HAND-WRITTEN DIGIT RECOGNITION SYSTEM

This project report is submitted to

Silicon Institute of Technology, Bhubaneswar

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

Uttam Sundar Ray (180310060)

Under the Esteemed Supervision of

Dr. Pradyumna Kumar Tripathy

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SILICON INSTITUTE OF TECHNOLOGY
SILICON HILLS, BHUBANESWAR – 751024, ODISHA, INDIA

May,2021

ACKNOWLEDGEMENTS

We have taken a lot of effort into this project. However, completing this project would not have been possible without the support and guidance of a lot of individuals. We would like to extend our sincere thanks to all of them.We are highly indebted to our esteemed faculty Dr. Pradyumna Kumar Tripathy for his guidance and supervision. We would like to thank him for providing the necessary information and resources for this project. We are grateful to Mr. Jiten Mohanty for his consistent efforts and guidance throughout the development of this project. We would like to express our gratitude towards our parents & our friends for their kind co-operation and encouragement which help us a lot in completing this project.

Our thanks and appreciations also go to our colleague in developing the project. Thank you to all the people who have willingly helped us out with their abilities.

Uttam Sundar Ray

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SILICON INSTITUTE OF TECHNOLOGY
BHUBANESWAR – 751024

ABSTRACT

Handwritten Digit Recognition is a classification problem, where the model tries to classify the handwritten digit to any of the 10 numbers (Starting from 0 to 9). This is a supervised learning problem, and there is a widely popular dataset — MNIST Dataset, that comprises of 70,000 images of handwritten numbers and its labels. While most of the techniques are based on strong foundation, there are subtle or wide differences in their efficiencies. By efficiency, the accuracy, time required to get the result and other factor which can give one algorithm or techniques an edge over the rest. In this project, we aim to build a Machine Learning model with a GUI that lets the user hand-draw a number on the screen, and the model predicts the digit. The infamous Machine Learning algorithms K-Nearest Neighbors and Support Vector Machine were used as classifiers. The GUI canvas for drawing the digits for training purposes is developed using Tkinter library of python. As the end result, we aim to deliver a working model for Handwritten Digit Classification problem with minimum error in accuracy.

LIST OF FIGURES

CONTENTS

CHAPTER 1

INTRODUCTION

The problem of handwriting recognition is to interpret intelligible handwritten input automatically, which is of great interest in the pattern recognition research community because of its applicability to many fields towards more convenient input devices and more efficient data organization and processing. As one of the fundament problems in designing practical recognition systems, the recognition of handwritten digits is an active research field. Immediate applications of the digit recognition techniques include postal mail sorting; automatically address reading and mail routing, bank check processing, etc.

1.1. BACKGROUND

As a benchmark for testing classification algorithms, the MNIST dataset has been widely used to design novel handwritten digit recognition systems. There are a great amount of studies based on MNIST dataset reported in the literature, suggesting many different methods. A comprehensive review by LeCun et al. compares the performance of previously proposed classifiers including linear and polynomial classifiers, Nearest Neighbor classifiers, and different neural networks. A best test error rate of about 0.7% is achieved by combining multiple neural network classifiers. One of the major challenges in the recognition of handwritten digits is the within class variance, because people do not always write the same digit in exactly the same way. Many feature extraction approaches have been proposed trying to characterize the shape invariance within a class to improve the discrimination ability. Experiments have shown that by extracting direction features, local structure features or curvature features, the accuracy and efficiency of many classifiers could be improved significantly

1.2. PROBLEM STATEMENT

In this project the problem of recognizing hand written digits and optimizing a model with better speed and accuracy. The statement of the problem is explicitly declared as:
“To develop a model to recognize handwritten digits with optimum speed and accuracy”.

1.3. OBJECTIVE AND MOTIVATION

Hand writing recognition of characters has been around since the 1980s.The task of handwritten digit recognition, using a classifier, has great importance and use such as – online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example - tax forms) and so on. There are different challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins. Our goal was to implement a pattern classification method to recognize the handwritten digits provided in the MINIST data set of images of hand written digits (0-9).Each image is a 28 x 28 grayscale (0-255) labeled representation of an individual digit.

1.4. PROPOSED METHOD

K-Nearest Neighbors Classifier

K-Nearest Neighbors algorithm is preferably used in this classification problem because of the lower model complexity of this algorithm. The lower model complexity of this algorithm enables better performance in terms of speed without compromising the accuracy of the model. We have chosen K=6 based on the below experimental test results of modeling for K=1 to 9.

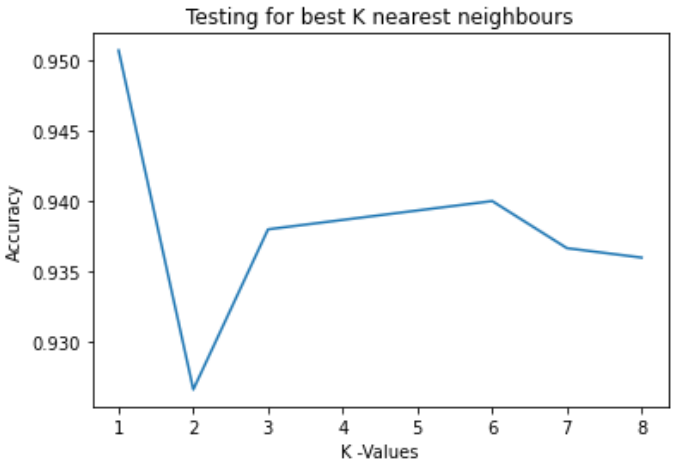


Figure 1.1: Plot for testing values of K

As it is visible from the figure, the optimum accuracy is achieved with K=6.

SUMMARY

In this chapter we discussed about:

- The idea and background of this project and its domain
- The exact problem statement, objective of the project
- The proposed method towards the problem statement

CHAPTER 2

LITERATURE REVIEW

2.1. INTRODUCTION

The procedure of transformation of images into understandable format which are hand-written, typewritten, or printed digits, for the purpose of editing, indexing/searching, and a storage size reduction, is commonly known as handwritten recognition. Handwritten recognition system has usefulness and importance Fig. 1. Handwritten Digit Recognition in numerous fields such as processing bank check amount, recognizing the zip codes on mails for postal mail sorting, online data indexing, handwriting recognition on computer, numeric entries in the form filled by hand and so on. The handwritten recognition system consists of two distinct domains on the basis of input signals, online and offline. The static representation of a digitized document is used in the offline system of digit recognition, example of which are check form, mail or document processing. Contrary to offline, in online system depends on the information acquired during the production of the handwriting. In the online handwritten recognition, the writing tool trajectory knowledge capturing instrument is required. New era electronics devices, such as smart phone, electronic pad and digital personal assistant have online handwritten recognition system in them. Hence it is important to improve and optimize the performance of the recognition system, aiming for the reducing the storage space and improving the processing speed. Figure 1 shows an online handwritten digit recognition system.

Figure 2.1: Flow of Control

We have chosen the K-Nearest Neighbors Algorithm as our primary classifier for this problem as it is a less complex algorithm hence we intend to build a fast model.

2.2. K-NEAREST NEIGHBORS CLASSIFIER

This classifier is one of the simplest classifier to implement because it is a brute force method. The algorithm is used to find the nearest match for a given case to the known cases stored in memory. The K is used to signify how many votes are used for decision making. It is most optimal to choose a value of K that is odd so it eliminates a tie between two sets. There are three steps to implement for this classifier: First step: compare known samples with the test sample using a distance metric. The distance metric is used to find the nearest neighbor. The most widely used distance metric is the Euclidean distance, because it gives a normalized value. The Euclidean distance is:

$$\vec{D} = \sum^{i} (known_i - testcase_i)^2$$

Second Step: Once we have the distances of the test subject to known subjects, we can rank them accordingly. If we are given 1000 known samples, we can rank distances of the results from 1 to 1000. The value K denotes the number of ranks to use. For example if K is equal to 99 then the top 99 distance vector value is considered.

Third Step: Considering a case where it is either true or false, within the 99 results the one with the greatest number is the value of the test case. Then if there were 50 results that point to true and 49 results point to false, then the test sample is true. This is why an odd value is chosen so there would be no ties. It is important to note that there could be more than 2 cases, so a tie could result even with odd value of K, for example if there were three sample spaces. A triangle, a circle and a square object, there could be 40 votes for circle and square out of 99 which would result in a tie. For this case it would be best to consider their ranking number as well.

2.3. ALTERNATIVE METHODS

From the literature we reviewed we found that the highest accuracies achieved by classifiers on the MNIST database were 99.05%, by individual classifiers and multiple classifier systems, and 99.3% (boosted), by multiple neural networks. Below listed are the main schemes that we reviewed, and their advantages and disadvantages, and the caveats around their implementation.

Neural Networks

Multilayer Neural Networks trained with the back-propagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Convolutional Neural Networks are a special kind of multi-layer neural networks used to recognize visual patterns with extreme variability (such as handwritten digits), and with robustness to distortions and simple geometric transformations.

Given, the high accuracy achieved by this technique, this was our initial proposed algorithm for this project. However, the implementation of the algorithm is highly complex. Convolutional neural networks are particularly well suited to hardware implementations because of their regular structure and low memory requirements for the weights. They also perform more efficiently and are well suited to the extensively large database of digits. However, from the reviews it has been found that the training time for these networks are expectedly longer, amounting to 2 to 3 days of CPU time in some cases for 10 to 20 passes through the training set. Given the limited time scope of the project, and major emphasis being on being able to review the literature and implement a basic algorithm, we decided to proceed with the K-Nearest Neighbor Algorithm.

SUMMARY

In this chapter we discussed about:

- The basic procedure of classifying hand written digits.
- The potential scope of the stated problem statement.
- The step-by-step flow of K-Nearest Neighbors classification algorithm.
- The alternative models / techniques used by various researchers to address the problem statement.

CHAPTER 3

METHODOLOGY

3.1. GUI FOR REAL TIME TESTING:

This project includes a User Interface, where the user can draw any number on the canvas, and the ML model will try to guess what that digit is.

Pre Processing the Image: Building the canvas and positioning the buttons were really easy, but the drawing on the Canvas had to be converted to an image that can be fed to the Machine Learning model.

1. Drawing on the Canvas was first to be retrieved.
   Co-ordinates of the Canvas were calculated and passed to the ImageGrab method as parameters, to accomplish this task.
2. The grabbed Image had to be inverted.
   The ML model is trained on images with Black background and white font. Whereas the grabbed image had white background and black font.
3. The Image was resized to 28*28
   The ML model required the input image to be in 28*28 dimensions, as it was trained only with 28*28 images.
4. Image had to be converted to Greyscale
   Converted the image from RGB to Greyscale.
5. Store the image as Numpy data for further processing.

The GUI window was developed in the following manner:

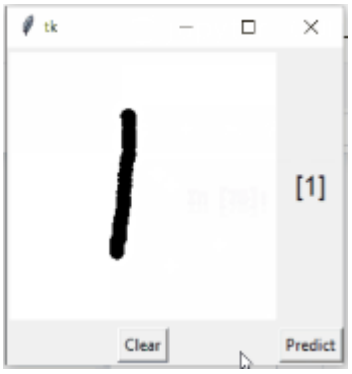


Figure 3.1: Snapshot of drawing canvas made using Tkinter GUI

3.2. DATA DESCRIPTION:

The MNIST database contains 60,000 digits ranging from 0 to 9 for training the digit recognition system, and another 10,000 digits as test data. Each digit is normalized and centered in a gray-level image with size 28 × 28, or with 784 pixel in total as the features. Some examples are shown in Figure 3.

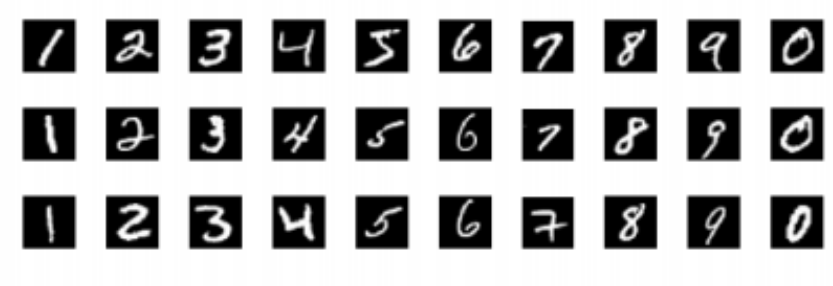


Figure 3.2: Example of Samples from MNIST Dataset

3.3. THE K-NEAREST NEIGHBOURS ALGORITHM:

A sample of the original dataset was used for initial training and testing purposes. The sample of the dataset consisted a total of 6000 images out of the 60000 images used for final training set. Out of the 6000 samples, 4800 were used for training purpose and 1200 were kept for testing. We have chosen K=6 based on the below experimental test results of modeling for K=1 to 9.

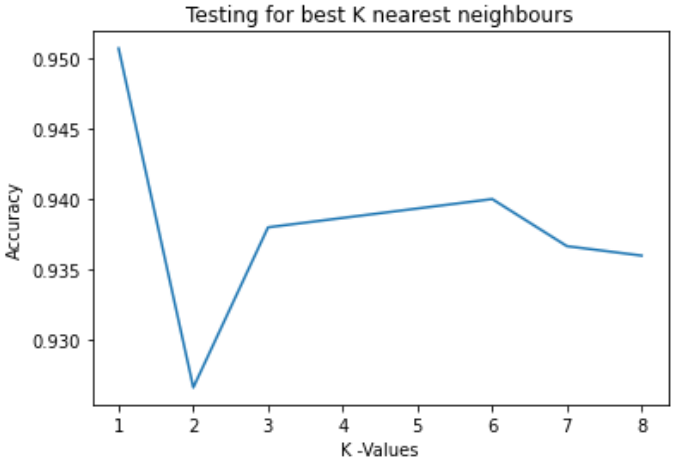


Figure 3.3: Test for best K

The nearest neighbour model with K=6 gives us the accuracy of 93.7% approximately. The accuracy was expected to improve with subsequent increase in training data.

3.4. FINAL TRAINING & SAVING THE MODEL:

The KNN model was then trained with K=6, with 60000 images as training data with a training accuracy of 94%. Upon testing with the left over 10000 samples, an accuracy of 96% was recorded.After building the K-Nearest Neighbour model, we used the ‘pickle’ library to save the ML model to a local file.

The K-Nearest Neighbours model trained on 60K 28*28 images resulted in about 430MB file. The file size was really huge, and it would become very hard if we had to use this on an Application. While searching for an alternate approach, that serializes ML models as a smaller file, ‘joblib’, which allows compression.

SUMMARY

In this chapter we discussed about:

- The processing of images drawn on the canvas.
- The images kept in the MNIST dataset.
- The step-by-step procedure followed to decisively finalize the final model

CHAPTER 4

RESULTS & FUTURE SCOPE

4.1. RESULTS

The above stated K-Nearest Neighbours model was built with accuracy up to 97% after subsequent trials of testing and randomizing the selection of training and testing data. This model was then integrated with python's Tkinter based GUI for real world testing. The results of testing are shown below.

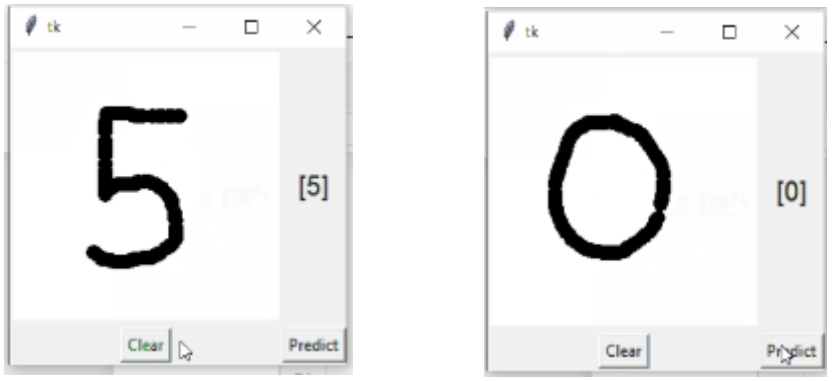


Figure 4.1: Final test results of model integrated with GUI

4.2. FUTURE SCOPE

To further improve our pattern recognition system we also consider combination of the classifiers. Common techniques for combination of classifiers are considered, such as bagging, boosting and other resampling methods. However a significant improvement can be achieved only if the component classifiers perform better than chance, whereas the six classifiers we focus on commonly have low error rate. Some other issues like higher cost again arise. Next, we consider weighted sum of several pairs of classifiers, using corresponding training error as the weight. The resulting test error rate, however, always falls between the error rates of the individual classifiers. We conclude that the methods for combining classifiers we implement so far do not significantly improve the recognition system. However it remains a potential way for achieving better results to combine the classifiers.

REFERENCES

[1] MNIST Database of Handwritten digits: http://yann.lecun.com/exdb/mnist/

[2] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner "Gradient-Based Learning Applied to Document Recognition", IEEE, November 1998

[3] Gaurav Jain, Jason Co, Handwritten Digits Recognition ECE462 - Multimedia Systems, University of Toronto, http://individual.utoronto.ca/gauravjain/ECE462-HandwritingRecognition.pdf

[4] Rahul Bhalley, Introduction to digit Classification, www.towardsdatascience.com

[5] Akkireddy Challa, Automatic Handwritten Digit Recognition On Document Images Using Machine Learning Methods, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

[6] Ming Wu ,Zhen Zhang , Handwritten Digit Classification using the MNIST Data Set, Michigan State University

APPENDIX – A

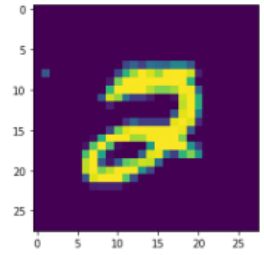
SOURCE CODE

```
def load_mnist(filename, type, n_datapoints):
    # MNIST Images have 28*28 pixels dimension
    image_size = 28
    f = gzip.open(filename)

    if(type == 'image'):
        f.read(16)    # Skip Non-Image information
        buf = f.read(n_datapoints * image_size * image_size)
        data = np.frombuffer(buf, dtype=np.uint8).astype(np.float32)
        data = data.reshape(n_datapoints, image_size, image_size, 1)
    elif(type == 'label'):
        f.read(8) # Skip Inessential information
        buf = f.read(n_datapoints)
        data = np.frombuffer(buf, dtype=np.uint8).astype(np.int64)
        data = data.reshape(n_datapoints, 1)
    return data
```

```
index = random.randint(0, train_size)
print('Index: ', index)
print('Training Set: ')
print('Label:', y[index])
img = np.asarray(X[index]).squeeze()
plt.imshow(img)
plt.show()
```

```
Index:  57319
Training Set:
Label: [2]
```




```
train_size = 60000
test_size = 10000
dirpath = '/content/drive/MyDrive/Projects/Digit Recognition/'
#/content/drive/MyDrive/Projects/Digit Recognition/train-images-idx3-ubyte.gz
X = load_mnist(dirpath + 'train-images-idx3-ubyte.gz', 'image', train_size)
y = load_mnist(dirpath + 'train-labels-idx1-ubyte.gz', 'label', train_size)
X_test = load_mnist(dirpath + 't10k-images-idx3-ubyte.gz', 'image', test_size)
y_test = load_mnist(dirpath + 't10k-labels-idx1-ubyte.gz', 'label', test_size)
```

```
from sklearn.model_selection import train_test_split
X_train, X_valid, y_train, y_valid = train_test_split(X[:(train_size//10)], y[:(train_size//10)], test_size=0.25, random_state=28)
print(X_train.shape, X_valid.shape, y_train.shape, y_valid.shape)
```

```
(4500, 28, 28, 1) (1500, 28, 28, 1) (4500, 1) (1500, 1)
```

KNN Classifier , k value testing

```
score = []

for k in range(1, 9):
  print('Begin KNN with k=',k)
  classifier = KNeighborsClassifier(n_neighbors=k)
  classifier = classifier.fit(X_train.reshape(X_train.shape[0], 28*28), y_train)
  pred = classifier.predict(X_valid.reshape(X_valid.shape[0], 28*28))
  accuracy = accuracy_score(y_valid, pred)
  score.append(accuracy)
  print("Accuracy: ",accuracy)

plt.plot(range(1,9),score)
plt.xlabel('K -Values')
plt.ylabel('Accuracy')
plt.title('Testing for best K nearest neighbours')
```

APPENDIX – A(Continued)

```
%%time
print('Training the Model')
classifier = KNeighborsClassifier(n_neighbors=6)
classifier = classifier.fit(X.reshape(X.shape[0], 28*28), y)
```

```
Training the Model
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d
  This is separate from the ipykernel package so we can avoid doing imports until
CPU times: user 16 s, sys: 105 ms, total: 16.1 s
Wall time: 16.1 s
```

```
y_pred = classifier.predict(X_test.reshape(X_test.shape[0], 28*28))
print(accuracy_score(y_test, y_pred))
```

```
0.9677
```

```
import joblib

joblib.dump(classifier, 'knn_model.gzip', compress=('gzip',3))
```

```
['knn_model.gzip']
```

Department of Computer Science and Engineering
Silicon Institute of Technology,
Silicon Hills, Bhubaneswar –751024,
Odisha, India