**HANDWRITTEN DIGIT RECOGNITION USING DEEP LEARNING**

**Enroll No. : 18102004, 18102254, 18102153**

**Name : Akshit Joshi , Himanshu Kumar , Ashutosh Sharma**

**Supervisor : Dr. Richa Gupta**



**DECEMBER-2020**

**Submitted in partial fulfillment of the Degree of**

**Bachelor of Technology**

**In**

**Electronics & Communication Engineering**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING**

**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

**DECLARATION**

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: …………… Signature:………………..

Date…………… Name .. .. .……………..

Enrollment…………………

**CERTIFICATE**

This is to certify that the work titled **Handwritten Digit Recognition Using Deep Learning** submitted by **Akshit Joshi, Ashutosh Sharma and Himanshu Kumar** in partial fulfillment for the award of degree of Bachelor of Technology of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Mentor: ……………

Name of Mentor: …....………

Designation: ..…………..

Date: ..…………..

**ACKNOWLEDGEMENT**

Words are not enough to express our gratitude, but we are taking this opportunity to present our deep sense of gratitude and respect towards all those who have helped us for the entire duration of the project. Working on a project involves continuous efforts in terms of consistency, quantity, energy, commitment, hard work, and also cooperation from people directly or indirectly related to the project domain. We appreciate the efforts of those who have made substantial contributions to our project.

Firstly, we want to thank our Project Mentor, Dr. Richa Gupta who was a staunch supporter of this project and a motivator. She was a constant source of support to us in our hard times. We would like to express our appreciation for the confidence she has shown in our capabilities and always encourage us to move forward with her vision.

We would also like to extend our thanks to Dr. Neetu Singh for sparing her valuable time to help us throughout the project whenever we required.

Last but not least, we thank all of our friends for providing critical feedback & support whenever necessary. There are times in such projects when the clock beats you time and you run out of energy, you just want to finish it once and for all, Parents and Friends made us with their unfailing humor & warm wishes to endure such times.

**SUMMARY**

To make machines more intelligent, the developers are diving into machine learning and deep learning techniques. A human learns to perform a task by practicing and repeating it again and again so that it memorizes how to perform the tasks. Then the neurons in his brain automatically trigger and they can quickly perform the task they have learned. Deep learning is also very similar to this. It uses different types of neural network architectures for different types of problems. For example – object recognition, image and sound classification, object detection, image segmentation, etc.

The handwritten digit recognition is the ability of computers to recognize human handwritten digits. It is a hard task for the machine because handwritten digits are not perfect and can be made with many different flavors. The handwritten digit recognition is the solution to this problem which uses the image of a digit and recognizes the digit present in the image.In this project, we are going to implement a handwritten digit recognition app using the MNIST dataset. We will be using a special type of deep neural network that is [Convolutional Neural Networks](https://data-flair.training/blogs/convolutional-neural-networks-tutorial/). In the end, we are going to build a GUI in which you can draw the digit and recognize it straight away.

Signature of Student: …………… Signature of Mentor: ……………

Name of Student: …………… Name of Mentor: ……………

Date: …………… Date: ……………

**LIST OF FIGURES**

Figure Page

* 1. Basic structure of deep learning model ……………………………… 08
  2. Loss Function …………………………………………………… 09

1.2 An example of a CNN …………………………………………… 10

1.3 Testing image table……………………………………………… 11

1.4 The Sample digit database of MNIST…………………………… 12

2.1 Raw Image (Left) and normalized image (Right)……………………… 13

2.2 Raw Image (Left) and image after Median Filter (Right)……………… 14

2.3 The image sharpening after processing Median Filter……………… 15

2.4 Image Attribute Reduction ……………………………….………… 16

2.5 Accuracy in % for various operators ………………………………… 17

2.6 Accuracy in % for various Attributes………………………………… 18

2.7 Accuracy for Various Training Levels in %………………………… 19

3.1 Plot of a Subset of Images From the MNIST Dataset……………… 21

3.2 Result Screenshots…………………………………………………… 22

3.3 Result Screenshots…………………………………………………… 23

3.4 Result Screenshots……………………………………………………… . 24

3.5 Result Screenshots………………………………………….……………. 25

3.6 Result Screenshots…………………………………………………… .. 26

3.7 Result Screenshots……………………………………………………….. 26

3.8 Difficulties……………………………………………………………….. 27

**TABLE OF CONTENTS**

CHAPTER 1 …………………. …………………………………………………………………………… . .. . . 08

INTRODUCTION

* 1. History of deep learning …………………………………………………………………………………… 08
  2. Research Methodology…………………………………………………………………………………… 09

1.2.1 Description of dataset……………………………………………………………………………………… 10

[1.2.2 Machine Learning Methods 11](#_Toc22351)

[1.2.3 Artificial neural network (ANN) 11](#_Toc7790)

CHAPTER 2

PREPROCESSING RECOGNITION SYSTEM

[2.1 Normalization . 12](#_Toc15279)

[2.2 Noise Reduction 12](#_Toc25149)

[2.3 Image Sharpening 13](#_Toc12561)

[2.4 Image Attribute Reduction 14](#_Toc1228)

[2.5 Accuracy Measures of the Model 14](#_Toc29160)

[2.5.1 Accuracy by Methods 14](#_Toc14038)

[2.5.2 Accuracy By Attributes 15](#_Toc13242)

[2.5.3 Accuracy By Training 16](#_Toc18618)

[CHAPTER 3 17](#_Toc23206)

[IMPLEMENTATION OF THE PROJECT 17](#_Toc5712)

[3.1 Description of the MNIST Handwritten Digit Recognition Problem 17](#_Toc8823)

[3.2 Loading the MNIST dataset in Keras 18](#_Toc17985)

[3.3 Simple Convolutional Neural Network for MNIST 18](#_Toc12564)

[3.4 Code Implementation 19](#_Toc8500)

[Tkinter Widgets 22](#_Toc25982)

[Results: 24](#_Toc10062)

[3.5 Difficulties 25](#_Toc4037)

[CONCLUSION 26](#_Toc32475)

**CHAPTER-1**

**INTRODUCTION**

**1.1 History of Deep Learning**

Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data.Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected.Deep learning utilizes both structured and unstructured data for training. Practical examples of Deep learning are Virtual assistants, vision for driver-less cars, money laundering, face recognition and many more.

Deep Learning Algorithms use something called a neural network to find associations between a set of inputs and outputs.

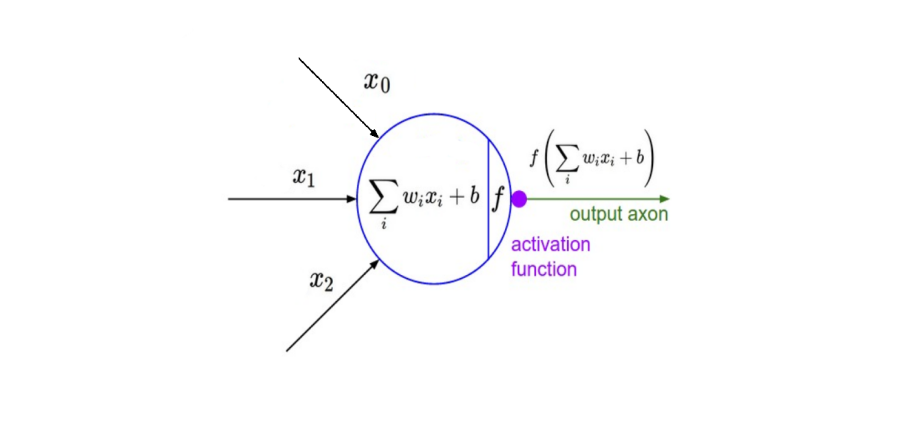


Figure 1.1 The basic structure is seen below above

A neural network is composed of input, hidden, and output layers — all of which are composed of “nodes”. Input layers take in a numerical representation of data (e.g. images with pixel specs), output layers output predictions, while hidden layers are correlated with most of the computation.The major points to keep note of here are the tunable weight and bias parameters — represented by w and b respectively in the function above. [2] These are essential to the actual “learning” process of a deep learning algorithm.After the neural network passes its inputs all the way to its outputs, the network evaluates how good its prediction was (relative to the expected output) through something called a loss function. As an example, the

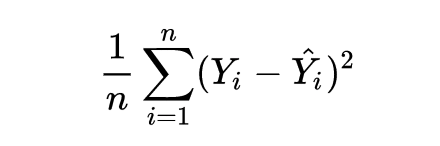


Figure 1.2 Mean Squared Error” loss function is shown.

Y hat represents the prediction, while Y represents the expected output. A mean is used if batches of inputs and outputs are used simultaneously (n represents sample count).[2]Deep learning is ultimately an expansive field, and is far more complex than described above. Various types of neural networks exist for different tasks (e.g. Convolutional NN for computer vision, Recurrent NN for NLP), and go far and beyond the basic neural network that have been covered above.

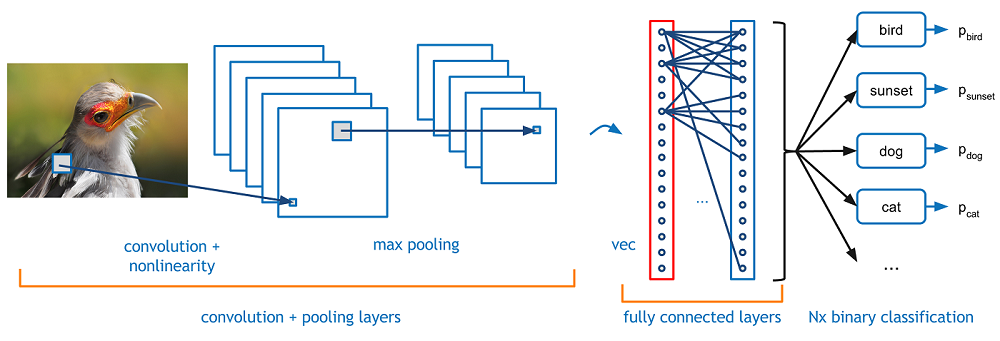


Figure 1.3 An example of a CNN is shown below in Fig.1.3

**1.2. Research Methodology**

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. [3] 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.

**1.2.1 Description of the dataset**

In this paper, we used the MNIST database consisting of offline handwritten digits ranging from 0-9. [4] 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 highschool 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 1.1. Each digit is a gray-level fixed-size image with a size of 28 x 28 (or 784 pixels) in total as the features.

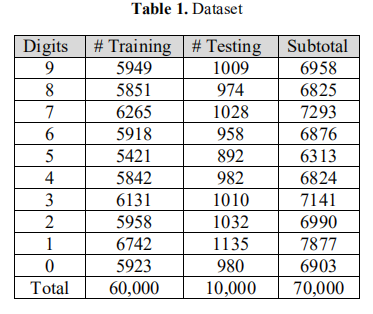


Table 1.1 Training and Testing Dataset of each digit

The dataset contained examples from approximately 250 writers. The sets of writers for the dataset were disjoint.

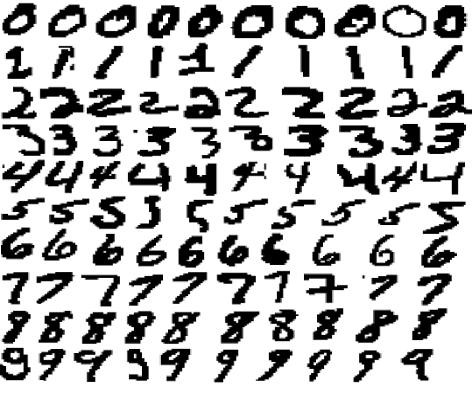
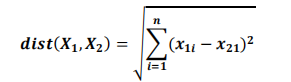


Figure 1.4 The Sample digit database of MNIST is shown in above

# 1.2.2 Machine Learning Methods

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. [5] 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



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



# 1.2.3 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 . [6] 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.

**CHAPTER 2**

**PREPROCESSING RECOGNITION SYSTEM**

The accuracy in recognition of handwritten numerals can be improved by preprocessing the raw data. With a brief investigation of the raw image data, the main issues discovered are image noise and unrecognizable handwriting. Thus, preprocessing of the raw data is deemed to be necessary before training them. A series of image processing techniques are conducted to lead through the preprocessing stage and they are discussed as follows.

**2.1 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]. [7] [8] 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 as long as 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].

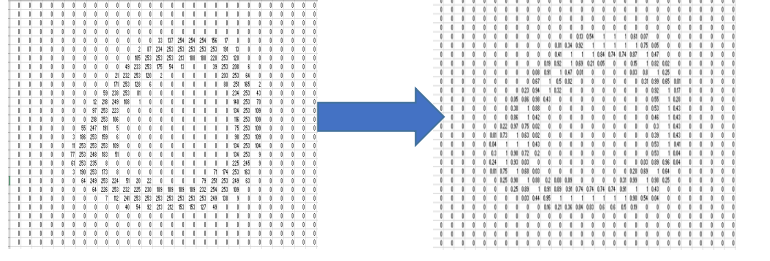


Figure 2.1 .Raw Image (Left) and normalized image (Right)

# 2.2 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.

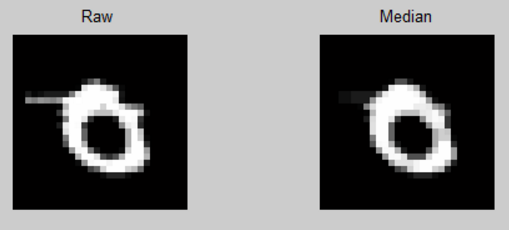


Figure 2.2Raw Image (Left) and image after Median Filter (Right)

# 2.3 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. [9] 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.

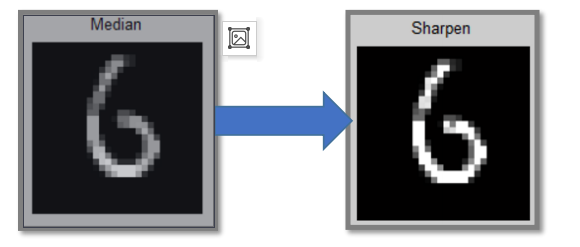


Figure 2.3 shows the image sharpening after processing Median Filter.

# 2.4 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. [10] [11] 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 .

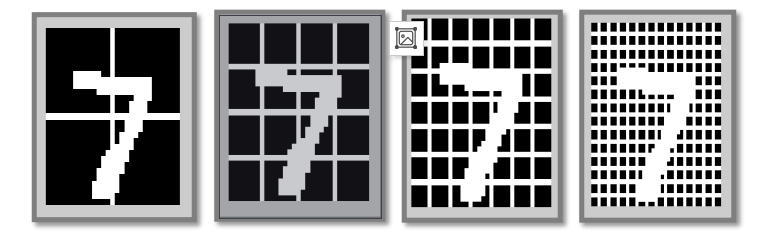


Fig. 2.4 Image Attribute Reduction 2 x 2, 4 x 4, 7 x 7, and 14 x14

Figure 2.4 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.

# 2.5 Accuracy Measures of the Model

The model that we make can be analysed on various grounds based on how we train our dataset.

2.5.1 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

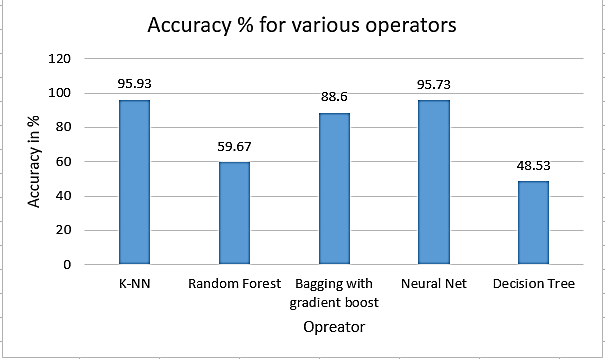
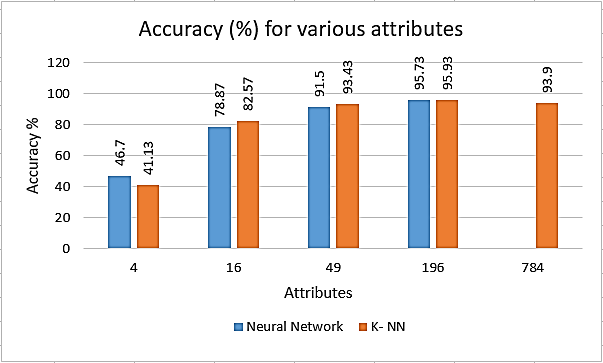


Fig 2.5 Accuracy in % for various operators

# 2.5.2 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.



.Fig 2.6 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.

# 2.5.3 Accuracy By Training

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.

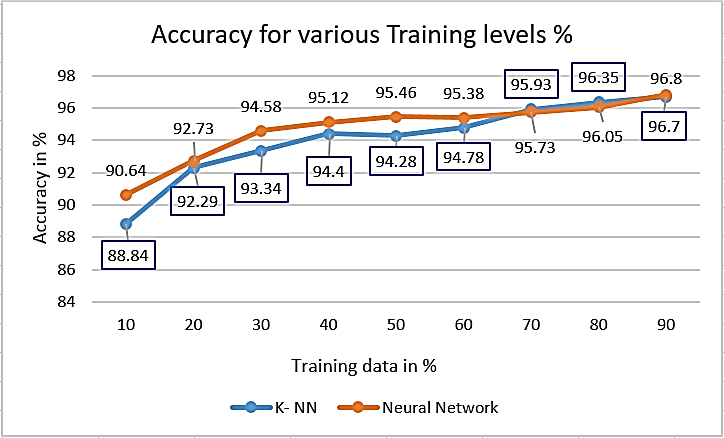


Figure **2.7.** Accuracy for Various Training Levels in %

On performing the analysis in Rapid Miner using 196 attributes for various training and testing ratios with classifier Neural Net and K-NN. Figure 2.7 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.

# CHAPTER 3

# IMPLEMENTATION OF THE PROJECT

# 3.1 Description of the MNIST Handwritten Digit Recognition Problem

The [MNIST problem](https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/) is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem.The dataset was constructed from a number of scanned document dataset available from the [National Institute of Standards and Technology](http://www.nist.gov/) (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset.Images of digits were taken from a variety of scanned documents, normalized in size and centered. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required.

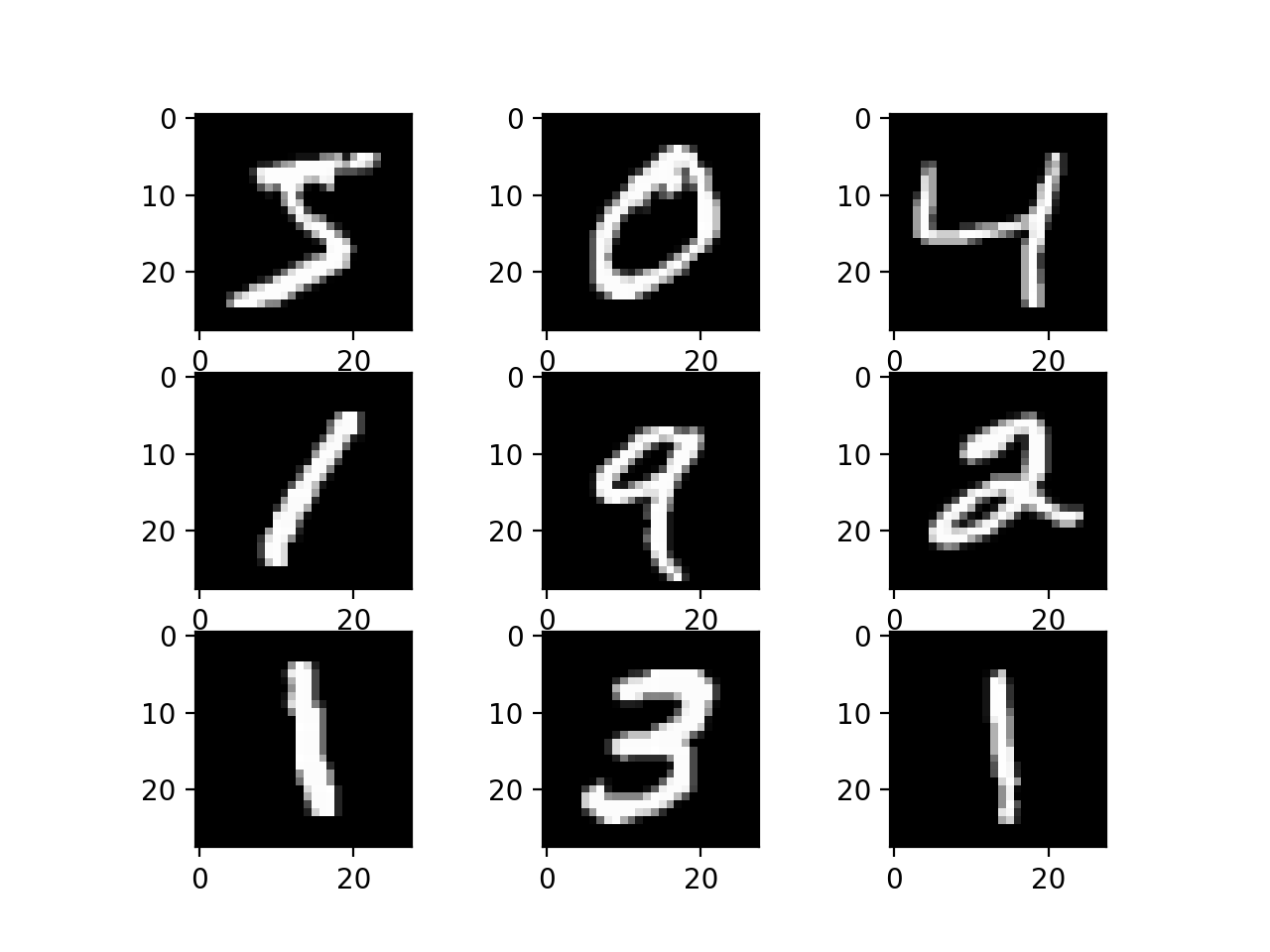


Fig 3.1 Plot of a Subset of Images From the MNIST Dataset

Each image is a 28 by 28 pixel square (784 pixels total). A standard split of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it.It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy.Excellent results achieve a prediction error of less than 1%. State-of-the-art prediction error of approximately 0.2% can be achieved with large Convolutional Neural Networks.

# 3.2 Loading the MNIST dataset in Keras

The Keras deep learning library provides a convenience method for loading the MNIST dataset.The dataset is downloaded automatically the first time this function is called and is stored in your home directory in ~/.keras/datasets/mnist.pkl.gz as a 15MB file.This is very handy for developing and testing deep learning models.

# 3.3 Simple Convolutional Neural Network for MNIST

Now that we have seen how to load the MNIST dataset and train a simple multi-layer perceptron model on it, it is time to develop a more sophisticated convolutional neural network or CNN model.Keras does provide a lot of capability for [creating convolutional neural networks](http://keras.io/layers/convolutional/).

The first step is to import the classes and functions needed.

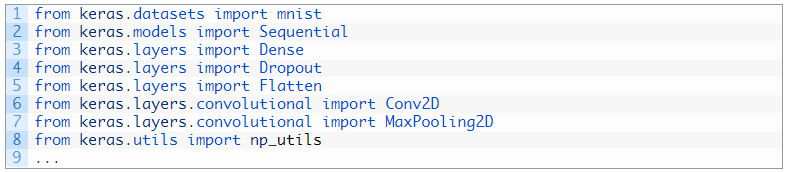


Fig. 3.2 Loading the libraries

Next we need to load the MNIST dataset and reshape it so that it is suitable for use training a CNN. In Keras, the layers used for two-dimensional convolutions expect pixel values with the dimensions[pixels][width][height][channels].we are forcing so-called [channels-last](https://machinelearningmastery.com/a-gentle-introduction-to-channels-first-and-channels-last-image-formats-for-deep-learning/) ordering for consistency in this example.In the case of RGB, the last dimension pixels would be 3 for the red, green and blue components and it would be like having 3 image inputs for every color image. In the case of MNIST where the pixel values are gray scale, the pixel dimension is set to 1.

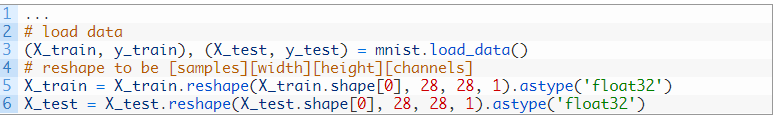


Fig. 3.3 Training dataset

As before, it is a good idea to normalize the pixel values to the range 0 and 1 and one hot encode the output variables

Convolutional neural networks are more complex than standard multi-layer perceptrons, so we will start by using a simple structure to begin with that uses all of the elements for state of the art results. Below summarizes the network architecture.

1. The first hidden layer is a convolutional layer called a Convolution2D. The layer has 32 feature maps, which with the size of 5×5 and a rectifier activation function. This is the input layer, expecting images with the structure outline above [pixels][width][height].
2. Next we define a pooling layer that takes the max called MaxPooling2D. It is configured with a pool size of 2×2.
3. The next layer is a regularization layer using dropout called Dropout. It is configured to randomly exclude 20% of neurons in the layer in order to reduce over-fitting.
4. Next is a layer that converts the 2D matrix data to a vector called Flatten. It allows the output to be processed by standard fully connected layers.
5. Next a fully connected layer with 128 neurons and rectifier activation function.
6. Finally, the output layer has 10 neurons for the 10 classes and a softmax activation function to output probability-like predictions for each class.

# 3.4 Code Implementation

As before, the model is trained using logarithmic loss and the ADAM gradient descent algorithm.

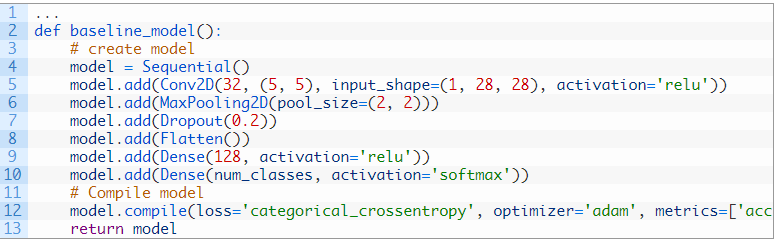


Fig. 3.4 create a model

Tying this all together and compiling the entire code together we get something like this.



Fig. 3.5 Main Code

Running the example, the accuracy on the training and validation test is printed each epoch and at the end of the classification error rate is printed.

[Results may vary](https://machinelearningmastery.com/different-results-each-time-in-machine-learning/) given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

And after running the model on the dataset on the Google Colab we get the following results

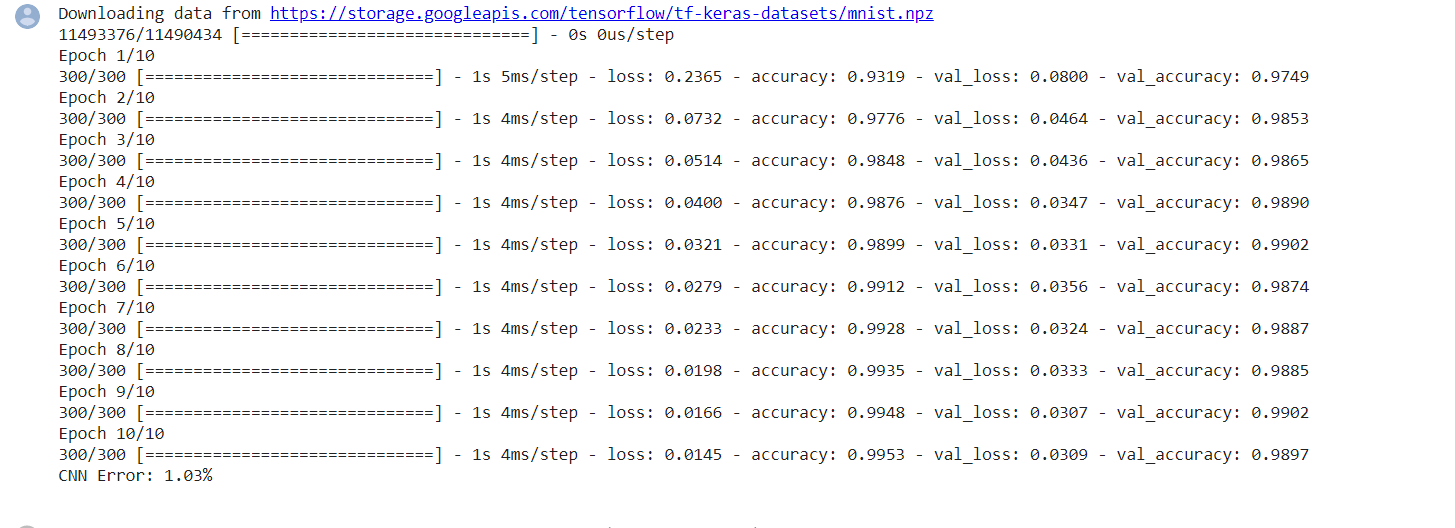


Fig. 3.6 Result of the model on dataset

Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a function predict\_digit() that takes the image as input and then uses the trained model to predict the digit.

Then we create the App class which is responsible for building the GUI for our app. We create a canvas where we can draw by capturing the mouse event and with a button, we trigger the predict\_digit() function and display the results.

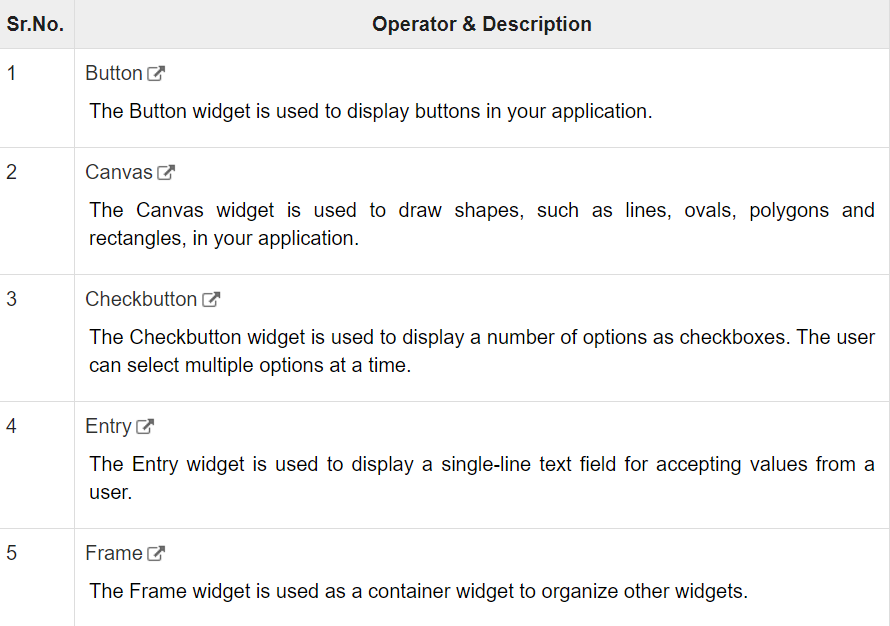
Tkinter is not the only [GuiProgramming](https://wiki.python.org/moin/GuiProgramming) toolkit for Python. It is however the most commonly used one. [CameronLaird](https://wiki.python.org/moin/CameronLaird) calls the yearly decision to keep Tkinter "one of the minor traditions of the Python world."

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

1. Import the Tkinter module.
2. Create the GUI application main window.
3. Add one or more of the above-mentioned widgets to the GUI application.
4. Enter the main event loop to take action against each event triggered by the user.

# Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.



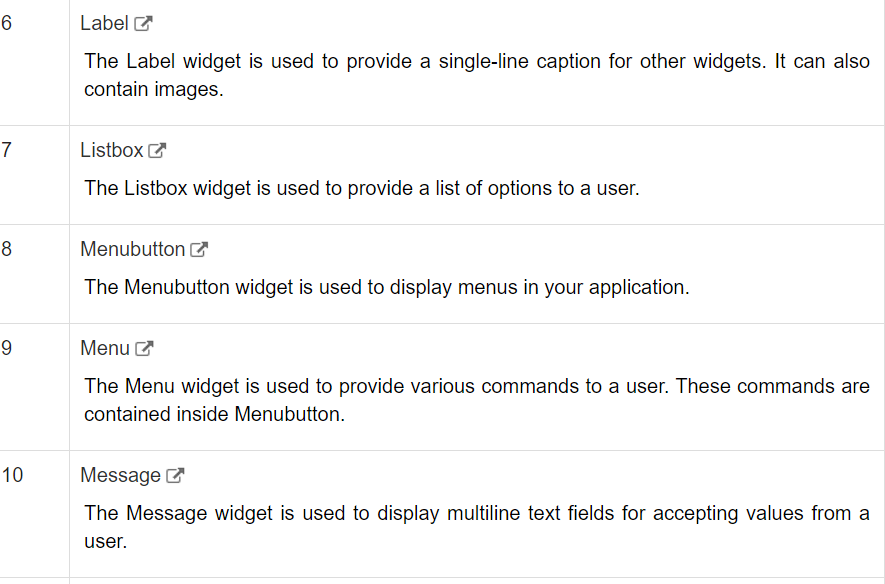
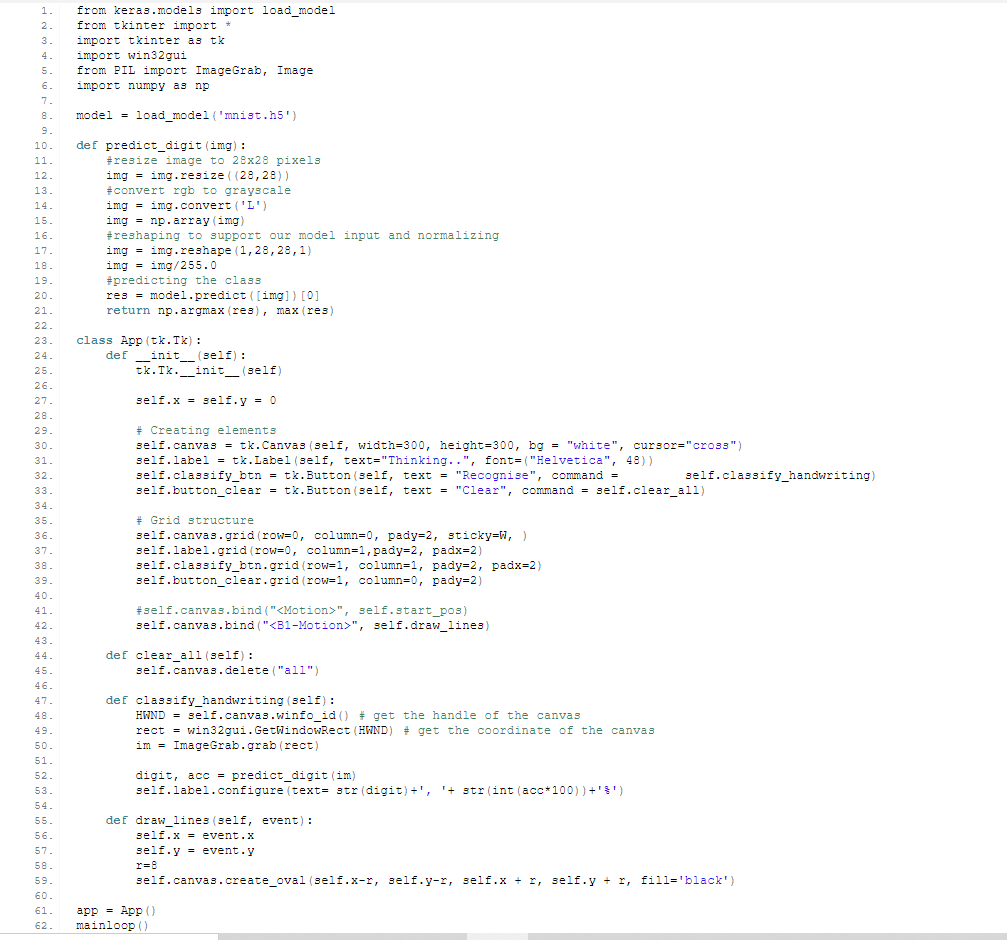


Table 1 Tkinter widgets used oftenly

Here’s the full code for our gui\_digit\_recognizer.py file:

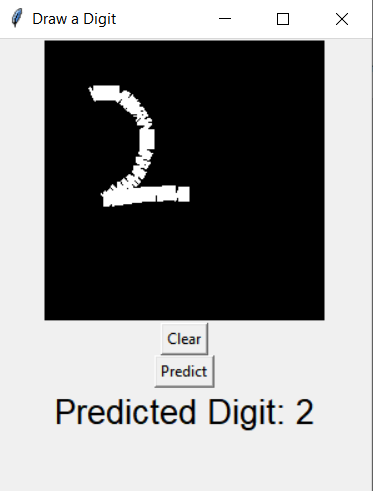
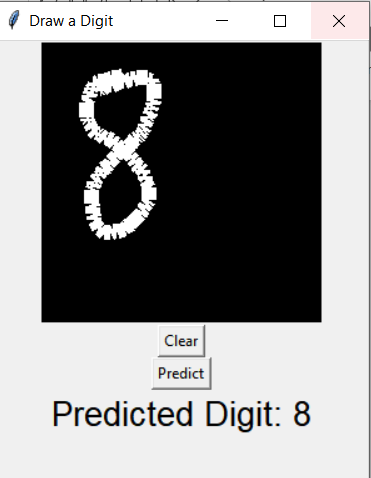


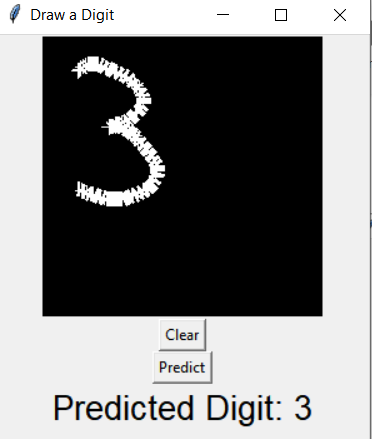
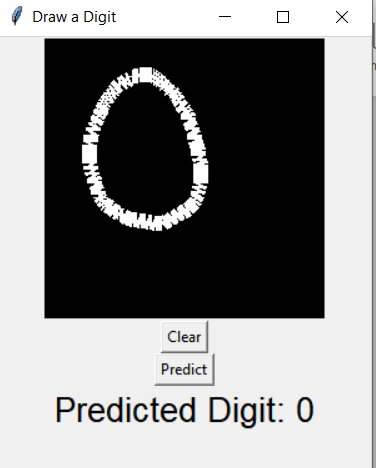
In python Tkinter application the canvas is created and used to paint on the mouse drag and

then it gets saved and re sized for that model architecture. The predicted result is shown in

the GUI Display.Some of the screenshot from the results are attached below.

# Results:





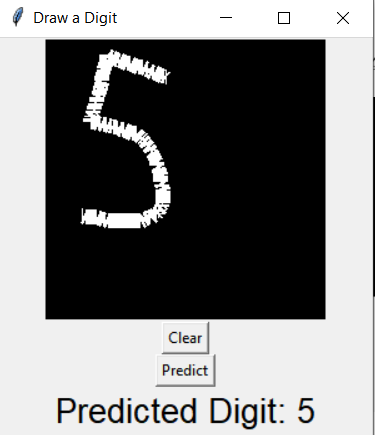


Fig.3.7 GUI output

# 3.5 Difficulties

Some of the difficulties faced during the preprocessing were discussed in this section. The figure 3.8 above on the left shows the handwritten image which is not even recognized by us. The images in the middle and the right were handwritten digit one and zero, when we pre process these images in Bewilder one is recognized as 2 (2 (objects) - 0 (holes) = 2) and zero is recognized as one (1 (object) - 0 (hole) = 1)

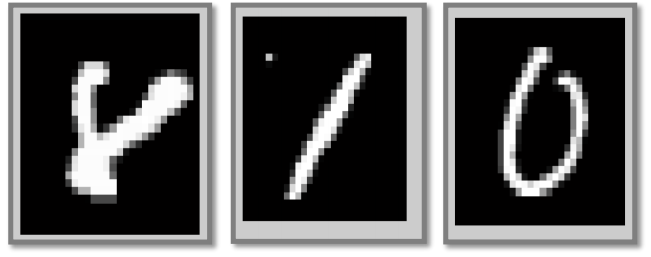


Fig 3.8 Digit not recognized correctly.

The overall accuracy of the handwritten image was 94.5%. Because it sometimes predicted Digit 8 as Digit 2. Also, since the raw data from the MNIST database, we used for training had been already processed, if we do more processing on the testing image we wrote, the accuracy can be improved further.

# CONCLUSION

In this post you discovered the MNIST handwritten digit recognition problem and deep learning models developed in Python using the Keras library that are capable of achieving excellent results.

Working through this tutorial you learned:

1. How to load the MNIST dataset in Keras and generate plots of the dataset.
2. How to reshape the MNIST dataset and develop a simple but well performing multi-layer perceptron model on the problem.
3. How to use Keras to create convolutional neural network models for MNIST.
4. How to develop and evaluate larger CNN models for MNIST capable of near world class results.

Then we we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library. We have created a graphic user interface that takes the image as input and then uses the trained model to predict the digit.

We have successfully built a Python deep learning project on handwritten digit recognition app. We have built and trained the Convolutional neural network which is very effective for image classification purposes. Later on, we build the GUI where we draw a digit on the canvas then we classify the digit and show the results.

**REFERENCES**

[1] A.U. Chukwu and K.A. Adepoju, On the power efficiency of artificial neural network (ANN) and the

classical regression model, Progress in Applied Mathematics 3(2) (2012), 28–34

[2] Ahmed, M., Rasool, A. G., Afzal, H., & Siddiqi, I. (2017). Improving handwriting-based gender classification

using ensemble classifiers. Expert Systems with Applications, 85, 158-168.

[3] Babu, U. R., Chintha, A. K., & Venkateswarlu, Y. (2014). Handwritten Digit Recognition Using Structural,

[4] Statistical Features and K-nearest Neighbor Classifier. International Journal of Information Engineering

and Electronic Business, 6(1), 62.

[5] Burel, G., Pottier, I., & Catros, J. Y. (1992, June). Recognition of handwritten digits by image processing and neural network. In Neural Networks, 1992. IJCNN., International Joint Conference on (Vol. 3, pp. 666-671). IEEE.

[6] D. Gorgevik and D. Cakmakov, “Handwritten digit recognition by combining SVM classifiers,” in Proceedings of the International Conference on Computer as a Tool (EUROCON '05), vol. 2, pp. 1393–1396, Belgrade, Serbia, November 2005. View at Publisher ·View at Google Scholar

[7] Ebrahimzadeh, R., & Jampour, M. (2014). Efficient handwritten digit recognition based on histogram of oriented gradients and svm. International Journal of Computer Applications, 104(9).

[8] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[9] Lawgali, A. (2016). Recognition of Handwritten Digits using Histogram of Oriented Gradients.International Journal

of Advances Research in Science, Engineering and Technology, 3, 2359-2363.

[10] Lee, Y. (1991). Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. Neural computation, 3(3), 440-449.

[11] Liu, C. L., Nakashima, K., Sako, H., & Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state of-the-art techniques. Pattern Recognition, 36(10), 2271-2285.