Artificial Intelligence
Assignment # 4
CS-E

17L-4010 Uzair Ahmed Bhatti
17L-6303 Ahmad Hayat
17L-6326 Ali Haseeb

Introduction

The aim of this assignment is to implement a classification algorithm including KNN, FFNN, SVM with HOG to recognize handwritten digits (0- 9). The scope of the assignment also included the elementary study of the different classifiers and combination methods, and evaluated the caveats around their performance in this particular problem of handwritten digit recognition. This report presents our implementation of the K-Nearest Neighbor, FFNN, SVM with HOG to recognize the numeral digits, and discusses the other different classification patterns.

Overview of the Project

Handwriting 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. 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 pattern classification methods to recognize the handwritten digits provided in the MINIST data set of images of handwritten digits (0-9). The data set used for our application is composed of 300 training images and 300 testing images, and is a subset of the MNIST data set [1] (originally composed of 60,000 training images and 10,000 testing images). Each image is a 28 x 28 grayscale (0-255) labeled representation of an individual digit. The general problem we predicted we would face in this digit classification problem was the similarity between the digits like 1 and 7, 5 and 6, 3 and 8, 9 and 8 etc. Also people write the same digit in many different ways - the digit ‘1’ is written as ‘1’, ‘1’, ‘1’ or ‘1’. Similarly 7 may be written as 7, 7, or 7. Finally the uniqueness and variety in the handwriting of different individuals also influences the formation and appearance of the digits.

Problem Statement

Given a set of greyscale isolated numerical images taken from MNIST database.

The objectives are:-

To recognize handwritten digits correctly.

To improve the accuracy of detection.

Scope of Study

MNIST Dataset The MNIST handwritten digit classification problem is a standard dataset used in computer vision and deep learning. Although the dataset is effectively solved, it can be used as the basis for learning and practicing how to develop, evaluate, and use convolutional deep learning neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data.

Handwritten-Digit-Classification-using-KNN

KNN algorithm K-Nearest Neighbors (or KNN) is a simple classification algorithm that is surprisingly effective. However, to work well, it requires a training dataset: a set of data points where each point is labelled (i.e., where it has already been correctly classified). If we set K to 1 (i.e., if we use a 1-NN algorithm), then we can classify a new data point by looking at all the points in the training data set, and choosing the label of the point that is nearest to the new point. If we use higher values of K, then we look at the K nearest points, and choose the most frequent label amongst those points.

However, in order to apply the k-Nearest Neighbor classifier, we first need to select a distance metric or a similarity function.

Running

The overall classification design of the MNIST digit database is shown in the following algorithm. Algorithm: Classification of Digits Input: Isolated Numeral images from MNIST Database Output: Recognition of the Numerals Method: Structural features and KNN classifier.

• Step 1: Convert the gray level image into Binary image

• Step 2: Preprocessing the Binary Image

• Step 3: Convert the Binary Image into a single Dimensional Array of [1,n]

• Step 4: Keep the label of each Array along with it.

• Step 5: Feed the classifier with the train_data set.

• Step 6: Repeat the steps from 1 to 5 for all images in the Sample and Test Database.

• Step 7: Estimate the minimum distance between feature vector and vector stored in the library by using Euclidean distances.

• Step 8: Feed the classifier with test_data set.

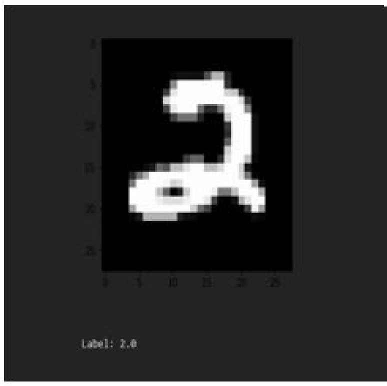
• Step 9: Classify the input images into appropriate class labels using minimum distance K-nearest neighbor classifier.

• Step10: End.

Implementation

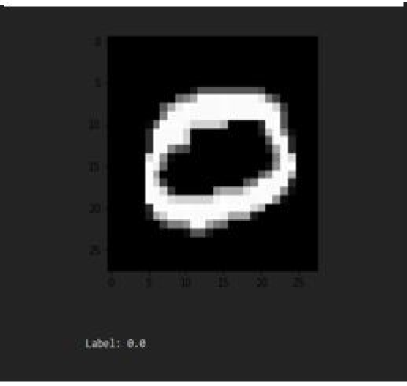
After reading in the data appropriately, we randomly split the data set into two pieces, train on one piece, and test on the other. The following function does this, returning the success rate of the classification algorithm on the testing piece. A run gives a surprisingly good 89% success rate. Varying , we see this is about as good as it gets without any modifications to the KNN algorithm Of course, there are many improvements we could make to this naive algorithm. But considering that it utilizes no domain knowledge and doesn’t manipulate the input data in any way, it’s not too shabby. As a side note, it would be fun to get some tablet software and have it use this method to recognize numbers as one writes it. Alas, we have little time for these sorts of applications.

Output



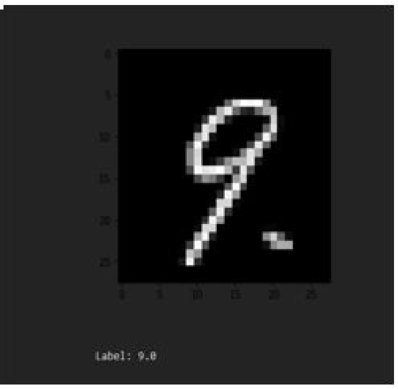
```
Label: 2.0
```

Screenshot (44)



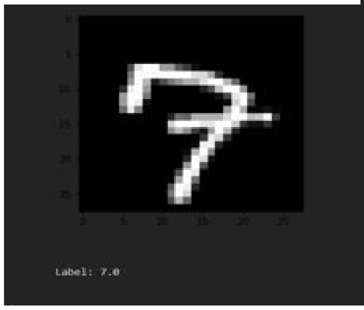
```
Label: 0.0
```

Screenshot (45)



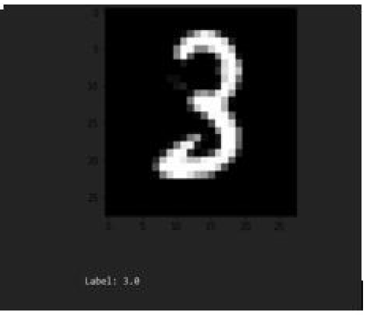
```
Label: 9.0
```

Screenshot (46)



```
Label: 7.0
```

Screenshot (48)



```
Label: 3.0
```

Screenshot (49)

KNN algorithm (classifier) is implemented with recognition accuracy of around 91-93%. The desired results have been obtained by training the machine first using the mnist_train data-set and later testing the obtained results using mnist_test data-set , to recognise the handwritten digit. Using KNN an accuracy rate of 92.8% was achieved.

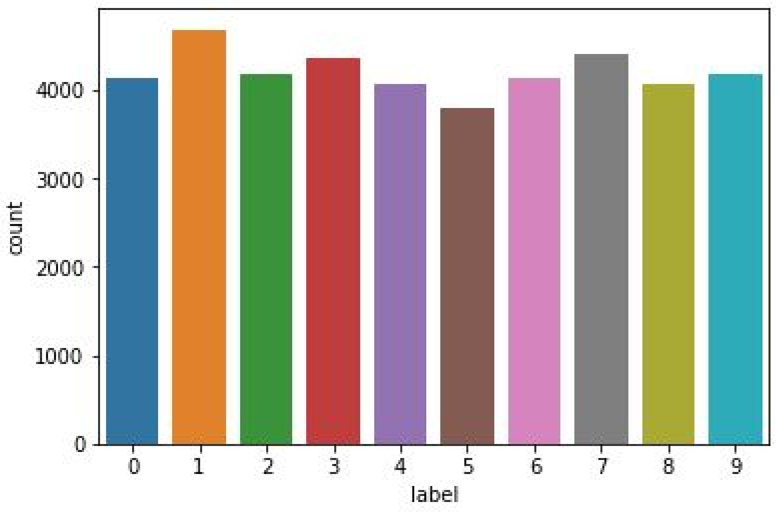
Handwritten-Digit-Classification-using-FFNN

The algorithm trains the model and obtains the results by running the code. The best accuracy that we were able to obtain was around 96%. As per the requirement of the assignment, the data was taken from the mnist library and were trained according to the FFNN model to get the best possible output.

Handwritten-Digit-Classification-using-SVM

We simply developed a model using Support Vector Machine to correctly classify Handwritten Digit from 0-9 based on their pixel values.SVM is one of the successful and most popular supervised learning classifiers in machine learning which constructs a hyper-plane in high order space which can be used as classification plane.It comes with different kernels such as **Linear,RBF,Sigmoid,Polynomial**.

To better understand Data we take a simple representation using matplotlib.

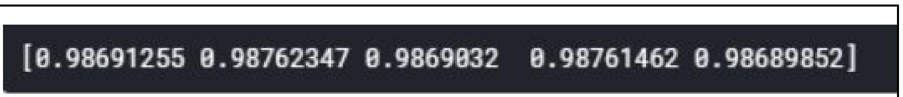
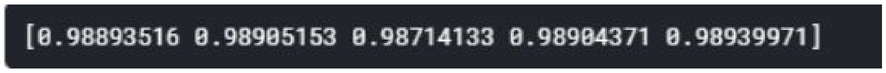


It shows us the label for each digit and total count in the test data.

Accuracy Comparison using different Kernels over 5 epochs(meaning over 5 time run over data)

Accuracy Comparison using different Kernels over 5 epochs(meaning over 5 time run over data)

The result shows us that our model gives accuracy of somewhat 98% on the training data.

Linear	 `[0.98691255 0.98762347 0.9869032  0.98761462 0.98689852]`
RBF	 `[0.98893516 0.98905153 0.98714133 0.98904371 0.98939971]`

FFNN MODEL:

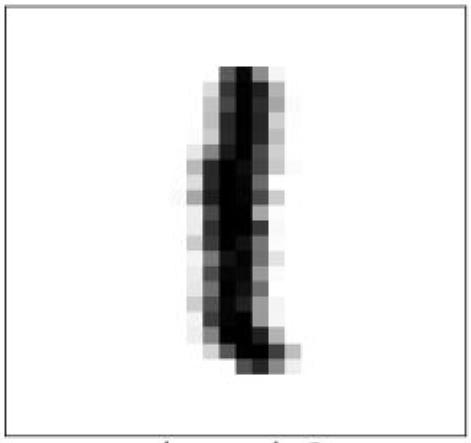
Layers	**Accuracy (correct to two dec)**
2	0.95
3	0.96
4	0.96
5	0.96

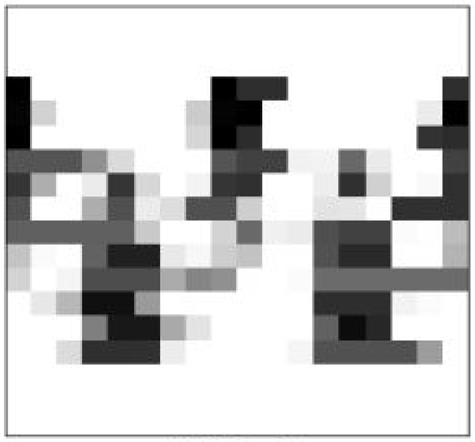
The accuracy did not improve much even after adding 5 layers.
Hence, the clear number for the best number of ‘hidden’ layers is 3.

Neurons (3 layers)	**Accuracy (correct to two dec)**
12 , 12, 10	0.88
32, 32 ,10	0.95
64, 64, 10	0.96

Effect of applying HOG on Image of Digit

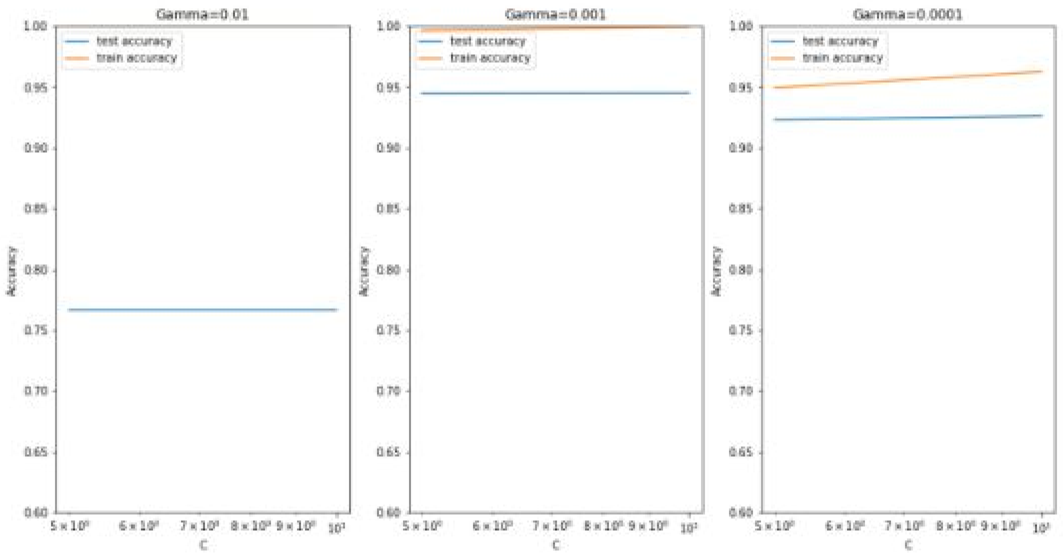
HOG divides the input image into small square cells (here we used 9×9) and then computes the histogram of gradient directions or edge directions based on the central differences.


skewed: 1


HOG: 1

Effect of Increasing Value of C Parameter on Accuracy(Hyper parameter Tuning)

Hyperparameter tuning helps us understand the best value for our model to help us obtain the best accuracy.The Below graph helps us under how value of c and gamma helps us to achieve a greater accuracy.



Conclusion

In this assignment, we learnt about machine learning classifiers — the k-Nearest Neighbor classifier, FFNN, SVM with HOG. The k-NN algorithm classifies unknown data points by comparing the unknown data point to each data point in the training set. This comparison is done using a distance function or similarity metric. Then, from the k most similar examples in the training set, we accumulate the number of “votes” for each label. The category with the highest number of votes “wins” and is chosen as the overall classification.

The assignment shows the whole process of supervised classification from the data acquisition to the design of a classification system and its evaluation.

Using the FFNN model can solve complex problems effectively but there is still somewhat of a catch to it. Adding unnecessary layers might or might not increase the accuracy of the classifier. After a certain number of adding the ‘hidden layers’, the code takes more time to train the data but does not generate more accuracy. Saying that, we can safely say that using FFNN classifier to classify mnist data was one the most effective (and simple!) solutions that one can find.