Handwritten Digit Recognition using Machine Learning Algorithms

[1] Manisha Patel , [2] Tulsidas Nakrani
[1]Research Scholar, Sankalchand Patel University, [2]Sankalchand Patel University
[1]manishaatwork84@gmail.com, [2]tvnakrani.mca@spcevng.ac.in

***Abstract*: Recognition of handwritten characters and numbers is one of the practically important tasks in pattern. Numeric recognition applications include mail sorting, bank check processing, form data entry, etc. The main purpose of this paper is to demonstrate and present the work related to handwritten digit recognition, and the challenge is to be able to develop an efficient algorithm capable of recognizing handwritten digits sent users through scanners, tablets and other digital devices.**
In this recognition exercise, the numbers are written or written inaccurately because they differ in shape or size; for this reason, feature extraction and segmentation of handwritten numerical writing is difficult. This article presents an offline handwriting digit recognition approach based on several machine learning techniques.The main goal of this document is to provide efficient and reliable approaches to handwritten digit recognition. Several machine learning algorithms have been used for digit recognition using MNIST, most notably multilayer perceptron, and support vector machine, naive Bayes, Bayesian network, random forest, J48 and random tree.

***Keywords:* pattern recognition, handwritten recognition, digit recognition, machine learning, MNIST, off-line handwritten recognition, machine learning algorithm, neural network, classification algorithm**.

I. INTRODUCTION

Handwritten digit recognition is the ability of a computer to recognize handwritten digits from various sources such as images, documents, touch screens, etc. and classify them into 10 predefined classes (0-9). This has been an unrestricted research topic in deep learning. Digit recognition has many applications such as license plate recognition, mail sorting, bank check processing, etc [2]. There are several problems when trying to solve this problem. Ink drawings are not always the same size, thickness, or orientation and position relative to margins. Our goal was to implement a model classification method for recognizing handwritten digits represented in the MNIST dataset of handwritten digit images (0-9).
When recognizing handwritten digits, we face many problems due to the different writing styles of different people as it is not OCR. This study provides a comprehensive comparison of various machine learning algorithms for handwritten digit recognition. For this, we used a support vector machine, a multilayer perceptron, and a convolutional neural network. These algorithms are compared based on their accuracy, errors, and test training time, as evidenced by graphs and charts built using matplotlib for visualization [3].

The accuracy of any model is the more accurate, the better the models make decisions. Models with low accuracy are not suitable for real applications. For example, for an automated system for processing bank checks, where the system recognizes the amount and date on the check, high accuracy is very important. If the system does not properly recognize a digit, serious unwanted damage can result. That is why these real world applications require an algorithm with high accuracy. Therefore, we compare different algorithms based on their accuracy so that the most

accurate algorithm with the lowest error rate can be used in various handwritten digit recognition applications [3].
Therefore, hard feature extraction is very important for improving the performance of a handwriting recognition system. At present, handwritten digit recognition has gained more concentration in the field of pattern recognition system, which has led to its application in various fields. In the coming days, a character recognition system could be the cornerstone for launching a paperless environment by digitizing and processing existing paper documents [11].

Handwritten digit datasets are vague in nature because there may not always be crisp, perfectly straight lines. The main goal of digit recognition is feature extraction to remove redundancy from the data and achieve a more efficient embodiment of the image of a word using a set of numeric attributes. It takes care of extracting most of the important information from the raw image data.Also, curves don't have to be as smooth as printed characters. In addition, the character data set can be drawn in different sizes and orientations, which must always be written on the guide at a vertical or vertical point. Therefore, an efficient handwriting recognition system can be designed with these limitations in mind.It's tedious enough to sometimes identify handwritten characters, as it can be seen that most people can't even recognize their own written scripts. Therefore, there is a limitation for the writer to write, presumably to recognize handwritten documents [3]-[7].

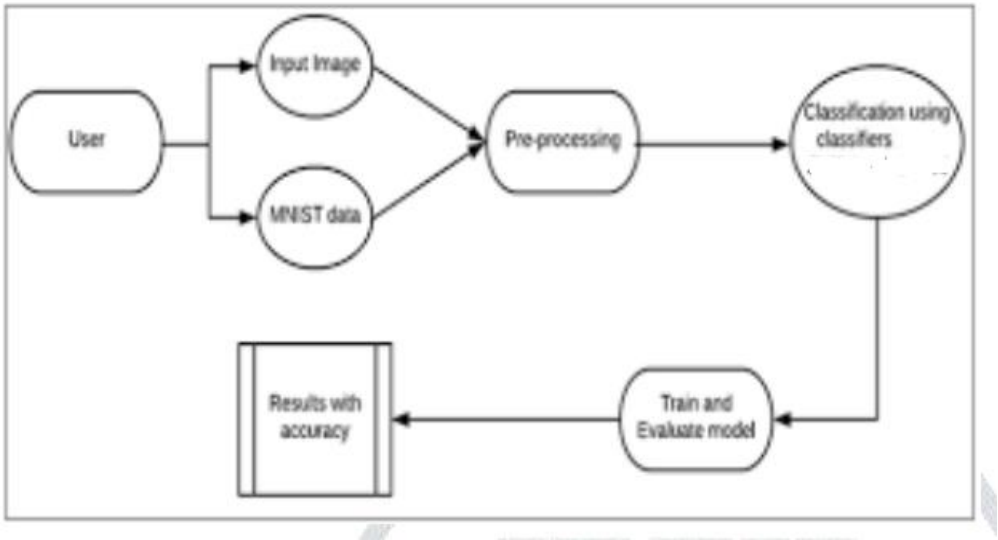

Figure 1: This figure shows the steps of process for Handwritten Digit Recognition system

II. METHOD AND MATERIAL

A. Multilayer Perception

A multilayer perceptron (MLP) is a class of feed forward artificial neural networks (ANNs). It consists of three layers: an input layer, a hidden layer, and an output layer. Each layer is made up of several nodes, also formally called neurons, and each node is interconnected with every other node in the next layer. There are 3 levels in the basic MLP, but the number of hidden levels can be increased to any number depending on the task without any limit on the number of nodes. The number of nodes in the input and output layers depends on the number of visible attributes and classes in the dataset, respectively. The specific number of hidden layers or the number of nodes in a hidden layer is difficult to determine due to the irregular nature of the model and is therefore chosen experimentally. Each hidden layer of a model can have different trigger functions to

process. For learning purposes, it uses a supervised learning technique called backpropagation. In MLP, a node's connection consists of a weight that is adjusted to be in sync with each connection during model training [3].

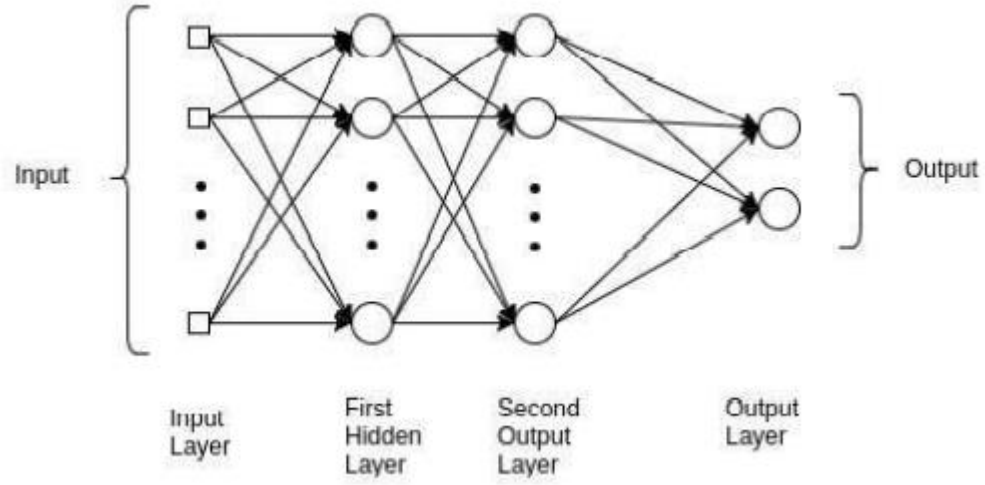

Figure 2: This figure illustrates the basic architecture of the Multilayer perceptron with variable specification of the network

B. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm. While we usually display the data items in an n-dimensional space, where n is the number of features, a specific coordinate represents the value of the feature, we perform the classification by finding a hyperplane that distinguishes the two classes. It will choose a hyperplane that properly separates the classes. The SVM selects the edge vectors which help in creating the hyperplane. These edge cases are called support vector machines and hence the algorithm is called support vector machine. There are basically two types of SVM: linear and non-linear SVM. In this article, we have used linear SVM for handwritten digit recognition [3].

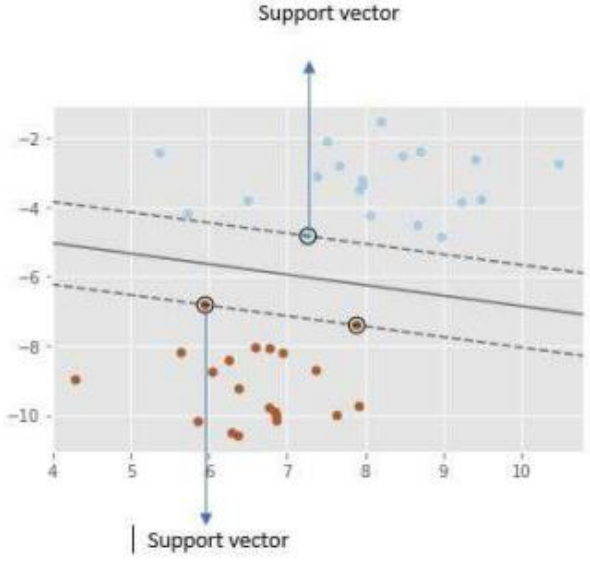

Figure 3. This image describes the working mechanism of SVM Classification with supporting vectors and hyperplanes

C. J48

The algorithm is an extension of the C4.5 decision tree algorithm. There are many options for tree pruning in the case of the J48 algorithm. Convenient classification algorithms in MNIST try to refine or thin out the results. This method will help us get more general results and can also be used to eliminate potential fitting issues. J48 helps classify recursively until each leaf is truncated, i.e. classified as closely related to the data. Thus, it will help ensure accuracy even if too many rules are created. However, pruning will cause the model to be less accurate on the training data. This is because pruning uses various means to weaken the specificity of the decision tree, which will hopefully improve its performance on test data. The whole concept is to generalize the decision tree more and more until a balance is found between precision and flexibility. J48 uses two cutting methods. The first is known as subtree replacement. From this it is concluded that the nodes in the decision tree can be replaced by a leaf, which will reduce the number of tests on a certain path. This process starts with the leaves of a fully formed tree and attempts to return to the root [8]. The second category of reduction adopted in J48 is called the ascending subtree. In this regard, the node can be moved up to the root of the tree, replacing other nodes in a different way. Repeated subtree growth has little effect on decision tree models. There is generally no clear way to predict the usefulness of this option, although it may be desirable to try turning it off if the induction process takes a long time. This is because growing a subtree can be somewhat computationally complex. Error rates are needed in order to make effective judgments about which parts of the tree should be enlarged or replaced. There are several ways to do this. The easiest way is to reserve part of the training data for testing on the decision tree. The reserved part can then be taken as evidence for the decision tree, which helps reduce the chance of overfitting. This method has been found to reduce decimation errors. While the approach is simple, it also reduces the overall amount of data available to train the model. For particularly small datasets, it may be desirable to avoid using reduced error decimation [1]-[9].

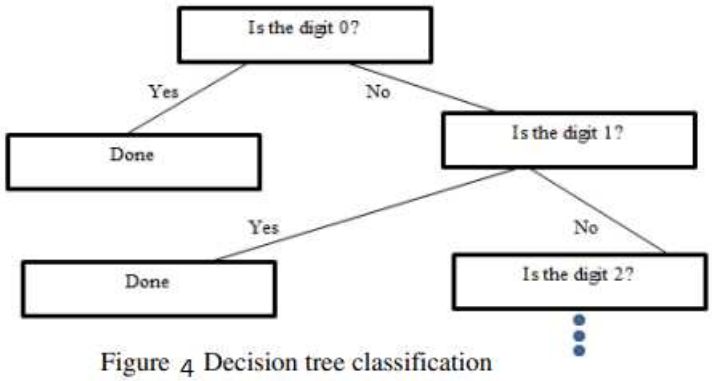

Figure 4 Decision tree classification

D. Random Forest Algorithm

As the name suggests, "A random forest is a classifier that contains a number of decision trees for various subsets of a specified dataset and uses the mean to improve the prediction accuracy of that dataset." Instead of relying on a decision tree, Random Forest takes a prediction from each tree and predicts the final outcome based on the majority of the prediction votes.

More trees in the forest results in greater accuracy and prevents the overfitting problem. [10]

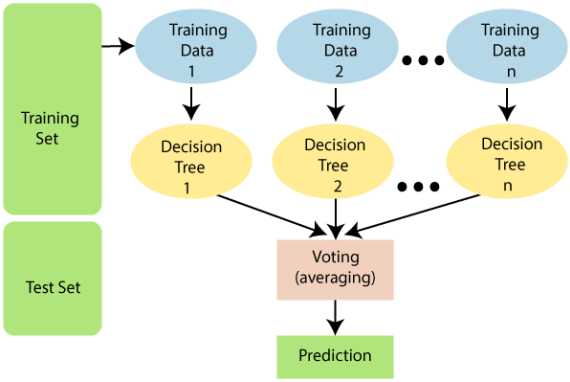

Figure 5: The above diagram explains the working of the Random Forest algorithm:

E. Naïve Bays

The Naive Bayes classifier offers a simple method to represent and learn probabilistic knowledge with clear semantics. It is defined as naive because it is based on two important simplifying assumptions that predictive attributes are conditionally self-sufficient for a given class, and no hidden attributes are considered to affect the prediction method. It is a probabilistic classifier based on Bayes' theorem with robust and naive independence assumptions. It is one of the best basic text classification approaches with numerous applications for personal email sorting, email spam detection, sexually explicit content detection, document categorization, sentiment detection, language detection [1].

F. Bayes Net

Bayesian networks are a powerful probabilistic representation and much attention has been paid to their use for classification. It reflects the states of certain parts of the simulated world and describes how these states are related by probabilities. A Bayesian network is a graphical model that encodes probabilistic relationships between variables of interest. When used in conjunction with statistical methods, the graph model has several advantages for data analysis. First, because the model encodes dependencies between all variables, it easily handles situations where some data records are missing. Secondly, the Bayesian network can be used to study causal relationships and thus to understand the domain and predict the effects of an intervention. This classifier learns from the training data the conditional probability of each attribute with a given class label [1].

G. Random Tree

The algorithm can solve both regression and classification problems. Random Trees is a set of tree predictors called a forest. The classification works like this: the random tree classifier takes a vector of input features, classifies it with the only tree in the forest, returns the tag of the class with the most "votes". In the case of regression, the classifier's response is the average of the responses across all trees in the forest. In the random tree algorithm, all trees are trained with the same parameters but on different training sets. These sets are generated from the original training set using a bootstrap procedure, and for each training set, the same number of vectors are randomly selected as in the initial set. Vectors are chosen with substitution. That is, some vectors will occur more than once, and some will be absent. Random trees do not require accuracy estimation methods such as cross-validation or bootstrap, or a separate test suite to obtain a training error estimate. The error is evaluated internally during training [1].

H. Dataset Description

Handwritten digit recognition is a broad research topic that provides a comprehensive overview of the field, including core feature sets, training datasets, and algorithms. In contrast to optical character recognition, which is oriented to recognize typewritten output, where special characters can be used, and the variance between characters along with the same font size, font, and attributes is quite small. Feature extraction and classification techniques play an important role in the performance of an offline character recognition system various approaches to feature extraction have been proposed for a character recognition system.

More extensive work has been done on digit recognition in many languages such as English, Chinese, Japanese and Arabic. In India he worked mainly in Devanagari, Tamil, Telugu and Bengali for number recognition. This dataset is divided into two parts: training set and test set.

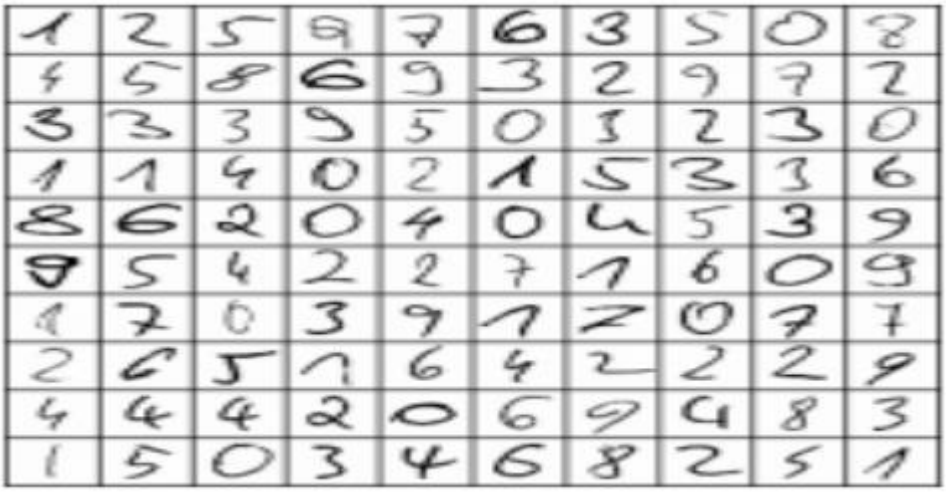

Figure 6:A small portion of handwritten dataset example

There are 1893 samples in the training set and 1796 samples in the test set. Details of the dataset are provided in [1].

III. EXPERIMENTAL TOOL

The Modified Database of the National Institute of Standards and Technology (MNIST) is a large database of handwritten drawings commonly used to train various imaging systems. The database is also widely used for training and testing in the field of machine learning.

It was created by "scrambling" samples from the original NIST datasets. The creators felt that since the NIST training dataset was taken by the US Census Bureau and the test dataset was taken by US high school students, it was not suitable for automated training experiments. In addition, the NIST black and white images were normalized to fit a 28x28 pixel antialiased bounding box that introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 test images. Half of the training set and half of the test set were taken from the NIST training dataset, while the other half of the training set and the other half of the test set were taken from the NIST test dataset. The original creators of the database keep a list of some of the methods tested against it.

In their original paper, they use a support vector machine to achieve an error rate of 0.8%. In 2017, an enhanced MNIST-like dataset called EMNIST was released, which contains 240,000 images of the mnist training datasets and 40,000 images of the MNIST test datasets of handwritten digits and symbols [4]-[6].

IV. EXPERIMENTAL RESULT

For classification by neural networks, we put the values of the characteristic vectors in the neurons of the input layer and know the desired result, we imposed the network in the International Journal of Advanced Science and Technology Vol.50, January 2013 108 converge to the final specific state (supervised learning). Each digit is characterized by a seven-component extraction vector. For network training (MLP multilayer perceptron), 1 set of 10 images and 100 sets of 1000 images, we started with a set of ten images to find the best parameters that maximize the network [5].

Table 1 : Recognition Rate for each Digit with 100 Sets of Images

Types of digits Database	Recognition rate of test	Error rate	Reject rate	Execution Time (s)
0	86.45	13.55	00.00	91.64
1	94.39	05.61	00.00	112.58
2	88.73	11.27	00.00	95.63
3	77.02	22.98	00.00	76.71
4	76.12	23.88	00.00	84.29
5	84.10	15.90	00.00	70.77
6	78.81	21.19	00.00	85.91
7	77.12	22.88	00.00	90.33
8	79.03	20.97	00.00	85.37
9	49.64	50.36	00.00	93.90

V. CONCLUSION

The main goal of this study is to find a representation of isolated handwritten digits that allows them to be recognized effectively. In this article, several machine learning algorithms were used to recognize handwritten numbers. In any recognition process, choosing the right approach to feature extraction and classification is an important issue. The proposed algorithm attempts to account for both factors in terms of accuracy and time complexity. General The highest overall accuracy of 90.37% is achieved with the Multilayer Perceptron recognition process. This work is done as an initial attempt, and the aim of the work is to facilitate the recognition of handwritten numbers without the use of standard classification methods [1].

REFERENCES

1. ”Handwritten Digit Recognition using Machine Learning Algorithms”, S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair
2. “What can a digit recognizer be used for?” online: https : //www.quora.com/W hat−can−a−digit−recognizer− be − used − f or
3. Pashine, Samay, Ritik Dixit, and Rishika Kushwah. "Handwritten digit recognition using machine and deep learning algorithms." arXiv preprint arXiv:2106.12614 (2021).
4. “What is MNIST dataset?”-https://www.engati.com/glossary/mnist-dataset#toc-why-is-mnist-a-good-dataset
5. El Kessab, B., et al. "Extraction method of handwritten digit recognition tested on the mnist database." International Journal of Advanced Science & Technology 50 (2013): 99-110.
6. Gomathy, C. K., Kakamanu Jaya Sairam, and Illuru Yadhu Vamsi. "A THE HANDWRITTEN DIGITS RECOGNITION."
7. Sudarshan Achole Rohan Adat Chitragupt Bangar Siddhant Bhalke “Handwritten Digit Recognition Using Machine Learning” © 2021 JETIR December 2021, Volume 8, Issue 12
8. Nakrani, Tulsidas, et al. "Genetic Algorithm Based Task Scheduling for Load Balancing in Cloud." Data Science and Intelligent Applications. Springer, Singapore, 2021. 283-293.
9. Assegie, Tsehay Admassu, and Pramod Sekharan Nair. "Handwritten digits recognition with decision tree classification: a machine learning approach." International Journal of Electrical and Computer Engineering (IJECE) 9.5 (2019): 4446-4451.
10. Random Forest Online:https://www.javatpoint.com/machine-learning-random-forest-algorithm,July 2022
11. Darji, M. J., & Nakrani, T. MACHINE LEARNING BASED PREDICTION TECHNIQUE FOR STUDENT’S PERFORMANCE.