ABSTRACT

The problem of handwritten digit recognition has long been an open problem in the field of pattern classification. Several studied have shown that Neural Network has a great performance in data classification. The main objective of this paper is to provide efficient and reliable techniques for recognition of handwritten numerals by comparing various existing classification models. This paper compares the performance of five machine learning classifier models namely Neural Network, K Nearest Neighbour (K-NN), Random Forest, Decision Tree and Bagging with gradient boost. Results indicate that K-NN classifier outperform Neural Network with significant improved computational efficiency without sacrificing performance. They both outperformed the other classifiers: Random Forest, Decision Tree and Bagging with gradient boost. We also discovered that as the training data is increasing the accuracy of the classifier is also improved. The result of this paper shows that K-NN has equally high accuracy of 96.7% compared to Neural Network of 96.8%, but K-NN achieves a processing speed with almost 10 times faster. The analysis presented in this paper suggests that the K-NN combined with pre-processing methods is capable of achieving great performance apart from Neural Network when used as a classification algorithm in offline handwritten digit recognition.

Introduction

The problem of handwritten numerals recognition has been widely studied in recent years and the large quantity of preprocessing methods and classification algorithms have been developed. (Wang & Yan, June 2000). However, handwritten numerals recognition is still a challenge for us. The main difficulty of handwritten numerals recognition is the serious variance in size, translation, stroke thickness, rotation and deformation of the numeral image because of handwritten digits are written by different users and their writing style is different from one user to another user. It is even impossible to construct a database, including all the typical samples of the unconstrained numerals. If handwriting is recognized while writing through touchpad using the stylus pen, it's called online handwriting recognition. In this case, handwriting is scanned and then understood by the computer, it is called offline handwriting recognition.

This study focuses on feature extraction and classification. The performance of a classifier can rely as much on the quality of the features as on the classifier itself. Moreover, our study presents an efficient offline handwritten character recognition system based on diagonal features and transitions features using the k-NN classifier. Diagonal and transitions feature of a character have been computed based on the distribution of points on the bitmap image of the character. In this study, we compare the performance of five different machine learning classifiers for recognition of digits. The five classifiers namely neural network, K-Nearest Neighbor, Random Forest, Decision Tree and Bagging with gradient boost. The combination of different

classifiers, such as ANN, and K-NN implanted in a recognition system is also a suitable method of increasing recognition performance, because different classifier combinations have different merits in dealing with discriminant problems. This study train and test K-Nearest Neighbors classifier in RapidMiner for pattern analysis in solving handwritten digit recognition problems using the MNIST database. Although K-NN is a lazy learner algorithm, it's selected because of its high accuracy in digits predicting and training efficiency when compared to other operators like Neural Net, random forest etc. Here, the preprocessing of the dataset is done using MATLAB. The main purpose of this thesis is to build a reliable method for the recognition of handwritten digit strings. In order to accomplish the recognition task, first, the digit string is segmented into individual digits. Then, a digit recognition module is employed to classify each segmented digit completing the handwritten digit string recognition task. In this study, a novel method, which uses the K-Nearest Neighbors classifier, is proposed to achieve high performance on the digit string segmentation problem. Our main contribution in this work is that KNN was more performance than Neural Net to discover that the result of our study is different than other studies. K-Nearest Neighbor (k-NN) classifier performed slightly better than the neural network.

Methodology

1. **Description of the dataset:**

In this report, we used the MNIST database consisting of offline handwritten digits ranging from 0-9. The database was constructed from Special Database 3 (SD-3) and Special Database 1 (SD-1) that contain binary images of handwritten digits. SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. For the results to be independent of both datasets, MNIST dataset was built by mixing NIST SD-1 and SD-3. The total number of digit image samples (70,000), the total number for training (60,000) and testing (10,000), and the subtotal number for each digit are shown in table 1. Each digit is a Gray-level fixed-size image with a size of 28 x 28 (or 784 pixels) in total as the features.

Table 1. Dataset

Digits	# Training	# Testing	Subtotal
9	5949	1009	6958
8	5851	974	6825
7	6265	1028	7293
6	5918	958	6876
5	5421	892	6313
4	5842	982	6824
3	6131	1010	7141
2	5958	1032	6990
1	6742	1135	7877
0	5923	980	6903
Total	60,000	10,000	70,000

The dataset contained examples from approximately 250 writers. The sets of writers for the dataset were disjoint. The Sample digit database of MNIST is shown in figure 1.

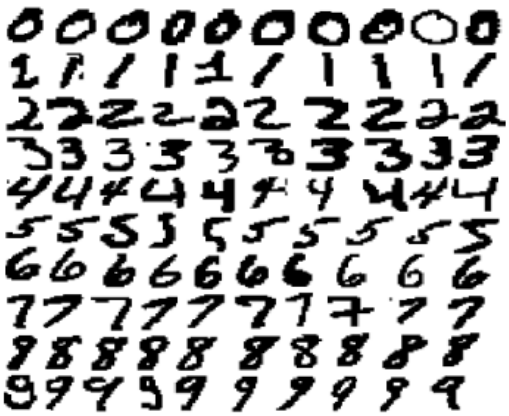

Figure 1. Sample images of MNIST digit database

2. Machine learning method:

2.1. K-Nearest-Neighbor method:

The K-Nearest-Neighbor is a type of Lazy Learners and one of the most commonly used nearest neighbor-based algorithms, works on learning by analogy, that is by comparing a given test example with training example that are similar to it. The training examples are described by n attributes. Each example represents a point in a n-dimensional space. In this way, all the training examples are stored in a n-dimensional pattern space. When given an unknown example, a k-nearest-neighbor classifier searches the pattern space for the k training example that are closest to the unknown example. These k training examples are the k "nearest neighbors" of the unknown example. The closeness that is the Euclidean distance between two examples, X1 = (x11, x12…, x1n) and X2 = (x21, x22…, x2n) is given by

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{21})^2}$$

Before using the equation above, each attribute is normalized as it helps to prevent attributes with initially large ranges from outweighed attributes with initially small ranges. This can be done by Min – max normalization, for transforming a value v of a numeric attribute A to v' in the range [0,1] by computing:

$$v' = \frac{v - min_A}{max_A - min_A}$$

where min_a and max_a are the minimum and maximum values of attribute A.

2.2. Artificial neural network (ANN):

ANNs are non-linear mapping structure. ANNs can recognize correlated patterns between input data set and corresponding target values. ANNs has huge capacity in prediction, pattern recognition, data compression, decision-making, etc. ANNs are recently used in the classification problem where regression model and other statistical techniques have traditionally been applied. Today, there are many different models of ANNs. The differences might be the topology, the functions, the hybrid models, the accepted values, the learning algorithms, etc.

3. Preprocessing Recognition System:

3.1. Normalization:

The first step in data preprocessing is data normalization. This is done to apply distance calculations on it. This involves transforming the data to fall within a smaller or common range, such as [0, 1]. The raw image data is based on the standard 8-bit unsigned integer which has a high value range of [0, 255] at each pixel (attribute). Expressing an attribute in smaller units will lead to a larger range for that attribute, thus tend to give such attributes greater effect or "weight." Normalizing the data attempts to give all attributes an equal weight and the input values for each attribute measured in the training tuples. It will help speed up the learning phase and it makes possible to add a new attribute to the dataset in the later stage if the new attribute is also normalized. Figure 2 shows the normalization effect on the handwritten image. The Raw image is in the range of [0, 255]. After normalizing it is reduced to the range of [0, 1].

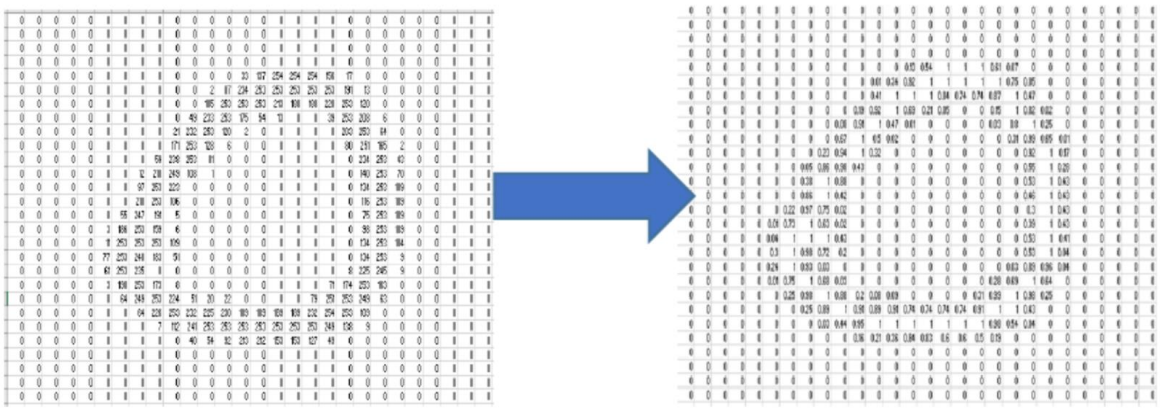

Figure 2. Raw Image (Left) and normalized image (Right)

3.2. Noise Reduction:

After Normalization, we used the median filter to remove noise this is a nonlinear digital filtering technique to improve the image by removing especially Gaussian noise. We used Median Filter because it preserves the edge while removing the noise as edge is an important aspect of an image. Figure 3 shows the processed image of the median filter. An unwanted horizontal line connected to number “Zero” is removed after Median Filter.

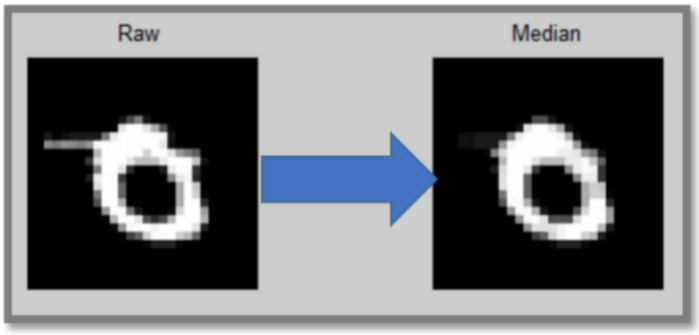

Figure 3. Raw Image (Left) and image after Median Filter (Right)

3.3. Image Sharpening:

The next step in preprocessing is the image sharpening technique which uses a blurred, or "unsharp", negative image to create a mask of the original image. The unshaped mask is then combined with the positive (original) image, creating an image that is sharper than the original. Sharpening uses a filter that amplifies the high-frequency components of a signal. It’s a necessary step taken after Median Filter as Median Filter not only remove noise, but also weaken the entire image in general. Sharpening can restore or enhance some of the useful information weakened by Median Filter. Figure 4 shows the image sharpening after processing Median Filter.

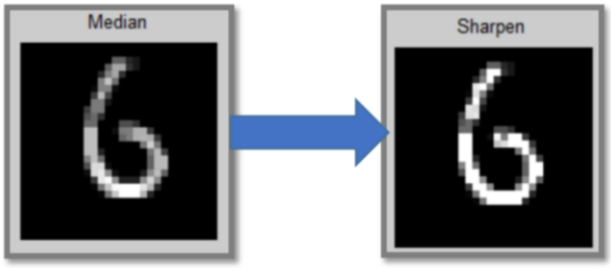

Figure 4. Median Filter and sharpened image

3.4. Image Attribute Reduction

Attribute reduction techniques is done to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced data set should be more efficient yet produce the same (or almost the same) analytical results. Each original image has a total of 784 attributes. It would be beneficial to reduce the total attributes to a relatively small amount so that it's more data efficient and easier to be processed. A direct way to reduce the attribute is by dividing the image into each block and finding the mean of that block as one attribute. Each block can be treated as one attribute. The number of blocks determines the number of attributes. This is done for 2 x 2, 4 x 4, 7 x 7, and 14 x14 attributes (blocks) of the image. Accuracy is improved as the no of blocked are increased (proved later in this report).

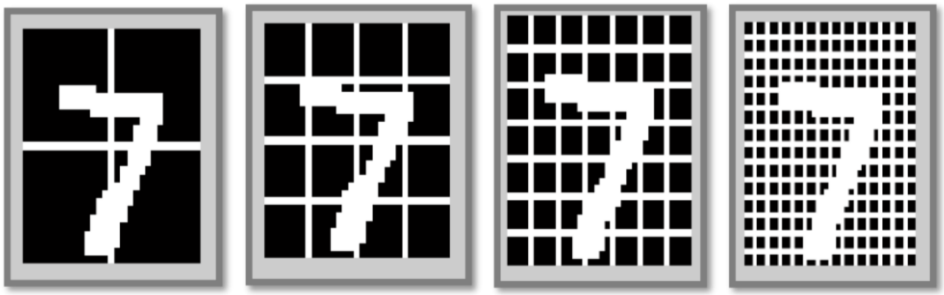

Figure 5. Image Attribute Reduction 2 x 2, 4 x 4, 7 x 7, and 14 x14

Figure 5 shows a visual representation on how the Image attribute reduction is done on the handwritten image. First the image is divided into equal blocks in a 2 x 2 block. Image attribute is reduced to 4 by taking mean for each block. A dramatic attribute reduction is achieved. Then, the same process is repeated for all 10,000 images. Apart from that, a 4x4, 7x7 and 14x14 attribute reduction is performed separately to compare and find the optimal number of attributes that best represent the image.

3.5. Add A New Attribute:

We used BWEULER function in the MATLAB to extract a feature of the handwritten image, it returns the Euler number for the binary image. The Euler number is the total number of objects in the image minus the total number of holes in those objects. Objects are connected sets of on pixels, that is, pixels having a value of 1. Figure 6 is an example that BWEULER evaluates an image and provides a Euler number. No. of objects is 1 and the No. of hole is 1 so Euler No is 0.

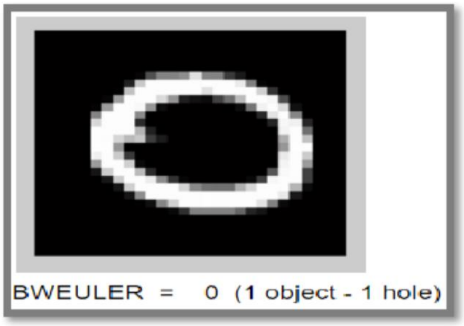

Figure 6. Sample Image and BWEULER Output for One Image

Figure 7 represents the data is run on 10,000 datasets. The portion of image "1" is clearly recognized as "1" than other portions of Number. From this, this new attribute could be added to the whole dataset and used to separate image "1" from other images

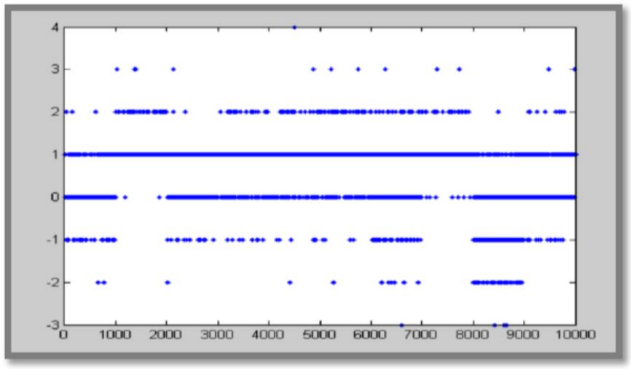

Figure 7. BWEULER output for Entire 10,000 Images

Test Set

The program to be developed needs to be tested against some images that contain handwritten digits so as to be able to assess its performance and calculate its success rate. That is why it is very necessary to create a test set. The test set represents an example of the images containing the handwritten digits which will have to be compared to the images in the reference set to identify them. This set was formed using the file from the MNIST database. The original file contained 60,000 images representing different digits. This made it difficult to look for each number using the label for the testing of the program. In order to make it easier to access each digit we want; we have decided to store several images from each digit in a separate file. That is why we have stored 20 images of each digit in ten different files. That is to say, the resulting test set was in the form of ten files, each one of them represents a digit and contains 20 images of it. These images were extracted from the initial file by reading them and their labels using Octave. In order to make the manipulation of the matrices/images easier, we had to make some modifications in the elements of all the matrices representing the test set as well. The black pixels were originally represented as zeros, so they were left the same. As for the white ones, each of them had a different nonzero number, so we turned them all into ones

Results

In this section, the results of recognition system for offline handwritten Gurmukhi characters are presented. The K-Nearest-Neighbor has used in this study. The results are based on two feature extraction techniques, namely, diagonal and transitions features. As sated earlier, we have also experimented some partitioning strategies. The training and testing the data are done in Rapid Miner with 70% of the data in training and 30% for testing. The operator is first trained using the Training dataset and later the images in the testing dataset are send to the apply model operator to test the accuracy of the classifier. Figure 8 shows the main process layout of the Rapid Miner. All the dataset (training and testing) is combined and split into 70:30 ratio gets independent results before sending it to be training and testing. Various classification models were used to check the accuracy of the model like Neural Net, K-NN, Random Forest, Decision Tree and Bagging with gradient boost.

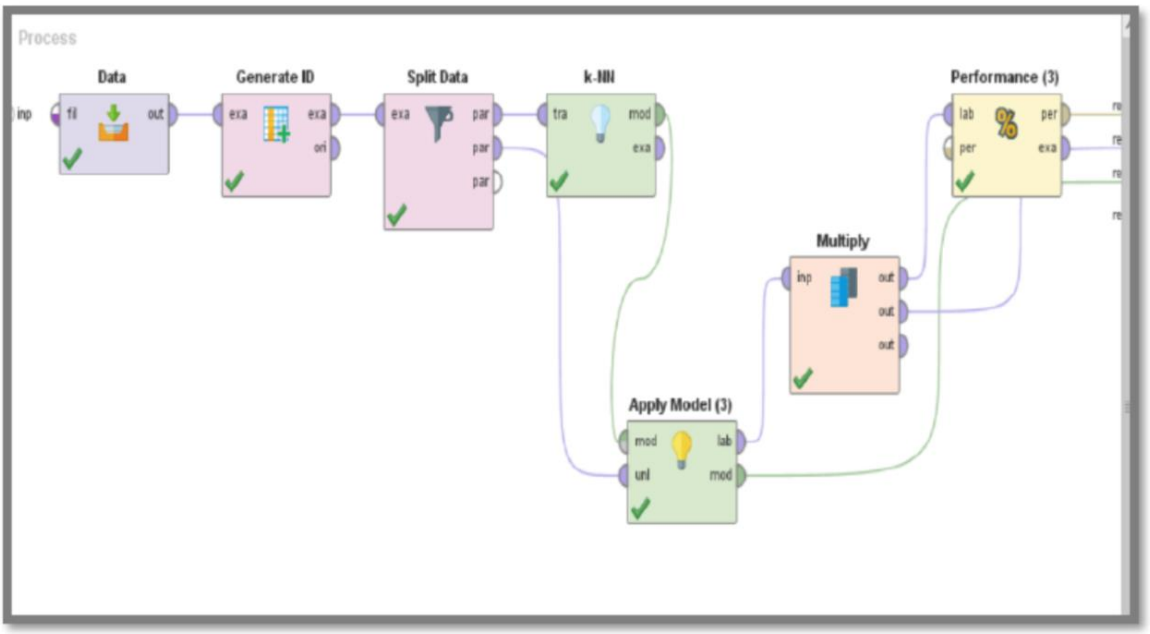

Figure 8. Main Process

1. **Accuracy by Methods:**

To answer the first hypothesis, the same dataset is used with 70:30 train/test ratio and classified by various classification models.

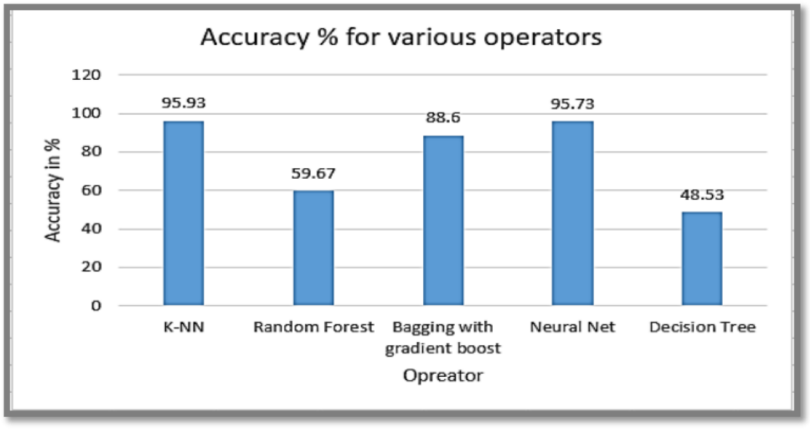

Figure 9. Accuracy in % for various operators

Figure 9 clearly shows that maximum accuracy is obtained by K-NN and Neural Net classifiers. Since, K-NN is significantly faster than Neural Net and its performance is similar as Neural Net, so we decided to use K-NN for prediction.

2. **Accuracy by Attributes:**

Using the dataset obtained by Image Attribute Reduction in MATLAB (discussed earlier in the preprocessing section) analysis is done to check the accuracy of the classifier K-NN and Neural Net. The results in figure 10 were obtained with 70:30 train/test ratio by varying attributes length to 4, 16, 49 and 196.

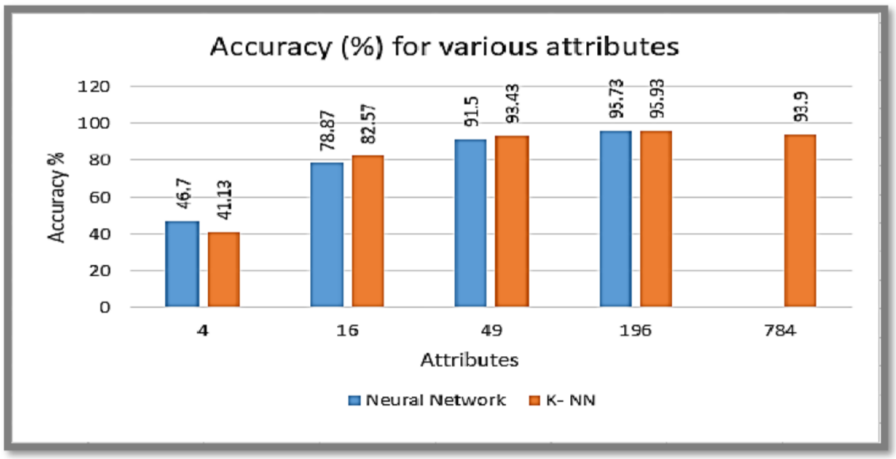

Figure 10. Accuracy in % for various Attributes

The graph clearly shows that when the attributes are increased the accuracy is also improved. With 196 attributes, we got an accuracy of 95.73% and 95.93% in Neural Net and K-NN respectively. It also shows that 784 attributes were reducing the accuracy compared to 196 attributes. It implies that more attributes preserve more information about the image and that helps classification, but excessive information carried by 784 attributes for instance did not improve the classification. Thus, we decided to use 196 attributes for both training and testing as it was the most efficient and enough representation of the image

3. Accuracy by Training Ratio

To check the impact of the training data on the accuracy, we decided to change the ratio of the training and testing data using the reduced 196 attributes.

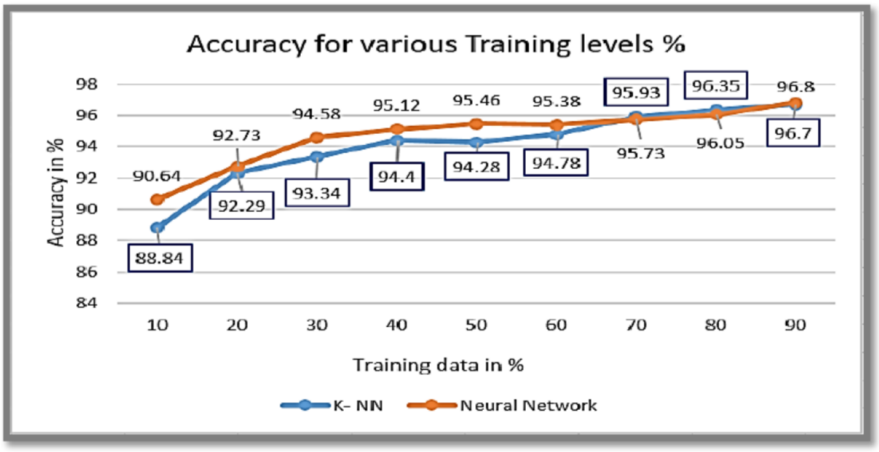

Figure 11. Accuracy for Various Training Levels in %

On performing the analysis in RapidMiner using 196 attributes for various training and testing ratios with classifier Neural Net and K-NN. Figure 11 clearly indicates that as the training data is increases the accuracy of the classifier is also improved. With 90:10 ratio of training and testing, a maximum accuracy of 96.8% and 96.7% were achieved by Neural Net and K-NN respectively. More data for training helps classification significantly.

Conclusion

This study settled on classifying a given handwritten digit image as the required digit using five different algorithms and consequently testing its accuracy. This study built handwritten recognizers evaluated their performances on MNIST (Mixed National Institute of Standards and Technology) dataset and then improved the training speed and the recognition performance. In addition to develop a system for word based handwriting recognition system and test the handwriting of a given word and detect the writer by selecting which is being recognized for most of the user for a given training sample. This study discusses in detail all advances in the area of handwritten character recognition. The most accurate solution provided in this area directly or indirectly depends upon the quality as well as the nature of the material to be read. Various techniques have been described in this paper for character recognition in handwriting recognition system. The result of this study shows that accuracy is improved as the no of blocked are increased. Apart from that, a 4x4, 7x7 and 14x14 attribute reduction is performed separately to compare and find the optimal number of attributes that best represent the image. The initial hypothesis was answered with concrete data support. Comparing all the classification models we tested K-Nearest Neighbor is the preferred choice in terms of its high accuracy and computational efficiency. However, there is no single classifier that works best on all given problems. A result shows that probabilistic methods suit better for handwriting recognition. By varying the training and testing ratios (from 10% to 90%) we found that the larger training data size improves accuracy, but smaller testing dataset may also favor better accuracy. Preprocessing such as Attribute reduction (784 reduced to 196)

reduce runtime and increase accuracy (from 93.2% to 95.9%). The proposed algorithm tries to address both the factors and well in terms of accuracy and time complexity. The general accuracy of tested K-NN was found to be 96.7% while 96.8% were achieved by Neural Network respectively. The overall highest accuracy 96.8% is achieved in the recognition process by Neural Network with the sacrifice of significantly extended runtime. This work is carried out as an initial attempt, and the aim of the paper is to facilitate for recognition of handwritten numeral without using any standard classification techniques. Image processing techniques Median filter, binary, Bweuler, and sharpening improve image quality, but we should be cautious to implement them as it may cause new problems in some cases. Handwritten digit recognition system can be extended to a recognition system that can also able to recognize handwritten character and handwritten symbols. Future studies might consider hardware implementation of the recognition system. In the future, we are planning to further explore the topic of data classification in perhaps other industrial related applications.