Handwritten Digit Recognition

Shivani Matta [1], **Mukul K. Rajak**[2] **and Gaurav Khurana** [1]
[1] Electronics and Communication Engineering, Indraprastha Institute of Information Technology, Delhi
[2]Computer Science Engineering, Indraprastha Institute of Information Technology, Delhi

Abstract

Humans' reliance on machines has never been so high that from classifying various objects in photographs to appending sounds to silent movies, almost everything can be achieved using deep learning and machine learning techniques. One of the significant fields of research and development with many possibilities is Handwritten text recognition. A machine can read and interpret handwritten characters from many sources like paper, photographs, laptop or mobile screens, and other mediums, known as Handwritten Text Recognition (HTR)[3]. This paper emphasizes Handwritten Digit Recognition, a subset of HTR, wherein we are focusing on to be able to classify handwritten numbers from 0-9. With the help of MNIST Datasets, we will train various Machine learning models like Support Vector Machine (SVM)[3], Random Forests[2], Convolutional Neural Networks (CNN)[1][2][3][6], and Multi-Layer Perceptron (MLP)[3]. This paper's primary objective is to get the best possible model for digit classification based on all the models' accuracy and execution time. Code of this paper can be found on github

Keywords- MNIST, Support Vector Machine, Perceptron, Neural Networks, Decision Tree

1. Introduction

Handwritten digit recognition is the computer's ability to read and accurately classify the human handwritten digits from different sources like photographs, papers, mobile displays, etc., into ten predefined classes (0-9). It is one of those topics involving boundless research in the field of Artificial Intelligence. From number plate detection to sorting postal mails, from bank cheque processing to reading numeric entries in forms filled up by hand, digit recognition has numerous applications. Due to other peoples' different writing styles, it is a challenging task to perform digit classification. This research brings up a comprehensive comparison between many known classifiers available in Machine Learning and Deep Learning. The algorithms we used are Convolutional Neural Networks, K-Nearest Neighbor, Logistic Regression, Support Vector Machines, Multilayer Perceptron, Decision Trees, Random Forests, and Gaussian Na¨ıve-Bayes. The comparison is carried out based on accuracies and training times.

The accuracy of a model is paramount because more accurate models will make better predictions. For real-world scenarios, a model with low accuracy is not suitable. In bank cheque processing, the system recognizes the amount filled and date written on the cheque, and high accuracy is crucial here. Incorrect recognition can cause significant damage, which is undesirable. So this paper emphasizes comparing different classifiers based on their accuracy to apply the best model in practical situations with the margin of error being as low as possible.

This report aims to provide a good understanding of many classifiers available in Machine learning and deep learning for handwritten digit classification. It also gives information regarding the efficiency of the algorithms in performing the required task. The upcoming sections of the paper will discuss some related works in this domain, followed by a description of the dataset used to train the classifiers and the required pre-processing. This will be followed by the methodology and implementation of the classifiers for a better understanding and, lastly, the results' analysis and conclusion. The last section of the paper has citations of references used.

2. Literature Survey

2.1. Spatially-sparse convolutional neural networks[2] link

This paper explains CNN's sparse data since CNN performs well on handwritten digit classification and image classification.

CNN consists of an input layer, hidden layers, and softmax classification layer. The authors also present the idea of online isolated character recognition. Here online means that we capture the character as a path using a touchscreen or electronic stylus, rather than being stored as a picture.

2.2. MNIST-MIX: A Multi-language Handwritten Digit Recognition Dataset[6] link

This paper presents a dataset of handwritten digits in 10 different languages, collected from MNIST, EMNIST, and Street View House Numbers SVHN(house numbers in Google Street View images).

A mix of all these data is providing significantly more challenging and unsolved real-world problems.

2.3. Effective Handwritten Digit Recognition using Deep Convolutional Neural Network [2]link

This paper proposes the CNN approach to HDR on the MNIST dataset giving 98.51It an idea about how CNN works. Intuitively, convolution is a process of applying various filters on images to highlight the details in it.

The paper presents a step-wise approach to the problem - data pre-processing, data encoding, model construction, training validation, and results.

3. Dataset

3.1. MNIST

("Modified National Institute of Standards and Technology")

The data initially has 60k samples of size 28*28. Converted it to 784*1 (direct data in this format is also available).[3][6][2]

The first column is the actual label (i.e., 0,1,..9) and rests 784 columns are features which have the value in range 0 to 255 representing the intensity of colour in that particular pixel.

Some columns have a value 0 always for all the 60k samples. So we have dropped those columns.

3.2. EMNIST

EMNIST Extended version of MNIST dataset and has similar features. Along with digits, it also has letters in the class labels

3.3. Pre-Data Analysis

We have Plotted some samples from the MNIST dataset. We checked for data-imbalance using frequency plots.[3] Figure 1. Samples from MNIST dataset Figure 2. Class Frequencies (MNIST)

3.4. Data Pre-processing

We have performed dimensionality reduction of input features from 784 to 50 using PCA (Principal Component Analysis) and SVD (Singular Value Decomposition), which accounts for almost 90%) of the total variance. We have used max-pooling to reduce the computation time, retaining useful features of our dataset.

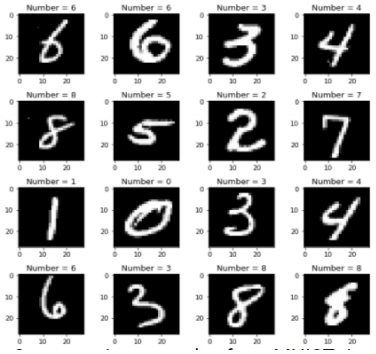

Figure 1. Samples from MNIST dataset[5][2]

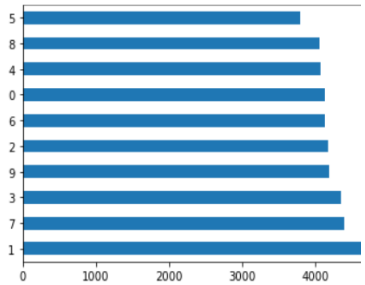

Figure 2. Class Frequencies (MNIST)

T-SNE is used for data visualization.[5] Since the pixel values vary from 0 to 255, we also used StandardScaler to standardize our data as it is a common requirement for many machine learning models.

EMNIST An Extended version of MNIST dataset and has similar features. It has letters along with the digits in training examples.

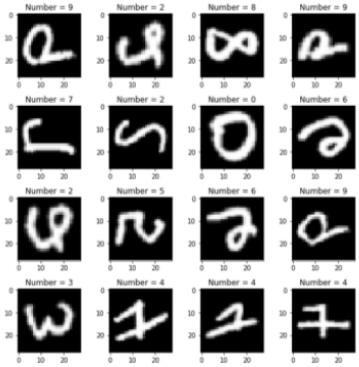

Figure 3. Samples from EMNIST dataset

4. Methodology

We have used MNIST and EMNIST dataset to train and test our model.

MNIST dataset consists of over 70,000 handwritten images of digits and letters of resolution 28 x 28 pixels.

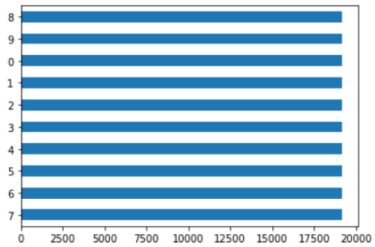

Figure 4. Class Frequencies (EMNIST)

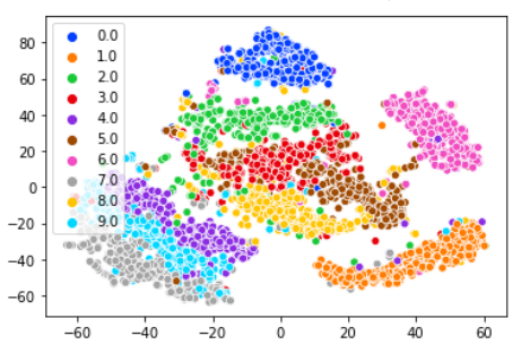

Figure 5. TSNE visualisation on PCA data (MNIST)[5]

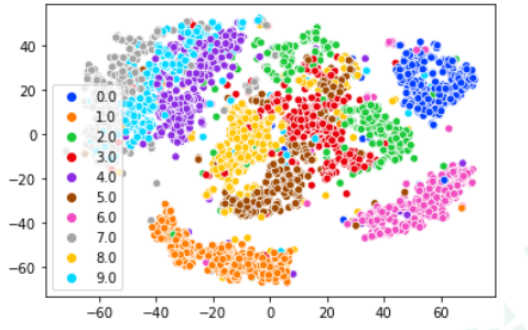

Figure 6. TSNE visualisation on SVD data (MNIST)

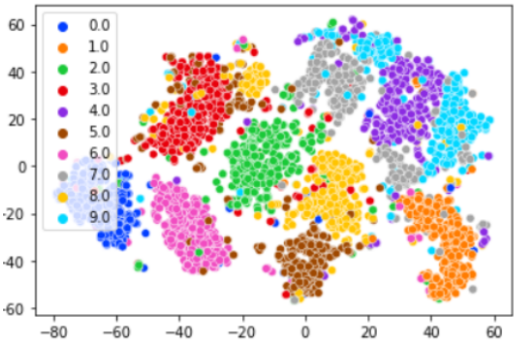

Figure 7. TSNE visualisation on EMNIST data

- Building up the model using the processed data.
- Training our model for appropriate number of epochs and avoiding overfitting of our model. K-fold CV can be used here as well.
- Once we've trained our model, we will pre process our testing data the same way we did the training data and then predict the labels for the testing data.
- After that we've measure the accuracy of different models and have benchmark our model on kaggle platform.
- *Note: Time is measured when we trained and tested the model on GPU enabled google colab's hardware.*

4.1. CNN

Convolution Neural Network consists of input layer, output layer and hidden layers. Input layer and hidden layers are 3D layers of NxNxM where NxN is the dimension of image and M is the no of channels in the layer.[1][3][4]

4.1.1 Convolution Layer

- Input matrix- NxNxM, filter matrix- fxfxM = Output Matrix - (N-f+1)x(N-f+1)x1
- Input matrix is convolved with filter (Element wise multiplication)
- When used fc such filter
- Output of this layer - (N-f+1)x(N-f+1)x fc

4.1.2 Pooling Layer

- It reduces the number of parameters to reduce computation.
- Input is processed through many such convolution layer and pooling layers, the size of input (N) decreases when going through this network.

4.1.3 Fully Connected Layer

- The matrix is flattened and classified using activation function such as softmax, ReLU.

4.2. KNN

K-Nearest Neighbour is one the ML Algorithm based on Supervised learning technique which can be used to solve classification and regression problem. KNN is a lazy learning algorithm; it does not perform much while training and do the real computation while testing. Its works even for non linearly separable data. [2]

This paper has used KNN classifier from Sklearn library for multi-class classification on different Datasets (i.e. MNIST, EMNIST). It takes the most frequent class value for

each test data. And No. of nearest neighbours are already passed as a hyperparameter. KNN is very fast and gives comparable accuracy.

4.3. Logistic Regression

Logistic regression is a supervised machine learning algorithm which can be used for prediction of probability for classification. This model is for binary class classification, but it can be used for multiclass classification as well through the one-vs-rest rule or changing the loss function to cross-entropy loss.

This paper has used Logistic regression from sklearn library for multiclass classification using the one-vs-rest scheme. This model is very fast.

4.4. SVM

Support vector Classifier is used to fit to the data we provide. It returns a hyperplane that divides or categorize our data. SkLearn library has SVC, NuSVC and LinearSVC classifiers which can be used to perform multi-class classification on a different type of datasets (sparse or dense).[3][2]

This paper has used SVC classifier from sklearn library for the classification of MNIST,EMNIST Datasets keeping all the hyperparameter as default. After pre-processing of the data, the SVC model has been used. First, it's fitted using train dataset and then used for classification. Finally Accuracy score and confusion matrix has been plotted.

4.5. MLP

In Multi-Layered Perceptron, we prepare a neural network which consists of an input layer, hidden layer and an output layer. It is also known as feedforward ANN. The input layer and hidden layers are connected using weights Wij where i is the input layer, and j is the hidden layer.[3] The activation function we are using is ReLU which adds non-linearity to our model. Input to this network is the 50 dimensions array (using PCA we reduced dimensions to 50 from 784).

MLP Classifier from sklearn library has been used to perform classification. Number of output layers are 10 (total number of labels) and the size of the hidden layer is taken to be 500. The model has been trained using the training dataset, and predictions are made on the testing set. Accuracy score and confusion matrix has been computed.

4.6. Decision Trees

Decision Tree is a flowchart-like tree structure where internal nodes represent features; the branch tells us the decision rule and leaf nodes gives us the outcome. The root node of the entire tree learns to partition by anticipating the best predictor.

We apply the approval condition recursively to get the right branch at every node. DT Classifier from sklearn was used to perform the classification. One of the important parameters it takes is the optimal depth of the tree. We performed a grid search by trying out various values for optimal depths and figured out the best performing one. For EMNIST Dataset, the optimal depth was 21, whereas, for MNIST, it was 13. We trained the model using these optimal depths and got the accuracy scores.

4.7. Random Forest

This is an ensemble technique. Many decision trees(uncorrelated) are trained on various sub-sets of data and classification is based upon the output of all these decision trees using averaging, voting etc.

4.8. Gaussian Naive Bayes

GNB uses Naive Bayes Algorithm to correctly classify model. It assumes independence between the given features. The sample is given label having maximum given

$$Y \leftarrow \arg\max_{y_k} \; P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

Figure 8. GNB

probability. This algorithm is based on Maximum A Posteriori (MAP).

5. Results and Analysis

After training eight different classifiers in Machine Learning and Deep Learning, this paper compares all these algorithms' testing accuracy and training time on both EMNIST and MNIST datasets. It has been found that CNN had the best accuracy and Gaussian Naive Bayes had the worst accuracy for both the datasets used.

In this paper we have taken 8 hidden layers with (600,500,400,300,200,100,50,25) hidden units per layer and used tanh activation function for each hidden layer and softmax for the output layer. We've noticed that if we increase the no. of hidden layers then we're getting more accuracy.

The execution time for Random Forests was moderate and was the highest for CNN[4]. The running time of the algorithms depends on the number of operations performed. Table 1 and Table 2 have the accuracy score and training time for all the models used in this paper.

6. Conclusion

This research paper has implemented eight different models from simple Logistic Regression to more complicated Convolutional Neural Networks for classifying handwritten digits using both MNIST and EMNIST datasets. It

Model	Accuracy	Time(sec)
Logistic Regression	90 %	10.8221
Gaussian Naive Bayes	79 %	0.0573
Decision Tree	83 %	2.7673
Random Forest	94 %	188.1144
K - Nearest Neighbours	95.1 %	17.0200
Support-Vector Machines	96.5 %	29.8179
MLP	97.2 %	53.1602
CNN	99.4 %	861.9948

Table 1. Accuracy and time of Different models on MNIST

Model	Accuracy	Time(sec)
Logistic Regression	92.7 %	117.11
Gaussian Naive Bayes	75.14 %	0.311
Decision Tree	86.71 %	25.14
Random Forest	95.37 %	1237.2547
K - Nearest Neighbours	97.38 %	561.5915
Support-Vector Machines	98.35 %	1744.4068
MLP	98.58 %	325.67
CNN	99.1 %	1908.3052

Table 2. Accuracy and time of Different models on EMNIST

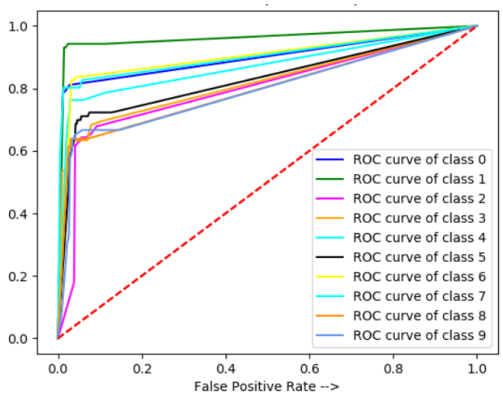

Figure 9. ROC curve for DT classifier

compared all these algorithms based on testing accuracy and training time to appraise the most suitable and accurate model. Owing to its simplicity, SVM is one of the best classifiers, but when it comes to more complex real-world scenarios, MLP and CNN have an edge.

It is evident from the tables that CNN provided the best accuracy score. Still, if we consider the training time, then MLP gives almost similar accuracy in an enormously efficient amount of time. CNN is capable of delivering the best accuracy in virtually every complex classification problems.

Hence, it solidifies that CNN is the best candidate among all the models we tested based on the accuracy of rate and learning capabilities.

7. References

[1] Alazzawi, M., Albehadili, H., Al-Wzwazy, H., Alzubaidi, L. and Rashed, J., 2017.*Fast And Accurate Real Time Pedestrian Detection Using Convolutional Neural Network.* [online] ResearchGate. Available at: https://bit.ly/35MSWim [Accessed 22 November 2020].

[2] Bhanu Prakash, K., P, R., SS Bharadwaj, Y., S, S. and V.P, S., 2020. *Effective Handwritten Digit Recognition Using Deep Convolution Neural Network.* [online] Paperswithcode.com. Available at: https://bit.ly/37NPmaj [Accessed 22 November 2020].

[3] Dixit, R., Kushwah, R. and Pashine, S., 2020. *Handwritten Digit Recognition Using Machine And Deep Learning Algorithms.* [online] ResearchGate. Available at: https://bit.ly/2HKheRP [Accessed 22 November 2020].

[4] Graham, B., 2014. *Spatially-Sparse Convolutional Neural Networks.* [online] Paperswithcode.com. Available at: https://paperswithcode.com/paper/spatially-sparse-convolutional-neural [Accessed 22 November 2020].

[5] Hintea, D. and Karayaneva, Y., 2018. *Object Recognition In Python And MNIST Dataset Modification And Recognition With Five Machine Learning Classifiers.* [online] Available at: https://bit.ly/3mx1oZQ [Accessed 22 November 2020].

[6] Jiang, W., 2020. Papers With Code - MNIST-MIX: *A Multi-Language Handwritten Digit Recognition Dataset.* [online] https://arxiv.org. Available at: https://arxiv.org/abs/2004.03848 [Accessed 24 November 2020].