Hand Written Digit Recognition Using Tensorflow and Python

Shekhar Shiroor
Department of Computer Science
College of Engineering and Computer Science
California State University-Sacramento
Sacramento, CA 95819-6021, USA
shekharshiroor@csus.edu

Abstract

People write in as many different ways as there are stars in a galaxy. This leads to development of different patterns in writing. Costly manual labor is required to do a mundane and tedious job of converting the physical written data and information into digital form for storing it in a digital form.

My paper discusses the solution to a part of problem as I have limited the scope to only the hand written digits(0-9). I have trained a model using deep neural networks for digit recognition using Google’s Machine Learning tool Tensorflow and Python Programming language. I have used the ‘MNIST DATABASE’ which consist of training and test set for hand written digits (0-9) of size (28x28) pixels i.e. 784 pixels. The data set consist of 60,000 training data and 10,000 test data. The limitation of this model will be if digits other than (0-9) are given then the model will not be able to recognize and classify it and the model will be able to predict numbers only in black and white images.

Keywords: Machine Learning, Deep neural networks, MNIST data set, Tensorflow, Tensorboard, Python.

1. Introduction

Since ancient times people have been recording information maybe in the form of cave painting during the stone ages and stone inscriptions later. After decades of centuries before computers were invented all the information was stored on paper and this paper information could be lost or destroyed accidently, hence it’s considered very inefficient form of storage. After the computers were invented the information storage on the computers was considered quite reliable as multiple copies of same information could be made easily and the data storage in digital form is more secure compared to physical i.e. written form of storage.

As a result of this many organizations started investing huge revenue on converting their data from physical to digital form via manual labor. Later when there was advancement in the field of digital imagery and machine learning research started to combine these both so that physical information could be directly converted into digital information without human intervention i.e. manual labor. My paper focuses on the part of identifying and converting handwritten digits into their respective digital form i.e. identifying the images of digits(0-9) and converting them into digital form for storage.

I will be focusing on how I have developed a 4 layer deep neural network model and why ?. How the MNIST data set was used, my results for various models of deep neural networks. Visualization of data, How Tensorflow is used to develop and visualize the data.

2. Data Set and Preprocessing

I have used the MNIST data set for implementing my project. MNIST stands for ‘Modified National Institute of Standards and Technology database’ which consist of large number of images of hand written digits(0-9). MNSIT data set is a subset of NIST data set (National Institute of Standards and Technology database) which consist of hand written images for Letters (A-Z, a-z), Numbers (0-9) and special symbols as well. The MNIST data set consist of black and white images of both training data set and test data set. There are in all 60,000 training data images and 10,000 test data images. The image data is stored as 1D array of pixel values in the Ubyte format.

The MNIST data set is already a preprocessed data set. For preprocessing the data “The original black and white (bilevel) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-
The aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.”[1]

3. Approach

Various approaches have been used to predict MNIST data set like Support Vector Machines(SVM), Linear regression, Linear Classification, K Nearest Neighbor, Neural Networks, etc. Among all these approaches the accuracy obtained via many of these algorithms was not high enough as compared to Artificial Neural Networks which is discussed later in the paper. As a result of which for this project implementation my approach was based on Neural networks especially Deep Neural networks. Initially I tried single layer neural network but the results were not so good hence I went on to multiple layer neural network i.e. deep neural network as the number of hidden layers in the neural network was more than two(>=2). Initially for deep neural networks I trained the model on 3 layer neural network each hidden layer having 500 nodes, after this the accuracy was still lower than expected so I went on to four hidden layer neural network each hidden layer having 500 nodes, for training the model the MNIST data set is used. It consist of image data i.e. pixel values in the form of one dimensional array which was used for training the model. The test data is also stored in same manner for testing the accuracy of the model. After training the model the model can be saved for later use as well.

4. Input Data

The input data is the MNIST data set which is the images of Hand written digits (0-9) stored in the form of 1D(one dimensional) array of pixels.

5. Implementation

The project is implemented using Google's machine Learning tool called Tensorflow. Tensorboard is used for data visualization and model visualization purposes. My model developed on Tensorflow uses RELU(Rectifier Linear Unit) at each layer to get smooth approximation for the output to its respective input 'f(x) = ln(1+e^x)’[4]. Softmax function is used at each hidden layer to predict the output as probabilities. Cross entropy loss is used at each hidden layer to measure the error in the output.
Adam optimizer is used to update the weights on hidden layers to reduce error and maximize output.

5.1. Tensorflow and Tensorboard

TensorFlow is an open source software library for machine learning across a range of tasks, and developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is currently used for both research and production at Google products[3]. TensorBoard is a suite of web applications for inspecting and understanding your TensorFlow runs and graphs.[5]

5.2. Softmax and Cross Entropy Loss

Softmax is the hidden layer activation function given by

\[ \sigma(Z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \]

*Figure 4. Softmax layer calculation[2]*

Cross Entropy loss is used to measure loss at Softmax layer which is given by

\[ L(w) = \frac{1}{N} \sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N} \sum_{n=1}^{N} \left[ y_n \log \hat{y}_n + (1-y_n) \log(1-\hat{y}_n) \right] \]

*Figure 5. Cross Entropy Loss calculation[2]*

5.3. Adam Optimizer

The adam optimizer is sophisticated version of gradient decent optimizer. “Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.”[15] The adam optimizer reduces the loss much faster compared to gradient decent optimizer so it is much preferred compared to gradient descent. Adam optimizer optimizes weights of the nodes after every epoch. An epoch is a cycle consisting of Feedforward in the network and after reaching the output layer ,based on the error back propagate while updating the weights on nodes at every layer.

6. Output and Accuracy

The output during the training phase of the model is the loss obtained at every epoch as shown in the diagram. After running the model for a number of epochs then the final output which is the accuracy is calculated on the test set of the MNIST data set. Here the accuracy is 96.28% approx. as shown in the diagram.

*Figure 6. Accuracy and Loss Reduction*

Other than the accuracy the output prediction obtained for me on the test set is in the form of ‘One Hot Encoding’ which is an 1D array of zeros and one. The array position of one indicates the number predicted (0-9) by the model as shown in the diagram below.

*Figure 7. One Hot Encoding of Digit(0-9)*

7. Visualizations

Tensorflow provides various visualizations which are realized using Tensorboard, Some of the Visualizations are

7.1. Scalars

Scalars consist of Accuracy graph and cross entropy loss graph which is explained as follows.

7.1.1. Accuracy graph. This graph shows the accuracy of the model over the number of epochs.
7.1.1. **cross entropy loss graph.** This graph shows the loss rate of the model decreases over time.

7.2. **Histograms**

Histogram depicts the visual distribution of numerical data over period of time. Here the weights and biases used in the DNN model is depicted visually over time.

7.3. **Embedding Visualizer**

The embedding visualizer is used to visualize the MNIST data in high dimensional space. The Embedding Visualizer uses PCA(Principal component Analysis) to visualize the data in 3D. “Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components (or sometimes, principal modes of variation). The number of principal components is less than or equal to the smaller of the number of original variables or the number of observations.”[10]. Also along with PCA, T-SNE can be used to visualize animation of data cluster formation over period of time.
8. Result Discussion and Related Work

The MNIST data set is a very popular data set for machine learning. When I ran my model initially with 3 hidden layers having 500 nodes each and 10 Epochs I got an accuracy of 92% approx. Later I ran the same model for 20 Epochs and got result of 93% approx. Increasing the number of Epochs was not giving optimal results. So I used a new model having 4 hidden layers and each layer having 500,1500,1500,500 nodes respectively for 10 Epochs and I got a result of 95% approx. Later I ran the same model for 20 Epochs and saw that that the Accuracy reduced to 94% Approx. So finally I ran the model for 15 Epochs and got the Accuracy of 96.28% which is the final accuracy obtained by me.

Other than Neural Network’s the MNIST data set results for some algorithms are using KNN in which additional preprocessing was done like DE skewing, blurring, Noise removal and an accuracy of 98.27% was obtained[1]. Also with Boosted Stumps and additional preprocessing of haar features 99.13% Accuracy was obtained[1]. SVM with additional preprocessing of DE skewing gave an accuracy of 98.9%[1]. Lastly by using conventional neural network and additional preprocessing of width normalization gave an accuracy of 99.77%[1].

9. Future Work and Conclusion

The accuracy of my deep neural network model can be increased by implementing more hidden layers and maybe more number of epochs. Use CNN with less layers to get better accuracy instead of traditional DNN as CNN gives better accuracy.

Hence it can be concluded that greatest accuracy with minimum additional preprocessing on MNIST data set is obtained with Neural Networks specifically Conventional Neural Network so Neural Networks can be said as the best approach for processing MNIST data set.

10. References


