2024 4th International Conference on Pervasive Computing and Social Networking (ICPCSN) | 979-8-3503-8634-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICPCSN62568.2024.00019

A Novel Model for Recognising Handwritten Devanagari Numerals using Machine Learning

Ch.Prathima
School of Computing
Mohan babu university (formerly Sree Vidyanikethan Engineering College),
Tirupati, India
chilukuriprathi@gmail.com

Ramprakash Reddy Arava
Assistant Professor
Department of CSE
KSRM College of Engineering
Prakash.Cse.502@gmail.com

K Sevitha
UG Scholar, Department of CSSE
*Sree Vidyanikethan Engineering College,*Tirupati, India
sevithak03@gmail.com

G Manikanth
UG Scholar, Department of CSSE
*Sree Vidyanikethan Engineering College,*Tirupati, India
manikanth2511@gmail.com

D Vinay
UG Scholar, Department of CSSE
*Sree Vidyanikethan Engineering College,*Tirupati, India
vinaydeenadayal@gmail.com

C Surya
UG Scholar, Department of CSSE
*Sree Vidyanikethan Engineering College,*Tirupati, India
chittemsurya0095@gmail.com

***Abstract*— This research study performs a comprehensive comparative analysis aimed at developing effective machine learning models for classifying handwritten Devanagari numerals. The research focuses on evaluating the performance of various models to determine the most accurate classification approach. Initiating with data pre-processing, including feature extraction and normalization, the data is prepared for model training and assessment. A diverse range of machine learning models, from traditional methods like Support Vector Machines (SVM) to advanced techniques such as Random Forests, K-Nearest Neighbors, and Convolutional Neural Networks, are considered for the comparative analysis, ensuring a thorough assessment of classification capabilities. Cross-validation techniques are employed during model training and testing to enhance reliability. Statistical tests are utilized to assess the performance variations among models, enhancing the robustness of the analysis. Visual representations of performance metrics and comparison results offer clear insights. This research study aims to identify the most suitable machine learning model for handwritten Devanagari numeral classification, potentially advancing character recognition systems and linguistic applications.**

Keywords*— *Handwritten character recognition, Machine learning, Support Vector Machine(SVM), Data augmentation, Transfer learning, Digital document processing

I. INTRODUCTION

Text recognition involves a computer's ability to comprehend written text, utilizing machine learning algorithms and frameworks for the interpretation and assessment of textual content. In this study, the dataset was generated by using a paint application. Input images were created from numbers 0 to 9 using an oil brush, chosen for its versatility in accurately predicting the digits. Recognizing handwritten Devanagari numerals poses a formidable challenge, given the script's complexity. This study introduces a novel approach leveraging machine learning techniques, particularly Support Vector Machines(SVMs)[11], K-Nearest-Neighbours (KNN), for feature extraction and classification. The proposed model is trained and tested on a substantial dataset, emphasizing robustness and generalization across diverse writing styles.[1] To tackle limited training data and variations, this study employs data augmentation and transfer learning. Handwriting recognition is a crucial process in computer science, finding applications in Optical character recognition (OCR) systems, signature authentication and document scanning . The evolution of digit categorization is evident in its application in banking for fraud prevention, healthcare for patient record digitization, and government for information extraction from official documents.[2] This study reflects the adaptability and significance of handwriting recognition in addressing contemporary challenges. To generate a dataset, our initial step involves assigning a value of 1 to the selected region and 0 to the background. Consequently, our dataset will comprise solely of these two values: 0 and 1. It's important to note that pixel values typically span from 0 to 255. Conventionally, 0 signifies black while 255 denotes white in most scenarios.

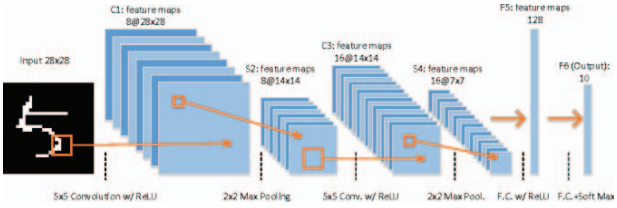

Fig. 1. System Architecture

II. RELATED WORKS

Ajitha K (2023) focused on centered around evaluating various machine learning models to determine their effectiveness in classifying handwritten digits in the Devanagari script. [1] The CNN model proposed in this study achieved an impressive accuracy of 99.522%. Furthermore, it surpassed the performance of other CNN models evaluated for Devanagari numerals classification.

K. Kancharla (ICACAT), Dec. 2018 focused on Handwritten Signature Recognition stands as a crucial behavioral biometric utilized across numerous identification and authentication applications. It predominantly operates through two distinct methods: on-line and off-line recognition. On-line recognition entails a dynamic approach, incorporating parameters such as writing pace, variations in stylus direction, and the count of pen ups and pen downs throughout the signature-writing process.

SM Shamim [3] It focuses on recognizing a digit through minist dataset. They concluded that highest precision, recall and f1score for Support Vector machine.

M. M. Al-Taee [4] focused on Handwritten recognition has garnered significant interest within the fields of pattern recognition and image processing over recent decades [13].

S. Gupta [5] In this paper, A comparison of different approaches for recognizing handwritten numerals utilizing solely the MNIST database is conducted. Over time, numerous researchers have introduced novel techniques for digit recognition, contributing to the enhancement of our daily routines [12].

S. Boroojerdi ‘‘Handwritten multi-digit recognition with machine learning’’ in (IETC). Offline recognition of handwritten digits poses a persistent challenge, with solutions thus far remaining only partially effective. This study delves into the realm of offline handwritten multi-digit recognition, employing three distinct algorithms—Decision Trees, Multilayer Perceptrons, and Random Forest—utilizing the MNIST dataset. Among these, Random Forest emerged as the most promising, boasting an impressive 96% accuracy alongside reasonable runtime efficiency.

III. PROPOSED WORK

The suggested endeavor to employ machine learning for the identification of Manually scripted Devanagari numerals constitutes a significant advancement in the realm of character recognition [14] particularly within the intricate context of challenging scripts like Devanagari.

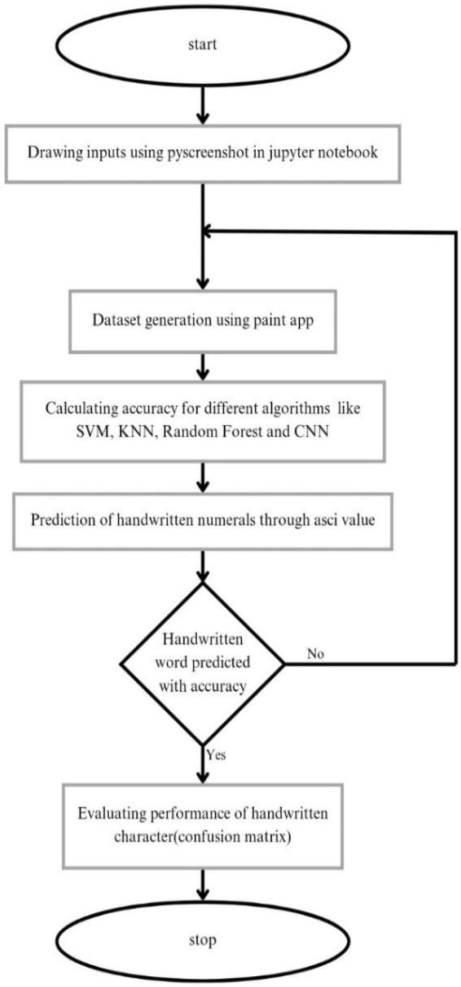

Fig.2. Process of handwritten recognizing

This study centers on assessing the appropriateness of various machine learning frameworks for classifying Manually written numerals using the Devanagari script. In this research, the study will evaluate various models, including K-Nearest-Neighbors (KNNs), Support Vector Machine (svms), Random Forest (rf), and Convolutional Neural Networks(cnn). Surprisingly, the proposed KNN model is implemented on F1 score.

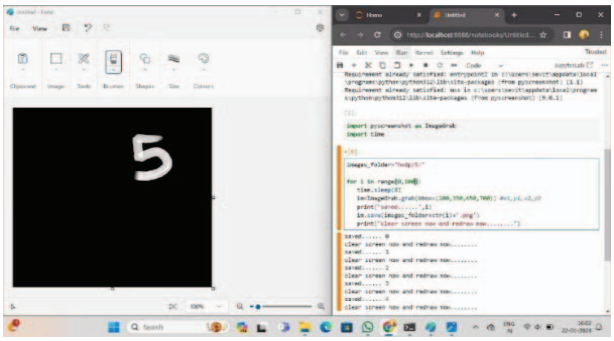

Fig. 3. Dataset generation using paint

KNN Algorithm:

The K-Nearest Neighbors (KNN) algorithm, a simple yet potent supervised machine learning technique, finds application in tasks encompassing both classification and regression. [6] At its essence, it entails producing predictions based on either the predominant class or the average of the KNN data points within the feature space. Dataset creation is achieved using a paint application. General formula is

$$\sum_{i=1}^{k} |xi - yi|$$

CSV File Setup: This work begins by importing the necessary libraries: cv2 for computer vision operations, "csv" for handling CSV files, and glob for file path pattern matching [8]. It sets up the header for the CSV file, where each image will be represented as a row with a label and pixel values. A CSV file named 'demo1.csv' is created, and the header is written to it.

Image Processing and CSV Data Generation: For each digit label (0 to 9), the code uses glob to get a list of image file paths in the corresponding subdirectories. For each image, it reads the image, converts it to grayscale, applies a Gaussian blur, and resizes it to 28x28 pixels. Based on the pixel value it predicts the digit. The label and flattened pixel values are then appended to the 'demo1.csv' file.

Data Loading and Shuffling: The code imports pandas for data manipulation and shuffle from scikit-learn to shuffle the dataset. It reads the CSV file into a Pandas Data Frame and shuffles the rows randomly.

Data Splitting: The features (X) comprise all columns excluding the "label" column, while the target variable (Y) is specifically the "label" column. The dataset undergoes division into training and testing sets facilitated by the 'train_test_split' function from scikit-learn.

Support Vector Machine algorithm:

The dataset is formed by employing a paint application to produce images, from which a CSV file is crafted using pixel values. Supervised machine learning tasks, specifically classification or regression, can be effectively addressed using SVM as an algorithm. SVM exhibit versatility in handling both linear and non-linear classification as well as regression tasks through the utilization of suitable kernel functions. Typical kernel options encompass linear, polynomial, and radial basis function (RBF). Hyper plane equation is

$$W.X+b=0$$

Importing Libraries: This work starts by importing necessary libraries. Joblib is imported for saving the trained SVM model, and SVC (Support Vector Classification) is imported from scikit-learn for implementing the Support Vector Machine classifier.[7]

Creating an SVM Classifier: An SVM classifier is created using the SVC class with a linear kernel. The random state parameter is set to 5 for reproducibility, ensuring that the same sequence of random numbers is generated if the code is run multiple times.

Training the Classifier: The created SVM classifier ('classifier') is trained using the `fit` method with the training data and where train_x represents the features (pixel values) and train_y represents the corresponding labels.

Saving the Model: The trained SVM model is then saved using the joblib.dump function. The model is saved in a file named "digit_recognizer" within a directory named model. This allows you to persist the trained model for later use without having to retrain it every time.

Random Forest algorithm:

A Random Forest is a popular machine learning algorithm used for both classification and regression tasks.[9] This algorithm focusses on following steps:

Importing Necessary Libraries: Here, we import the 'RandomForestClassifier' class from the ensemble module of scikit-learn, which contains ensemble-based learning algorithms, including Random Forest. We also import accuracy_score from 'sklearn.metrics' which will be used to calculate the accuracy of the classifier.

Creating the Random Forest Classifier: Creates a RF classifier instance, with the 'n_estimators' parameter determining the number of trees within the forest.

Training the Classifier: The Random Forest classifier is trained on the provided training data. Usually, 'train_x' comprises the features of the training dataset, while 'train_y' contains the corresponding labels.

Making Predictions: In this step, the Random Forest classifier, which has been trained previously, is employed to predict outcomes on the test dataset denoted as 'test_x'. The resulting predictions are then stored in the variable 'y_pred'.

Calculating Accuracy and Confusion Matrix: In this context, we're presenting both the confusion matrix and accuracy metrics. The confusion matrix serves as a crucial instrument for assessing the performance of a classification model on a test dataset with known true values. It provides a comprehensive breakdown, revealing the counts of tp, fp, tn, and fn, thereby offering significant insights into the classifier's efficacy.

CNN algorithm:

Importing Necessary Libraries: Keras are imported. Keras is TensorFlow's high-level neural networks API, which

provides a convenient way to define and train deep learning models.[10].

Loading and Preprocessing the Data: Own dataset is loaded and then preprocessed will result Reshaping the input images to have a single channel (grayscale) and adjusting pixel values to be between 0 and 1.

Compiling the Model: The model undergoes compilation using the Adam optimizer, categorical cross-entropy loss function (ideal for multi-class classification tasks), and accuracy serves as the metric for evaluation.

Training the Model: The model undergoes training on the provided training data ('x_train', 'y_train') for a total of 5 epochs, utilizing a batch size of 64 for optimization. Validation data is designated through the `validation_split` parameter during the training process.

Evaluating the Model: The trained model is evaluated on the test data ('x_test', 'y_test'). Predicted probabilities are obtained using `predict` method, then converted to predicted labels. True labels are also converted from one-hot encoded format. Finally, accuracy is calculated and printed.

IV. RESULTS AND DISCUSSION

The Support Vector Machine (SVM) model achieves the highest accuracy at 99.5%. Prediction analysis can be conducted through the Paint app using essential packages like joblib, pyscreenshot, numpy, time, and cv2. The SVM model predicts the digit by utilizing the predict method with the prepared input array, displaying the predicted digit on the console. Results are presented in the form of the original image, annotated with the predicted digit using "cv2.putText". A window named "Result" is created, showcasing the annotated image. The program awaits a key press for a specified duration (10 seconds in this case) using "cv2.waitKey". If the pressed key is Enter (13), the loop continues; otherwise, it breaks out of the loop. Upon termination of the loop, typically triggered by pressing Enter, the OpenCV windows are closed using "cv2.destroyAllWindows()".

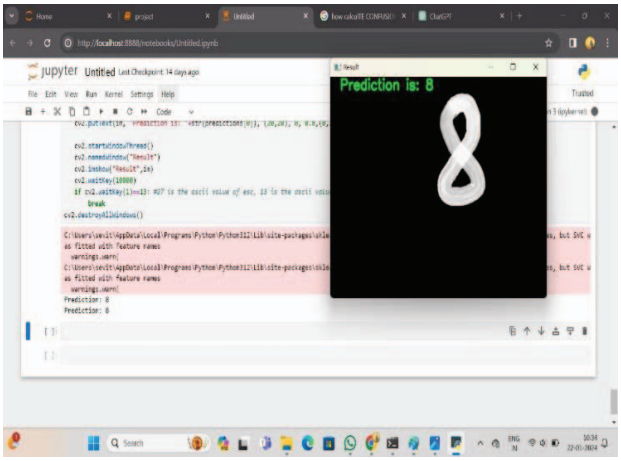

Fig.4. prediction of 8

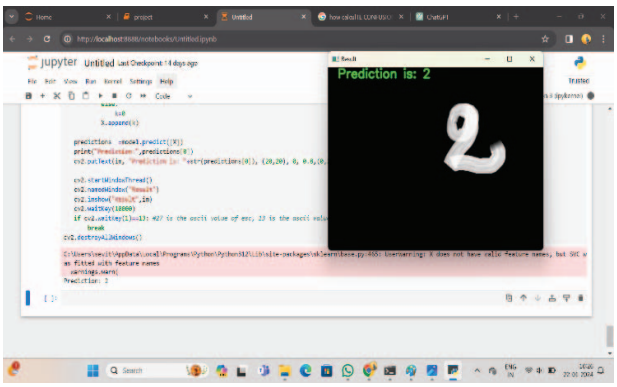

Fig. 5. Prediction of 2

By leveraging the inherent strengths of convolutional neural networks (CNNs), the proposed model surpasses both SVM and KNN. CNNs excel in capturing local patterns and relationships, demonstrating characteristics such as translational invariance and non-linearity. When dealing with a relatively compact dataset and the straightforward nature of grayscale images (with dimensions of 28 × 28 and 32 × 32), the diminished accuracy observed in higher-capacity models can be linked to their heightened capacity to capture intricate patterns.

However, achieving optimal performance on smaller datasets with such models may necessitate additional data or thorough hyperparameter tuning. The heightened complexity of such models could pose challenges in effectively learning from the limited information available, consequently leading to a marginally reduced accuracy. Figures 4, 5, and 6 illustrate digit predictions and compare various frameworks along with their corresponding accuracies.

TABLE. 1. COMPARISON OF ACCURACY FOR USED FRAMEWORKS

Algorithm	Accuracy
SVM	0.995
KNN	0.955
Random Forest	0.985
CNN	0.987

```
Confusion Matrix:
[[22  0  0  0  0  0  0  0  0  0]
 [ 0 14  0  0  0  1  0  0  0  0]
 [ 0  0 24  0  0  0  0  0  0  0]
 [ 0  0  0 22  0  0  0  0  0  0]
 [ 0  2  0  0 16  0  2  0  0  0]
 [ 0  0  0  1  0 14  0  0  0  0]
 [ 0  1  0  0  0  0 25  0  1  0]
 [ 0  0  0  0  0  0  0 24  0  0]
 [ 0  0  0  0  0  0  0  0 14  0]
 [ 0  1  0  0  0  0  0  0  0 16]]
```

Fig. 6. Confusion Matrix of KNN

```
Confusion Matrix:
[[22  0  0  0  0  0  0  0  0  0]
 [ 0 15  0  0  0  0  0  0  0  0]
 [ 0  0 24  0  0  0  0  0  0  0]
 [ 0  1  2 19  0  0  0  0  0  0]
 [ 0  0  0  2 18  0  0  0  0  0]
 [ 0  0  0  0  0 15  0  0  0  0]
 [ 0  1  0  1  0  0 25  0  0  0]
 [ 1  0  0  0  0  0  0 23  0  0]
 [ 0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  0 17]]
```

Fig. 7. Confusion Matrix of SVM

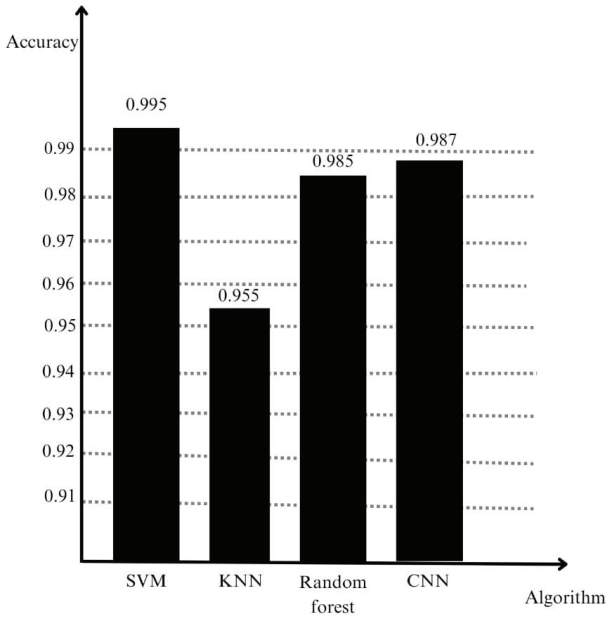

Fig. 8. Comparison of Algorithm Accuracies

V. CONCLUSION

In conclusion, the recognition of handwritten Devanagari numerals using machine learning algorithms presents an opportunity to apply advanced techniques in a multilingual and script-specific context. Through an extensive literature review, it is evident that the application of k-Nearest Neighbors (KNN) has shown promising results in character recognition tasks, including those involving the Devanagari script. Comparative analysis of various algorithms, including but not limited to SVM, Random Forest, k-Nearest Neighbors and CNN have revealed their unique strengths and limitations in recognizing handwritten Devanagari numerals. Recognizing handwritten Devanagari numerals has provided valuable insights into the efficiency of different models. The recommended model, leveraging the strengths of Support Vector Machines(SVMs), KNN and Random Forest (RF). The observed lower accuracy in higher-capacity models suggests the need for careful consideration of dataset size and simplicity. Despite the trails posed by the relatively small dataset and the simplicity of grayscale images, our findings emphasize the importance of balancing model complexity with the available data, offering valuable considerations for future endeavors in the domain of handwritten Devanagari numeral recognition.

VI. FUTURE WORK

Expansion to Full Devanagari Script: Extending the recognition model to cover the entire Devanagari script, including consonants, vowels, and compound characters, to create a comprehensive recognition system for the script as a whole. Multilingual Recognition: Exploring the feasibility of

adapting the recognition model to recognize numerals in other Indic scripts, such as Bengali, Gujarati, or Tamil, to create a multilingual character recognition system.

REFERENCES

[1] Performance Comparison of Machine Learning Models for Handwritten Devanagari Numerals Classification Agastya Gummaraju; Ajitha K. B. Shenoy; Smitha N. Pai IEEE Access Year: 2023 | Volume: 11 | Journal Article | Publisher: IEEE

[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.

[3] Handwritten digit recognition using machine learning algorithms SM Shamim, MBA Miah, A Sarker, M Rana, A Al Jobair Global Journal Of Computer Science And Technology, 2018•researchgate.net

[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] 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.

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

[7] 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

[8] Liu, C., Fujisawa, H., "Handwritten digit recognition: Benchmarking of state-of-the-art techniques," Pattern Recognition, 2003.

[9] Fialoke, S., "Predicting Digits from their handwritten images, (16.10.2020), http://suruchifialoke.com/2017-06-15- predicting-digits_tensorflow

[10] Arica, N., Vural, Y., F.T., "An overview of character recognition focused on offline handwriting," IEE Transactions on Systems Man and Cybernetics, 2001

[11] Handwritten digit recognition using machine learning algorithms SM Shamim, MBA Miah, A Sarker, M Rana, A Al Jobair Global Journal Of Computer Science And Technology, 2018•researchgate.net

[12] Handwritten digits identification using MNIST database via machine learning models B Gope, S Pande, N Karale, S Dharmale, P Umekar, IOP conference series: materials science and engineering, 2021.

[13] Impact of Deep Learning on Localizing and Recognizing Handwritten Text in Lecture Videos Media, L.H., Ramani, KInternational Journal of Advanced Computer Science and Apllications 2021, 12(4), pp. 336-344.

[14] Muppalaneni, N.B. (2021). Deep Learning Approach for Prediction of Handwritten Telugu Vowels. In: Reddy, A., Marla, D., Favorskaya, M.N., Satapathy, S.C. (eds) Intelligent Manufacturing and Energy Sustainability. Smart Innovation, Systems and Technologies, vol 213. Springer, Singapore. https://doi.org/10.1007/978-981-33-4443-3_35.

[15] K. N. K. Reddy, K. N. V. S. Divyanjali, L. J. Royal, M. H. Chandana and D. M. Krishna, "A Novel Model for Prediction of Next Word using Machine Learning," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 393-400, doi: 10.1109/ICICCS56967.2023.10142737.