relationship between data points as if they are in a higher dimension (without actually transforming the data) using kernel functions. In higher dimensions, it becomes possible for us to find a hyperplane that can separate the data points. The choice of kernel function depends on the type of data we are working with. If we are dealing with simple, linearly separable data, then we can make use of the linear kernel function. If we are dealing with more complex, linearly non-separable data, then we can make use of either the polynomial or radial basis function kernel. In this case, we make use of the radial basis function kernel. When a new image needs to be classified, the SVM algorithm first extracts the features from the image, and then finds out which class the new image belongs to, based on where it lies with respect to the hyperplane. SVM works well on small and simple datasets, and is robust to overfitting. SVM does not work as well when the number of features per data point is greater than the total number of data points in the training dataset. More about SVM model and it's working is given in [27] and [28].

C. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNNs) are specialized neural networks that are used to learn and recognize patterns in images. They are particularly well-suited for image classification tasks, because they are able to automatically learn features from the input images. While regular artificial neural networks treat input images as flat vectors of pixels, CNNs first extract features from the input images at different levels of abstraction and then pass this information to a regular artificial neural network (the fully connected layer) in order to perform classification. The basic building block of a CNN is the convolutional layer. These layers applies a set of filters to the input image and these filters are used to detect particular features in the image, such as edges, lines, and other abstract patterns. The output of the convolutional layer is called a feature map, which is then passed through a non-linear activation function, such as a rectified linear unit (ReLU), to introduce non-linearity to the model. Non-linearity is important because the relationship between the pixels of the image and the object that they represent is normally a highly non-linear relationship which cannot effectively be captured by a simple linear activation function. The feature maps are then typically passed through a pooling layer, which reduces the spatial dimensions of the feature maps and makes the model more robust to small translations of the input image. This process is called down-sampling and it helps to reduce the computational cost of the model. After one or more convolutional and pooling layers, the feature maps are then passed through a fully connected layer, which computes the final output of the model. The final output is a probability distribution over the possible classes, i.e., the digits from 0 to 9. CNNs are able to understand representations of images at different levels of abstraction. They are easier to train, more robust to overfitting, and have much fewer parameters as compared to traditional artificial neural networks, and are hence, the better option for image classification tasks. The structure of the CNN model used in the proposed study is represented in Figure 1. The proposed Convolutional Neural Network (CNN) model is able to classify hand-written Devanagari numerals with a testing accuracy of 99.522%. It begins with two convolutional layers, each with 32 filters, a 3×3 kernel size, and the ReLU activation function. After the pair of convolutional layers, there is a max-pooling layer with a 2×2 pool size and a dropout layer with a dropout rate of 0.25. Then, the model repeats the same pattern with two more convolutional layers with 64 filters, followed by max-pooling, and dropout layers. After these convolutional layers, the model flattens the output and passes it through two fully connected dense layers with 256 units and the ReLU activation function, along with a dropout layer. Finally, the model has an output layer with 10 units and the softmax activation function for multi-class classification. The model takes input images of size 32×32 and 1 channel, since the images are grayscale.

D. VGG-16

VGG-16 is a type of Convolutional Neural Network (CNN) architecture that is commonly used for object detection and classification. It contains a total of thirteen convolutional layers, five max-pooling layers, and three dense layers. In this study, we used VGG-16 pre-trained on the ImageNet database. Using randomly assigned weights gave us sub-par results. The model was trained over 20 epochs, with a batch size of 128. The used VGG-16 model is represented in the Figure 2. The more about VGG 16 is found in the reference [30], [31].

E. GoogLeNet (INCEPTION V1)

GoogLeNet is a deep convolutional neural network architecture, proposed by researchers at Google in 2014. It was responsible for achieving state of the art classification and detection results in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). It has 22 layers, but 9 of these layers are made up of 'inception modules', that allow for a reduction of parameters and an increase in computational efficiency. These inception modules consist of multiple parallel convolutional layers of different sizes (1×1, 3×3 and 5×5) and pooling operations. This allows the network to capture information at various scales and extract both local and global features efficiently. A large number of filters can be used without running into computational problems. The architecture also incorporates a 1×1 convolutional layer before the larger convolutions to reduce the dimensionality of the input and control computational complexity. This helps in reducing the number of parameters and allows for deeper network architectures. GoogLeNet also uses various techniques to combat overfitting, such as dropout and L2 weight decay. Overall, GoogLeNet is an efficient and accurate deep learning architecture, and its success has influenced the development of other popular deep learning architectures such as ResNet. It is important to note that GoogLeNet was created to perform well on larger and more sophisticated datasets such as ImageNet, which is drastically more visually complex than the database used for the purpose of this research project, as it contains colour images of 1,000 different classes of objects. Thus, it will be noteworthy to observe how well GoogLeNet performs on the dataset that is being used in this case. For the GoogLeNet architecture diagram refer the base paper [33] (Figure 3 in [33]).

F. ResNet-50

ResNet-50 is a deep convolutional neural network architecture, proposed by researchers at Microsoft in 2015. It is a part of the ResNet (Residual Network) family. It has 50 layers. The key feature associated with the ResNet family is its ability to train deep networks effectively without suffering from degradation issues. Generally, as we increase the depth of a network, it becomes harder to train and can lead to lower accuracy. ResNet tackles this issue with the help of 'residual connections' or 'skip connections'. Residual

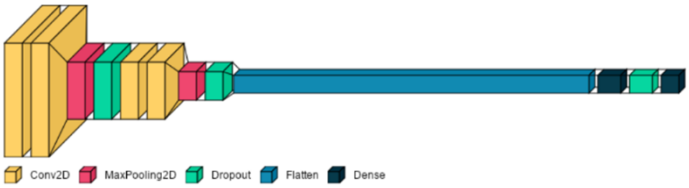

FIGURE 1. Structure of CNN used in this study.

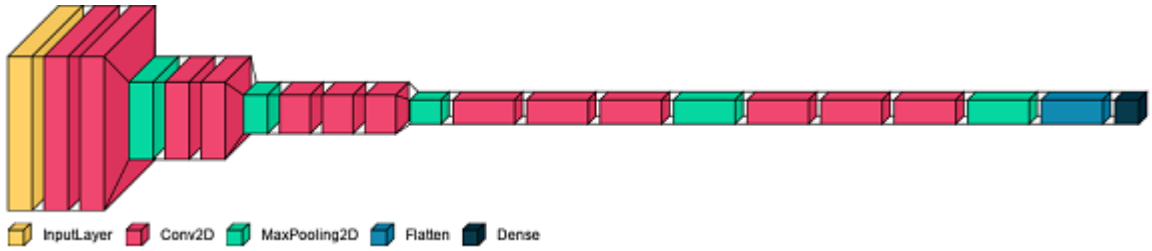

FIGURE 2. Structure of the VGG-16 used in this study.

connections allow information to bypass certain layers and be directly propagated to subsequent layers. By doing so, the network can effectively learn residual functions, which capture the difference between the desired output and the current output of a particular layer. This mechanism helps combat the problem of vanishing gradients, where gradients become increasingly small as they propagate backward through many layers during training. ResNet-50 is composed of building blocks called ''residual blocks.'' These blocks typically consist of two or three convolutional layers along with shortcut connections. The core component of ResNet-50 is the ''bottleneck block,'' which involves a sequence of 1×1, 3×3, and 1×1 convolutions. This design reduces the computational cost and parameter count while enabling the network to learn complex features effectively. At the end of ResNet-50, there is a global average pooling layer that condenses the spatial dimensions of the features while preserving important information. This is followed by a final classification layer, such as a softmax layer, which determines the most likely category or label for the given image. Like GoogLeNet, ResNet50 is also a much deeper, more complex architecture than the previous models that have been discussed in this report, but it will be noteworthy to see how it performs on our dataset. For the ResNet-50 architecture diagram refer the paper [35] (Figure 5 in [35]).

In the next section we describe the datasets used for training and testing the models.

III. DATASET

The data that was used for this research project was a combination of 2 datasets, [17] and [19]. Dataset [19] contains 20,000 images of size 32×32 pixels, with black backgrounds and white lettering and dataset [17] contains 2,880 images of size 28×28 pixels, with white backgrounds and black lettering. We used two datasets since they contain images with varying backgrounds. This makes our model more

FIGURE 3. Sample images from first dataset [19].

FIGURE 4. Sample images from second dataset [17].

background-independent and promotes better generalization. We are essentially increasing the diversity of our training data, allowing the model to learn features and representations that are invariant to background variations. Using two datasets substantially increased the model's accuracy when feeding it hand-drawn images (either on paper or digitally), with noisy backgrounds. Two datasets were used in order to introduce variations in terms of writing style, background and resolution. Since both datasets are individually quite small, putting them together increases the amount of information that we can feed to our models during the training phase. Our merged training set contains 10 classes (0 to 9), with 1,988 images per class. Our merged testing set contains 10 classes (0 to 9), with 300 images per class. The training images from [19] are of size 32×32, with black backgrounds and white lettering, whereas the training images from [17] are of size 28×28, with white backgrounds and black lettering. Sample images from dataset [17] and [19] are represented in Figure 3 and Figure 4 respectively. The Devanagari numerals from 0 to 9 is given in Figure 5 [19]. Testing was only performed on [19] and not a combination of [17] and [19], for the sake of simplicity and continuity with respect to other published models that have performed testing on the same dataset. The same training and testing folders were used for all 5 models.

FIGURE 5. Devanagari numerals from 0 to 9 [19].

TABLE 1. Parameter settings.

Models	Parameter Settings
K-NN	k=3
SVM	Kernal: Radial Basis Function (RBF) The value of $C = 10$, and the value of gamma was set to 'scale', which is $1/(n_{features} * X.var())$, where '$n_{features}$' is the number of features in our dataset.
CNN	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 32 Steps per Epoch: 622 Number of Epochs: 15 Validation Steps: 20
VGG-16	thirteen convolutional layers five max-pooling layers three dense layers 20 epochs, with a batch size of 128
GoogLeNet	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 32 Steps per Epoch: 622 Number of Epochs:10
ResNet-50	Optimizer- Adam (Stochastic Gradient Descent) Loss: Categorical Crossentropy Batch Size: 50 Steps per Epoch: 50 Dense Layer Activation Function: Softmax Number of Epochs: 30 Early Stop Criteria: 15 Epochs

IV. RESULTS AND DISCUSSION

Parameter settings for different machine learning models are chosen by experimenting with different values through trial and error method and selecting the ones which gives best results. The best chosen parameter settings for different models are given in Table 1.

Confusion matrix for KNN and SVM and proposed CNN models are given in Figure 6, Figure 7 and Figure 8 respectively.

Out of 3000 handwritten Devanagari numerals image considered, K-NN and SVM misclassifies 52 and 22 respectively. Few of them are shown in Figure 9. From the confustion matrices, it is evident that the proposed CNN model is better than KNN and SVM. Number of images misclassifed by propsed CNN model is very less.

Precision, Recall and F1-score for all the models considered are tabulated in Table 2. It is evident from the result (Table 2) that CNN outperform basic machine learning models K-NN and SVM as well as deep learning models like GoogLeNet and ResNet-50. The accuracy and loss graph for the proposed CNN model is given in Figure 10 and Figure 11 respectively.

To generalize the result further, we have applied K-fold cross validation technique for CNN model. Standard value for K as per the literature is 10. The results of K-fold validation (for K=10) is given in Table 3. From the Kfold validation result (mean and standard deviation obtained in Table 3,

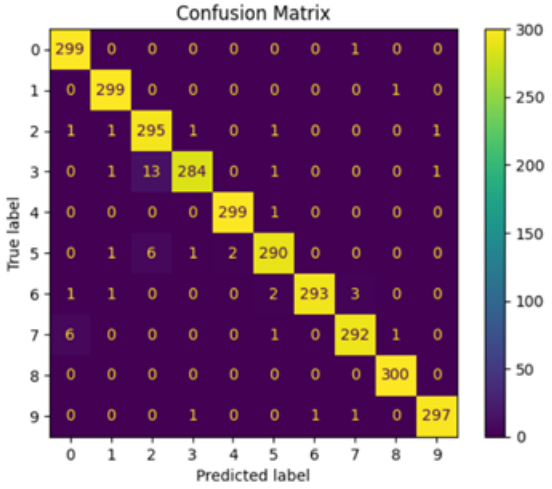

FIGURE 6. Confusion matrix for KNN.

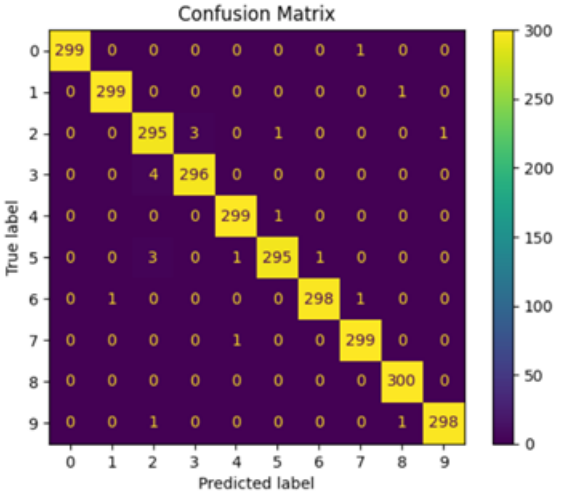

FIGURE 7. Confusion matrix for SVM.

we can generalize that that the proposed CNN model perform very well for handwritten Devanagari numerals classification problem.

CNN emerged as the best performer, achieving the highest accuracy among all of the models tested. Its ability to learn complex patterns and hierarchical features from image data proved instrumental in achieving very good classification results. Overfitting was accounted for through data augmentation and the use of dropout layers. The model also performed very well on unseen, hand-drawn images of Devanagari digits, tested using MS Paint. The proposed model outperforms SVM and KNN due to some of the inherent strengths of convolutional neural networks, such as their ability to capture local patterns and relationships, translational invariance, and non-linearity. Since our dataset is quite small, and since the images themselves are quite simple (being grayscale images of size 28×28 and 32×32), the most likely reason for the lower accuracy of

TABLE 2. Overall comparison of results.

Models	Training Accuracy	Testing Accuracy	Precision	Recall	F1-Score
K-NN		98.266%	0.98	0.98	0.98
SVM		99.266%	0.99	0.99	0.99
CNN	99.766%	99.522%	0.998	0.9976	0.9978
VGG-16	96.399%	94.059%	0.9756	0.9496	0.9625
GoogLeNet	99.133%	97.469%	0.9923	0.99	0.991
ResNet-50	98.5%	95.256%	0.987	0.984	0.985

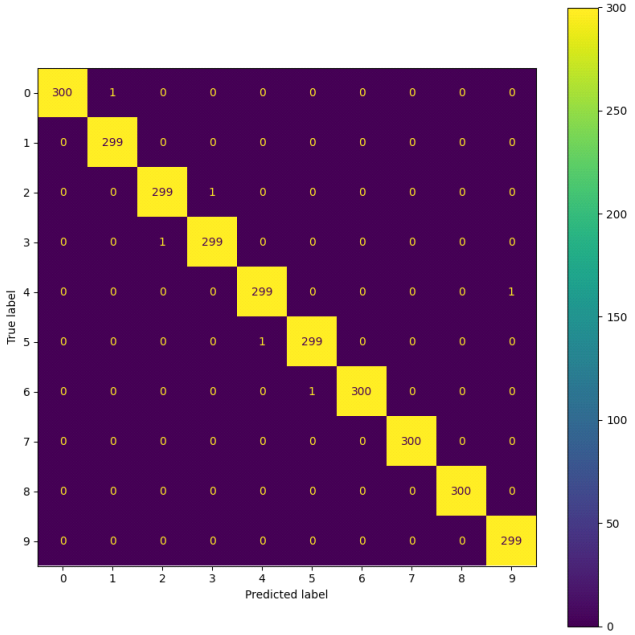

FIGURE 8. Confusion matrix for CNN.

FIGURE 9. (a) 2 misclassified as 9 (b) 3 misclassified as 2 (c) 5 misclassified as 4 (d) 6 misclassified as 5 (e) 7 misclassified as 8.

TABLE 3. K-Fold validation results (K=10).

Folds	Accuracy in %	Precision	Recall
Fold 1	99.299	0.995	0.992
Fold 2	99.366	0.993	0.993
Fold 3	99.400	0.995	0.994
Fold 4	99.466	0.994	0.994
Fold 5	99.266	0.992	0.992
Fold 6	99.500	0.995	0.995
Fold 7	99.466	0.995	0.994
Fold 8	99.400	0.994	0.993
Fold 9	99.566	0.995	0.995
Fold 10	99.566	0.995	0.995
Mean	**99.430**	**0.994**	**0.993**
Standard Deviation	**0.09712**	**0.00096**	**0.00095**

ResNet-50 and GoogLeNet when compared to the proposed simple CNN model, is the high complexity of these models. Deeper architectures like ResNet-50 and GoogLeNet have

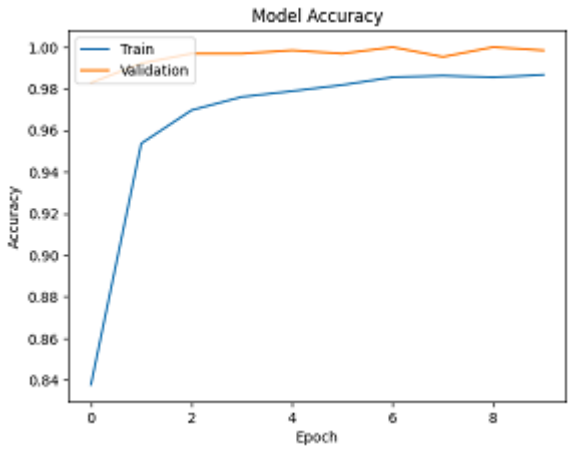

FIGURE 10. Accuracy graph for CNN.

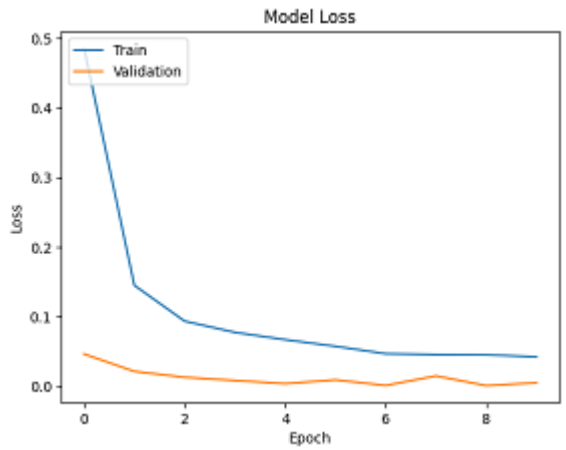

FIGURE 11. Loss graph for CNN.

a higher capacity to learn complex patterns, but they might require more data or more extensive hyperparameter tuning to achieve optimal performance on smaller datasets. The additional complexity of these models may make it more difficult to effectively learn from the limited information available in the dataset, resulting in slightly lower accuracy.

V. CONCLUSION AND FUTURE SCOPE

In this study, we performed Devanagari number classification using various machine learning models, with the goal of determining which one of them achieves the highest testing accuracy. Our study highlights the superior performance of CNN, followed by SVM, GoogLeNet, ResNet-50, and K-NN.

Surprisingly, the proposed CNN model outperformed all the other model in terms of accuracy, precision, recall and F1 score. The results of our study have significant implications for real-life applications. Accurate digit classification can greatly contribute to the efficiency and automation of tasks such as ID card recognition, document scanning, and the digitization of physical records. By leveraging the advancements in deep learning and CNN architectures, the accuracy and reliability of these applications can be significantly improved. While our study has provided valuable insights, there are several avenues for future research and enhancement. Further fine-tuning and optimization of ResNet-50 and GoogLeNet can be explored, including hyper parameter tuning and architecture optimization. Transfer learning from larger datasets could also be investigated to leverage pre-trained models for improved performance. Ensembling techniques that combine the strengths of multiple models could be employed to harness their complementary capabilities and potentially achieve even higher accuracy. Additionally, research focused on interpret-ability can provide insights into the decision-making process of deep learning models. Techniques to visualize learned features or identify important regions in input images would facilitate a better understanding of the classification process. In conclusion, our findings have practical implications for real world applications, and future research directions include fine tuning and optimizing existing models, utilizing transfer learning from larger datasets, exploring ensembling techniques and working on techniques that focus on interoperability.

REFERENCES

[1] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)," *IEEE Access*, vol. 8, pp. 142642–142668, 2020.

[2] K. Kancharla, V. Kamble, and M. Kapoor, "Handwritten signature recognition: A convolutional neural network approach," in *Proc. Int. Conf. Adv. Comput. Telecommun. (ICACAT)*, Dec. 2018, pp. 1–5.

[3] O. Henniger, D. Muramatsu, T. Matsumoto, I. Yoshimura, and M. Yoshimura, *Signature Recognition*. Boston, MA, USA: Springer, 2009, pp. 1196–1205.

[4] M. M. Al-Taee, S. B. H. Neji, and M. Frikha, "Handwritten recognition: A survey," in *Proc. IEEE 4th Int. Conf. Image Process., Appl. Syst. (IPAS)*, Dec. 2020, pp. 199–205.

[5] S. Ahlawat and A. Choudhary, "Hybrid CNN-SVM classifier for handwritten digit recognition," *Proc. Comput. Sci.*, vol. 167, pp. 2554–2560, Jan. 2020.

[6] A. Shrivastava, I. Jaggi, S. Gupta, and D. Gupta, "Handwritten digit recognition using machine learning: A review," in *Proc. 2nd Int. Conf. Power Energy, Environ. Intell. Control (PEEIC)*, Oct. 2019, pp. 322–326.

[7] S. Boroojerdi and G. Rudolph, "Handwritten multi-digit recognition with machine learning," in *Proc. Intermountain Eng., Technol. Comput. (IETC)*, May 2022, pp. 1–6.

[8] J. Hou, H. Zeng, L. Cai, J. Zhu, J. Cao, and J. Hou, "Handwritten numeral recognition using multi-task learning," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS)*, Nov. 2017, pp. 155–158.

[9] S. Thomas, "A study of representation learning for handwritten numeral recognition of multilingual data set," in *Information and Communication Technology for Sustainable Development*, D. K. Mishra, M. K. Nayak, and A. Joshi, Eds. Singapore: Springer, 2018, pp. 475–482.

[10] W. Li, X. He, C. Tang, K. Wu, X. Chen, S. Yu, Y. Lei, Y. Fang, and Y. Song, "Handwritten numbers and English characters recognition system," in *Intelligent Data Analysis and Applications*, J.-S. Pan, V. Snášel, T.-W. Sung, and X. D. Wang, Eds. Cham, Switzerland: Springer, 2017, pp. 145–154.

[11] M. Jabde, C. Patil, S. Mali, and A. Vibhute, "Comparative study of machine learning and deep learning classifiers on handwritten numeral recognition," in *Proc. Int. Symp. Intell. Inform.*, S. M. Thampi, J. Mukhopadhyay, M. Paprzycki, and K.-C. Li, Eds. Singapore: Springer, 2023, pp. 123–137.

[12] L. M. Seng, B. B. C. Chiang, Z. A. A. Salam, G. Y. Tan, and H. T. Cha, "MNIST handwritten digit recognition with different CNN architectures," *J. Appl. Technol. Innov.*, vol. 5, no. 1, pp. 7–10, 2021.

[13] R. V. K. Reddy, B. S. Rao, and K. P. Raju, "Handwritten Hindi digits recognition using convolutional neural network with RMSprop optimization," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2018, pp. 45–51.

[14] M. H. Ahamed, S. M. I. Alam, and M. Islam, "SVM based real time hand written digit recognition system," in *Proc. Int. Conf. Eng. Res. Educ. School Appl. Sci. Technol.*, Jan. 2019, pp. 1–7.

[15] U. R. Babu, Y. Venkateswarlu, and A. K. Chintha, "Handwritten digit recognition using K-nearest neighbour classifier," in *Proc. World Congr. Comput. Commun. Technol.*, Feb. 2014, pp. 60–65.

[16] D. S. Prashanth, R. V. K. Mehta, and N. Sharma, "Classification of handwritten Devanagari number—An analysis of pattern recognition tool using neural network and CNN," *Proc. Comput. Sci.*, vol. 167, pp. 2445–2457, Jan. 2020.

[17] A. K. Pant, S. P. Panday, and S. R. Joshi, "Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks," in *Proc. 3rd Asian Himalayas Int. Conf. Internet*, 2012, pp. 1–5. Accessed: Jul. 7, 2023. [Online]. Available: https://www.kaggle.com/datasets/ashokpant/devanagari-character-dataset

[18] M. Mhapsekar, P. Mhapsekar, A. Mhatre, and V. Sawant, "Implementation of residual network (ResNet) for Devanagari handwritten character recognition," in *Advanced Computing Technologies and Applications*, H. Vasudevan, A. Michalas, N. Shekokar, and M. Narvekar, Eds. Singapore: Springer, 2020, pp. 137–148.

[19] S. Acharya, A. K. Pant, and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," in *Proc. 9th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, pp. 1–6, 2015. Accessed: Jul. 7, 2023. [Online]. Available: https://www.kaggle.com/datasets/rishianand/devanagari-character-set

[20] M. I. Bhat and B. Sharada, "Recognition of handwritten Devanagari numerals by graph representation and Lipschitz embedding," in *Recent Trends in Image Processing and Pattern Recognition*, K. Santosh, M. Hangarge, V. Bevilacqua, and A. Negi, Eds. Singapore: Springer, 2017, pp. 102–110.

[21] V. J. Dongre and V. H. Mankar, "Devanagari offline handwritten numeral and character recognition using multiple features and neural network classifier," in *Proc. 2nd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2015, pp. 425–431.

[22] S. Arya, I. Chhabra, and G. S. Lehal, "Integrated feature exploration for handwritten Devanagari numeral recognition," in *Proc. 2nd Int. Conf. Comput. Vis. Image Process.*, B. B. Chaudhuri, M. S. Kankanhalli, and B. Raman, Eds. Singapore: Springer, 2018, pp. 145–157.

[23] A. Trivedi, S. Srivastava, A. Mishra, A. Shukla, and R. Tiwari, "Hybrid evolutionary approach for Devanagari handwritten numeral recognition using convolutional neural network," *Proc. Comput. Sci.*, vol. 125, pp. 525–532, Jan. 2018.

[24] R. Pramanik, P. Dansena, and S. Bag, "A study on the effect of CNN-based transfer learning on handwritten indic and mixed numeral recognition," in *Document Analysis and Recognition*, S. Sundaram and G. Harit, Eds. Singapore: Springer, 2019, pp. 41–51.

[25] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICCS)*, May 2019, pp. 1255–1260.

[26] IBM. *What is the K-Nearest Neighbors Algorithm?* Accessed: Jul. 7, 2023. [Online]. Available: https://www.ibm.com/topics/knn

[27] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[28] M. K. Hossen, "Heart disease prediction using machine learning techniques," *J. Appl. Technol. Innov.*, vol. 5, no. 3, pp. 146–154, 2022.

[29] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, 1995, vol. 3361, no. 10, p. 1995.

[30] D. Vo, B. Ngo, T. Nguyen, T. Nguyen, and T. Nguyen, "A convolutional neural network with the VGG-16 model for classifying human brain tumor," in *Proc. 6th Int. Conf. Green Technol. Sustain. Develop. (GTSD)*, Jul. 2022, pp. 714–719.
[31] N. Veni and J. Manjula, "VGG-16 architecture for MRI brain tumor image classification," in *Futuristic Communication and Network Technologies*, N. Subhashini, M. A. G. Ezra, and S.-K. Liaw, Eds. Singapore: Springer, 2023, pp. 319–328.
[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014, *arXiv:1409.4842*.
[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[34] B. Koonce, *ResNet 50*. Berkeley, CA, USA: Apress, 2021, pp. 63–72.
[35] R. I. Bendjillali, M. Beladgham, K. Merit, and A. Taleb-Ahmed, "Illumination-robust face recognition based on deep convolutional neural networks architectures," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 18, no. 2, p. 1015, May 2020.

AGASTYA GUMMARAJU received the B.Tech. degree in computer and communication engineering from the Manipal Institute of Technology, Manipal, India, in 2023. His current research interests include natural language processing, computer vision, and data engineering.

AJITHA K. B. SHENOY received the M.Tech. degree in computer and information science from the Cochin University of Science and Technology and the Ph.D. degree from the Department of Computer Science and Engineering, Indian Institute of Technology Kanpur. He is currently a Professor with the Department of Information and Communication Technology, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal. His current research interests include algorithms, machine learning algorithms, computational complexity theory, artificial intelligence, randomized local search heuristics, evolutionary algorithms, computer vision, and data compression.

SMITHA N. PAI (Senior Member, IEEE) received the bachelor's degree from MCE, Hassan, and the master's and Ph.D. degrees from the Manipal Institute of Technology (MIT), Manipal. She is currently a Professor with the Department of Information and Communication Technology, MIT, Manipal Academy of Higher Education. Her current research interests include wireless sensor networks, machine learning, and computer vision. She has a good number of conferences and journals to her credit in this area.