The Modified NIST Database of handwriting digits

Introduction

Handwriting digits recognition is the ability of a computer system to recognize the handwritten inputs like digits, characters etc. from a wide variety of sources like emails, papers, images, letters etc. This has been a topic of research for decades. The most popular handwriting digits dataset is the MNIST database

The MNIST database was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than SD-1. The reason for this can be found on the fact that SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Hence, it was necessary to build a new database by mixing NIST's datasets. The new training set was completed with enough examples from SD-3, starting at pattern # 0, to make a full set of 60,000 training patterns. Similarly, the new test set was completed with SD-3 examples starting at pattern # 35,000 to make a full set with 60,000 test patterns. Only a subset of 10,000 test images (5,000 from SD-1 and 5,000 from SD-3) is available on this site. The full 60,000 sample training set is available. The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. Our test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000 pattern training set contained examples from approximately 250 writers [1].

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-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. The labels are numbers from zero to nine.

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

Pixels are organized row-wise. Pixel values are 0 to 255. 0 means background (white), 255 means foreground (black).

Figure1. Dataset illustration

These files are not in any standard image format. All the integers in the files are stored in the Binary format (high endian). From figure1, it is concluded that we can read the training set label from the 8th offset because that the first four is magic number and the 4th to 7th is the number of labels. For the same reason, the training set image file should be read from the 16th offset. Figure 2 is the visualization of the training set label file. It can be seen that the number from offset 8 is no bigger than nine, and the number from offset 16 is no bigger than 255 in the training set image file. The image is a 28x28 matrix in which the value of each element is among 0 to 255. Each element represents the color of one pixel and the number represents how black the pixel is. 0 means white and 255 means black [2].

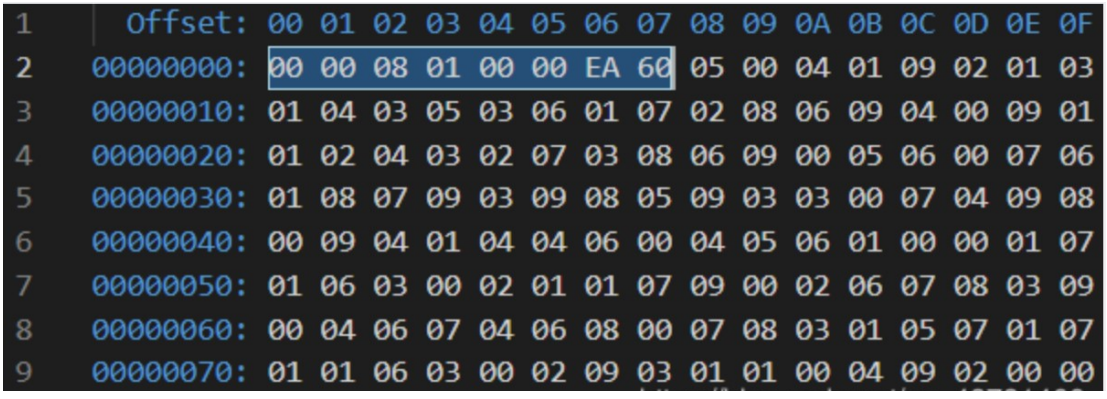

Figure 2. Visualization of the training set label file(hex format)

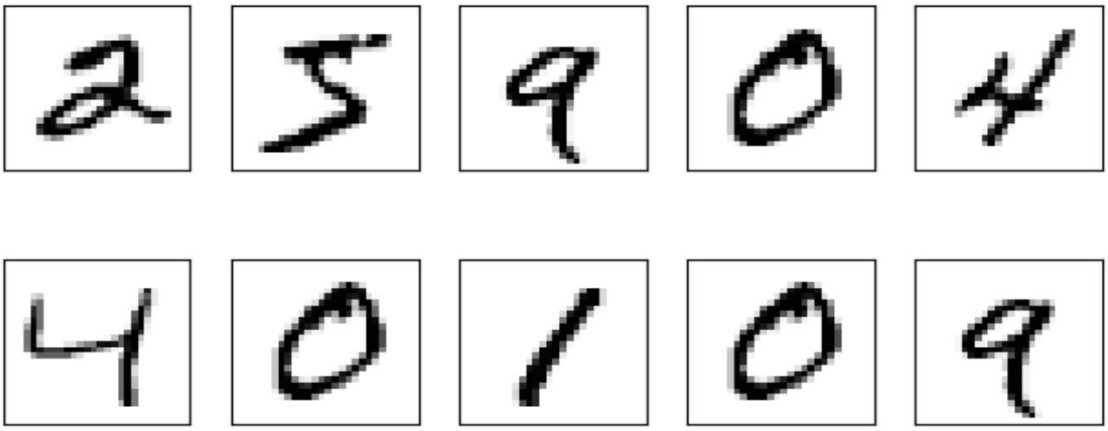

Figure 3. Images of the handwriting digits

The dataset brings us a classification problem which is supervised learning. All images should be classified to ten classes.

Neural Network Structure

There is a pre-process of the original data. The Neural Network requires each input data to be a vector. Hence, the original 28x28 image matrix should be flatten into a vector with the length of 784. Moreover, the label should be translated into the one hot encoding in which only one bit of the state vector is asserted for any given state [3]. All other state bits are zero. Thus, if there are n states then n state flops are required. State decode is simplified, since the state bits themselves can be used directly to indicate whether the machine is in a particular state. No additional logic is required. There are numerous advantages to using the one hot design methodology:

• One hot encoding enables the neural network model to calculate the distance between the target and prediction more accurately such as Euclidean distance, Cosine similarity

• Maps easily into register-rich FPGA architectures such as QuickLogic and Xilinx.

• One-hot state machines are typically faster. Speed is independent of the number of states, and instead depends only on the number of transitions into a particular state

• Don’t have to worry about finding an "optimal" state encoding. This is particularly beneficial as the machine design is modified, for what is "optimal" for one design may no longer be best if you add a few states and change some others. One-hot is equally "optimal" for all machines.

• Modifications are straightforward. Adding and deleting states, or changing excitation equations, can be implemented easily without affecting the rest of the machine.

• There is typically no area penalty over highlyencoded machines.

• Critical paths are easy to find using static timing analysis.

• Easy to debug.

Therefore, the label should be transformed from one integer into a vector with length of ten. For instance, the vector is [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], which means that the label is 0.

It is essential to decide the layer of the neural network. To decide the layer, we start with a single layer neural network with ten neurons because the classes is ten. The transfer function is Softmax function, which make the output 10 probability [4].

$$a_i = f(n_i) = \exp(n_i) \div \sum_{j=1}^{S} \exp(n_j).$$

Figure 4. Equation of Softmax Function

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} a_1^m\left(\sum_{i=1}^{S^m} a_i^m - a_1^m\right) & -a_1^m a_2^m & \cdots & -a_1^m a_{S_m}^m \\ -a_2^m a_1^m & a_2^m\left(\sum_{i=1}^{S^m} a_i^m - a_2^m\right) & \cdots & -a_2^m a_{S_m}^m \\ \vdots & \vdots & & \vdots \\ -a_{S_m}^m a_1^m & -a_{S_m}^m a_2^m & \cdots & a_{S_m}^m\left(\sum_{i=1}^{S^m} a_i^m - a_{S_m}^m\right) \end{bmatrix}$$

Figure 5. Derivative of Softmax Function

Hence, we choose the cross entropy loss function to calculate the similarities between the prediction result and the target. KL-Divergence represent the similarities between two distribution. If the entropy fix, the cross entropy will be change with the KL-Divergence. Hence, the cross entropy represents the KL-Divergence. The reason why we pick cross entropy as the loss function is that Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. In Figure 6, y represents the target and s represent the prediction

$$e = crossEntropy(s, y) = -\sum_{i=1}^{k} y_i log(s_i)$$

$$\nabla e_{(s)} = (-\frac{y_1}{s_1}, -\frac{y_2}{s_2}, \cdots, -\frac{y_k}{s_k})$$

Figure 6. Derivative of Cross Entropy

Here is the summary of the required information for the neural network:

Problem Type: classification

Classes: 10

Input format: 1X 784 vector

Batch Size: 100 input

Output format: 1X 10 vector(Outcome probability distribution)

Target format: 1X 10 vector(One hot encoding)

Cost function: Cross entropy

Then, we use the backpropagation and the gradient descent to train this neural network. The accuracy of this neural network is 91%. The loss function is plotted in Figure 7.

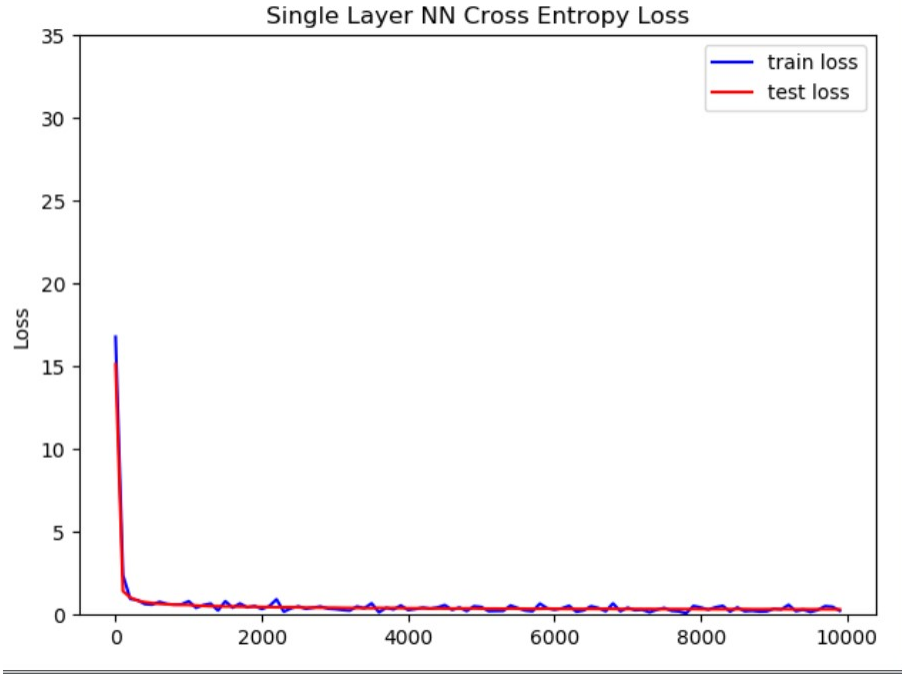

Figure 7. Single Layer NN Cross Entropy Loss

From Figure7, It can be conclude that the numbers of layer and neuron is not enough because there is no overfitting in the diagram and the 91% accuracy is not enough. Hence, we decide to add more layer and neurons to recognize there images more accurately. We build 4 layers neural

network. All weights are initialized among -0.2 to 0.2 and all bias are initialized with 0.1. The layer structures are:

1.First layer consists of 200 neurons. Hence, the weight is a 784 by 200 matrix, the bias is a 1 by 200 matrix. The transfer function is Relu function

2.Second layer consists of 100 neurons. Hence, the weight is a 200 by 100 matrix, the bias is a 1 by 100 matrix. The transfer function is Relu function

3.Third layer consists of 50 neurons. Hence, the weight is a 100 by 50 matrix, the bias is a 1 by 50 matrix. The transfer function is Relu function

4.The fourth layer consists of 50 neurons. Hence, the weight is a 50 by 10 matrix, the bias is a 1 by 10 matrix. The transfer function is Softmax function

From Figure 8, It can be seen that there are many 'sharp mountains' which means that the gradient step is to large to find the minimum point. Hence, we should use the learning rate decay to make the curve smooth. The learning decay rate is 1/exp [5]. On the other hand, the cost function goes up during the last iterations. Therefore, we need to drop out some neurons to solve this problem [6]. Hence, we choose the drop out rate 0.2, and the decay function: 0.0001 + 0.003 * (1/e)^(step/2000)).

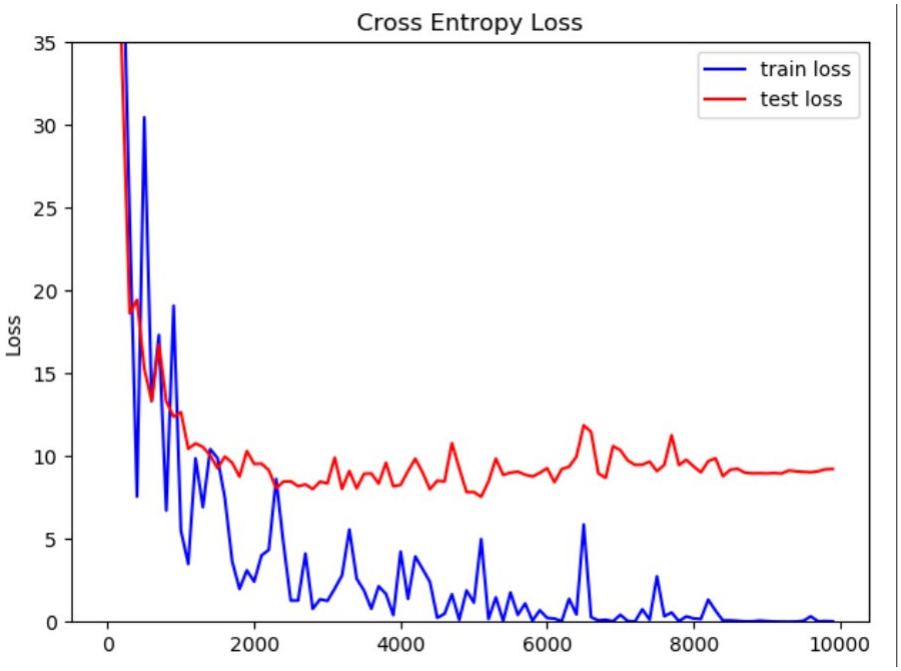

Figure 8. 4 layers NN Cross Entropy Loss

After dropping out and learning rate decay, the result is shown in Figure 9. There is no overfitting and the curve is smooth. However, the accuracy is still 98% which does not change. Although we construct five layers with more neurons, the accuracy still maintain in 98% level. The reason is that we flatten the image into a vector. We only consider the relationship in rows, not both in rows and columns. In the other word, the shape feature of the image is ignored. If we

want to improve the accuracy, we should utilize the convolution neural network structure to train these data.

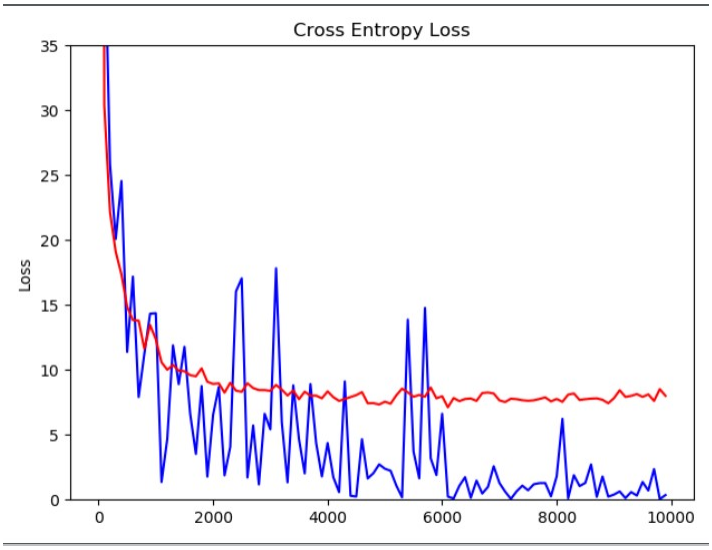

Figure 9. Modified 4 layers NN Cross Entropy Loss

[1] "THE MNIST DATABASE," *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 20-Aug-2019].

[2] R. Vaidya, D. Trivedi, S. Satra, and P. M. Pimpale, "Handwritten Character Recognition Using Deep-Learning," *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018.

[3] "The One Hot Technique in Finite-State Machine Design," *FSM-Based Digital Design Using Verilog HDL*, pp. 105–143, Jan. 2008.

[4]M. T. Hagan, H. B. Demuth, M. H. Beale, and Jesús Orlando De, *Neural network design*. S. l.: s. n., 2016.

[5] "Google Cloud Platform," *GitHub*. [Online]. Available: https://github.com/GoogleCloudPlatform. [Accessed: 20-Aug-2019].

[6] "Ghosh4AI - Overview," *GitHub*. [Online]. Available: https://github.com/Ghosh4AI. [Accessed: 20-Aug-2019].