Project Report

Handwritten Digit Recognition Using MNIST Dataset

Karthikeya Bolla

Abstract

Demonstrating the capability of deep learning models using image datasets has been a practice over the years. *Object Recognition*, the task of identifying objects in digital images and/or videos, is one of the fundamental problems in the areas of Computer Vision and Deep Learning. Among the diverse problem statements in object recognition, recognizing hand-written digits is a benchmark problem. The project emphasizes on discussing various techniques and their efficiencies employed to classify hand-written digits.

1 Introduction

The goal of **Artificial Intelligence** since its birth in 1956 [3] has been to create technological advancements in computers and machines to function in an intelligent manner. For instance, a computer programmer intends to detect cars in photographs. To perform the same, it is necessary that the programmer be able to program the features of a car viz. body shape, wheels and steering. Although it sounds appealing, programming these features as pixels is tough and can vary across different problem scenarios viz. different cars, image distortions, lighting and contrast effects. One effective solution to this problem is to use **representation learning** which learns representations of features and maps it to output. These allow artificial intelligent systems to learn and adapt to new tasks. **Deep learning** allows for representation learning by building representations that are expressed in terms of simpler representations. Mapping a feature from set of pixels to an object is complicated and deep learning solves this problem by fragmenting complicated mapping to series of simple mappings performed along *layers* of the model [4]. In the area of machine learning advancements, deep learning marks a new era with an objective of moving machine learning closer to artificial intelligence [1]. ***Deep learning*** *is a class of machine learning algorithms that use a cascade of many layers of nonlinear processing units for feature extraction and transformation* [2].

The scope of this project is to classify images, images of hand-written digits. **Image Classification**, one of the classic problems of Computer Vision, is the task of assigning an input image a label from a fixed set of labels. In the project, class labels are numbers from 0 - 9. To solve this problem of hand-written digit classification, data-driven approach is being used. A

simple algorithm cannot be written to classify images pertaining to digits "1", "2" and so on, unlike writing an algorithm to sort list of numbers. In the data-driven approach, the computer is provided with many examples of each class and a learning algorithm is developed to *learn* through these examples and map it to necessary class labels. For the process of learning, training dataset, which is a collection of input data with class labels is provided to the learning model. The model maps features from training data to class labels and hence learning is done. To test the goodness of trained model, a test set with inputs but no class labels is provided to the model. After applying the model on test set, outputs are recorded and standard evaluation metrics are applied to measure model performance.

Specific contributions of this project include (1) implementing backpropagation algorithm as matrix multipliers (2) mimicking the work of LeCun et. al. [10] and achieving better performance by applying combination of different techniques in neural networks developed over the years

2 Description

2.1 Artificial Neural Networks

Artificial Neural Networks are computation models that are based on collection of connected simple units called *artificial neurons* which loosely mimic the behavior of biological neurons. Activation signals of varying strength are passed among neurons which activate other connected neurons. These activations and effectiveness of these activations help in learning. Neurons are connected in layers. For any *neural network*, group of neurons form a layer and collection of layers constitute a neural network. Typically, there is an *input* and *output* layer as the first and last layer for any neural network. Modern ANNs have multiple (>= 1) *hidden* layers with each layer having hundred to thousands to millions of neurons [7]. Connections among neurons are modeled as acyclic graphs. For a regular neural network, neurons from adjacent layer are fully pairwise connected (Figure 1) to neurons of other layer and this behavior is observed among all adjacent layers in the network [8]. The mathematics involved in modeling neurons as computational units is best described in [4, 8].

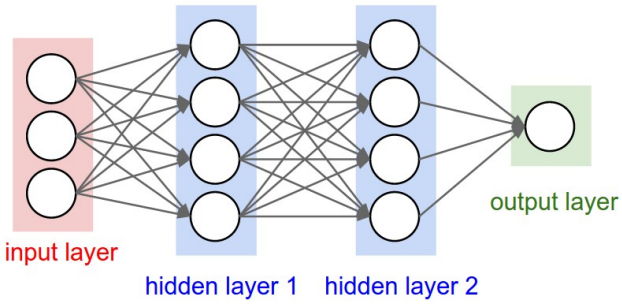

Figure 1

2.2 Dataset

MNIST (Modified NIST) dataset of handwritten digits has 60,000 training examples and 10,000 test examples. It is a subset of larger dataset available from **NIST** (National Institute of Standards and Technology). The original images from NIST were size-normalized into 20 X 20 pixel box. These images were size-normalized and centered in 28 X 28 pixel box (Figure 2) for MNIST dataset. This pre-processing is done for easier use with machine learning algorithms (LeCun et. al.) [5].

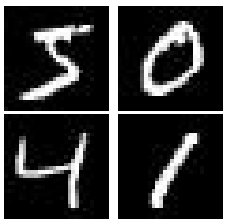

Figure 2

3 Related Work

MNIST hand-written digit recognition has achieved "near-human performance" over the past few years with neural networks as classifiers. Below table (Table - 0) summarizes some of the notable results achieved [5, 15, 16, 17]

Method	Test Error Rate
Regularization of Neural Networks using DropConnect Wan, Li, et al. "Regularization of neural networks using dropconnect." Proceedings of the 30th International Conference on Machine Learning (ICML-13). 2013.	0.21%
Multi-column Deep Neural Networks for Image Classication Ciregan, Dan, Ueli Meier, and Jrgen Schmidhuber. "Multi-column deep neural networks for image classification." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.	0.23%
Batch-normalized Maxout Network in Network Chang, Jia-Ren, and Yong-Sheng Chen. "Batch-normalized maxout network in network." arXiv preprint arXiv:1511.02583 (2015).	0.24%
Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree Lee, Chen-Yu, Patrick W. Gallagher, and Zhuowen Tu. "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree." International conference on artificial intelligence and statistics. 2016.	0.29%
Recurrent Convolutional Neural Network for Object Recognition Liang, Ming, and Xiaolin Hu. "Recurrent convolutional neural network for object recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.	0.31%
Training Very Deep Networks Srivastava, Rupesh K., Klaus Greff, and Jrgen Schmidhuber. "Training very deep networks." Advances in neural information processing systems. 2015.	0.45%
PCANet: A Simple Deep Learning Baseline for Image Classification? Chan, Tsung-Han, et al. "Pcanet: A simple deep learning baseline for image classification?." IEEE Transactions on Image Processing 24.12 (2015): 5017-5032.	0.62%
Deep Boltzmann Machines Salakhutdinov, Ruslan, and Geoffrey Hinton. "Deep boltzmann machines." Artificial Intelligence and Statistics. 2009.	0.95%

Table - 0

4 Architecture

Trends in machine learning methods have been applied to recognize objects in images. To learn digits form thousands of images and to classify the same, models with significant training capacity are needed. **Artificial Neural networks (ANNs)** are one such model. Also, the complexity of these object recognition tasks demand that the model should have prior knowledge about data [14]. **Convolutional Neural Networks (CNNs)** belong to this category. The implementation described in this paper focuses on Artificial Neural Networks.

The model, as part of the project, has been implemented from scratch without using pre-implemented machine learning packages viz. Scikit-learn,

Weka, R. The programming language used is Python 2.7. Describing the model, it is a 2-layer neural network viz. it has 1 hidden layer and 1 output layer (and hence the name 2-layer NN) along with an input layer. Training images available as binary input file are read. Every individual image is a 28 X 28 binary representation. To convert binary data into integer values, `idx2numpy` function [9], an open source python package, has been used. After reading through the training data (X_{input}), which is a 60000 X 28 X 28 matrix, it has been compressed to 60000 X 784 matrix where records indicate training images and columns represent image pixels. The number of neurons used in input layer (#input) are 784, which represent number of pixels in an image. Experiments are run using 800 and 1000 neurons in hidden layer (#hidden), mimicking the work of LeCun et. al. [10] and Simard et. al. [11]. Also, digits in images have to be classified into one of 10 categories viz. 0 - 9, hence there are 10 neurons in output layer (#output). The network is fully connected. Weights flowing from input layer to hidden layer ($W_{input,hidden}$) are initialized as [8, 12]

$$W_{input,hidden} = \frac{rand(\#input,\#hidden)}{\sqrt{\frac{2}{\#input}}}$$

Therefore $W_{input,hidden}$ is a matrix of dimensions (784, 800) and (784, 1000). Also, bias $b_{input,hidden}$ is initialized to all zeros whose dimensions are (1, 784). Weights flowing from hidden layer to output layer $W_{hidden,output}$ are initialized as small random values. The dimensions of $W_{hidden,output}$ is (1000, 10). Bias $b_{hidden,output}$ is again initialized to all zeros. To choose better weights **L2 regularization** is used. Upon experimenting, hyper-parameters viz. learning rate and regularization rates are set to 0.09 and 0.001. Initially implementation of the algorithm using batch gradient descent over 30, 50 and 100 epochs was computationally not feasible, although matrix level multiplication was used. Therefore, for faster computation, **mini-batch gradient descent** [13] is used with 1800 mini-batches, every batch constituting 30 records. The number of epochs is set to 10. These epochs are implemented as cross-validation for effective results.

During forward propagation of weights, in the hidden layer, **ReLU (Rectified Linear Unit)** *f(x) = max(0,x)* is used as activation function [14]. Using ReLU as activation function removes the problem of saturation of gradients [8]. In the output layer, **Softmax function** $f(x) = \frac{e^{x_i}}{\Sigma_j e^{x_j}}$ is used as activation function. To calculate total loss of the model, **Cross-entropy** loss and regularization is computed. Gradients are backpropagated along

the network using **backpropagation** algorithm. Technicalities involved in the computation of gradients and propagating them back through the network can be found at [8, 14]. The project implements backpropagation using matrix multiplication. Gradient of output layer $D^{(output)}$ is computed first. Using $D^{(output)}$, gradient of hidden layer $D^{(hidden)}$ is computed as

$$D^{(hidden)} = F'^{(hidden)} \odot D^{(output)}(W_{hidden,output})^T$$

Using $D^{(hidden)}$, gradient of input layer $D^{(input)}$ is computed as

$$D^{(input)} = F'^{(input)} \odot D^{(hidden)}(W_{input,hidden})^T$$

For the data flowing from layer i to j, $D^{(i)}$ represents gradient of i^{th} layer. $F'^{(i)}$ represents derivative of activation function used in i^{th} layer. $D^{(j)}$ represents gradient of j^{th} layer. $(W_{i,j})^T$ represents weights flowing from i^{th} layer to j^{th} layer. $\odot$ represents element-wise matrix multiplication.

Weight updates are performed as

$$W_{hidden,ouput} = W_{hidden,output} - \{ \eta \ Z^{(hidden)^T} D^{(output)} + \lambda (W_{hidden,output})^T \}$$
$$W_{input,hidden} = W_{input,hidden} - \{ \eta \ Z^{(input)^T} D^{(hidden)} + \lambda (W_{input,hidden})^T \}$$

η, λ being learning and regularization rate, $Z^{(input)}$ and $Z^{(hidden)}$ being outputs from input and hidden layers.

5 Results

The model implemented is derived from LeCun et. al. and Simard et. al.[10, 11]. Therefore it is important to first quote the results of LeCun et. al. and Simard et. al. [5], shown in Table - 1

Classifier	Preprocessing	Test Error Rate (%)	Reference
2-layer NN, 300 HU, MSE	none	4.7	LeCun et.al 1998
2-layer NN, 300 HU, MSE [distortions]	none	3.6	LeCun et.al 1998
2-layer NN, 300 HU	deskewing	1.6	LeCun et.al 1998
2-layer NN, 1000 HU	none	4.5	LeCun et.al 1998
2-layer NN, 1000 HU [distortions]	none	3.8	LeCun et.al 1998
2-layer NN, 800 HU, Cross-Entropy Loss	none	1.6	Simard et.al ICDAR 2003
2-layer NN, 800 HU, Cross-Entropy [affine distortions]	none	1.1	Simard et.al ICDAR 2003
2-layer NN, 800 HU, MSE [elastic distortions]	none	0.9	Simard et.al ICDAR 2003
2-layer NN, 800 HU, Cross-Entropy [elastic distortions]	none	0.7	Simard et.al ICDAR 2003

Table - 1

Neural network developed as part of project uses 800 and 1000 hidden neurons. The results are as follows

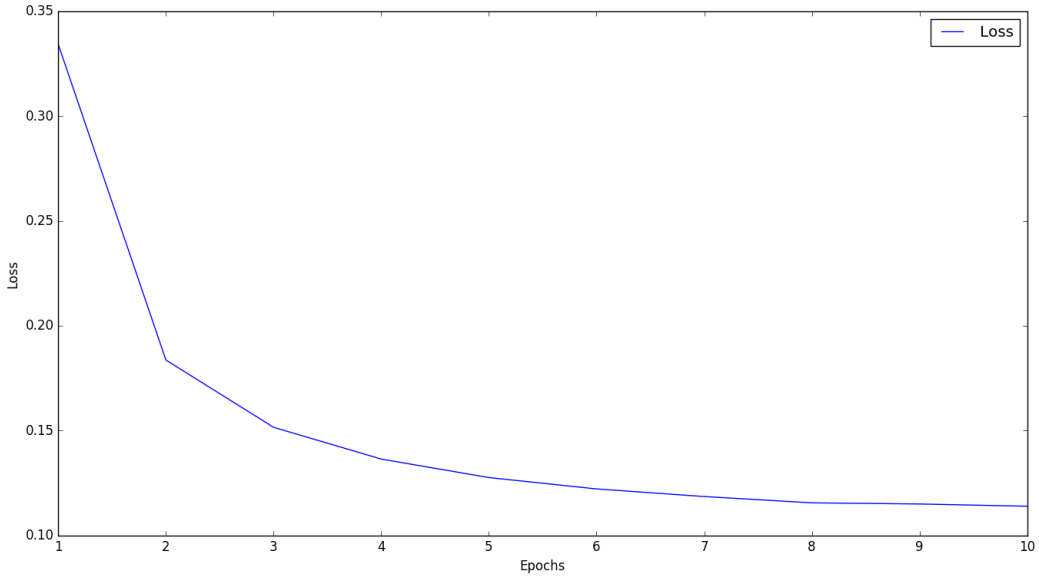

Figure 3: Loss vs. Epochs, 800 hidden units, learning rate = 0.09

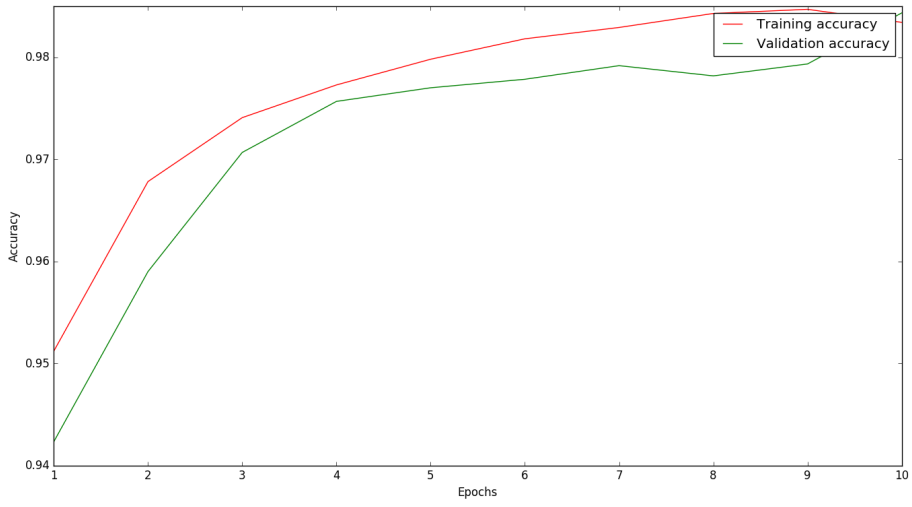

Figure 4: Accuracy vs. Epochs, 800 hidden units, learning rate = 0.09

Curve in Figure-3 shows the decrease in loss value with an increase in epoch value, which indicates positive learning capability of model. Figure 4 shows the curve of training vs. validation set accuracies which indicates that model does not overfit because validation accuracy curve trails training accuracy curve. Model parameters representing Figure 3, 4 are #hidden neurons = 800, learning rate = 0.09, epochs = 10, mini-batch size = 30.

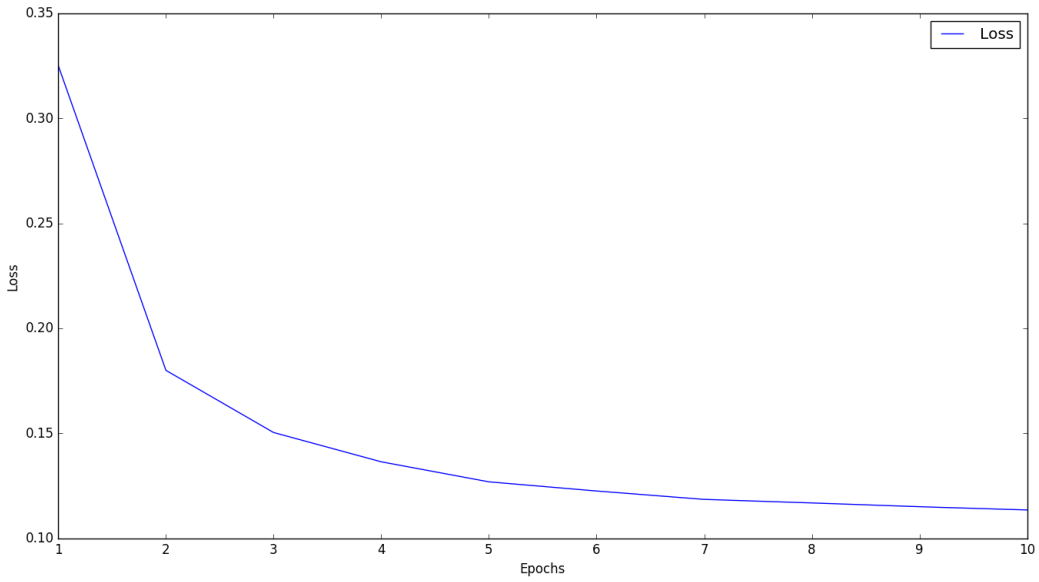

Figure 5: Loss vs. Epochs, 1000 hidden units, learning rate = 0.09

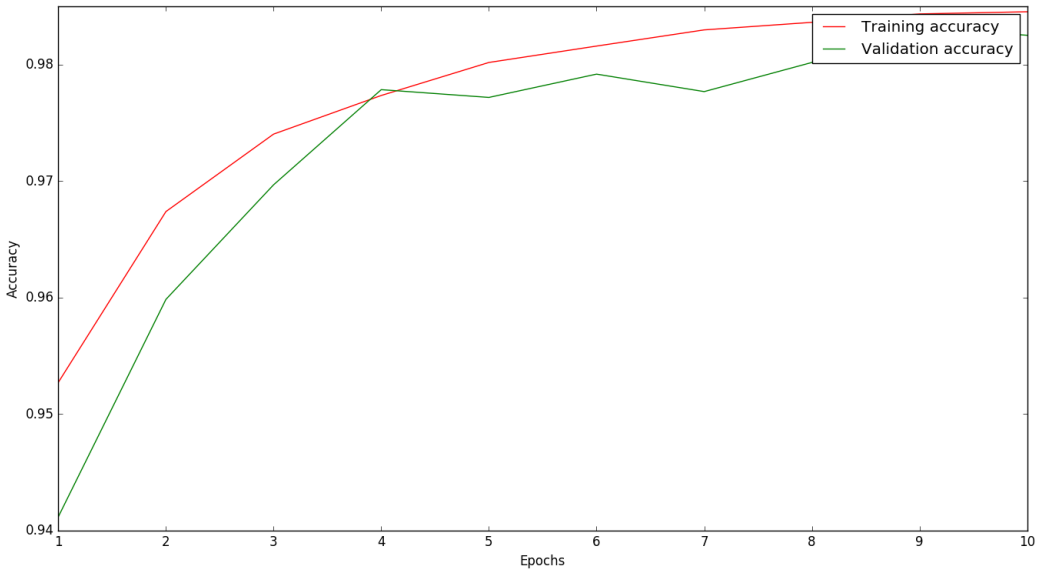

Figure 6: Accuracy vs. Epochs, 1000 hidden units, learning rate = 0.09

Figure 5 & 6 exhibit similar behavior as 3 & 4. Model parameters representing Figure 5, 6 are #hidden neurons = 1000, learning rate = 0.09, epochs = 10, mini-batch size = 30. Running the model with #hidden neurons = 800, 1000, learning rate = 0.09, epochs = 10, mini-batch size = 30 on test set produced the following test error rate

Classifier	Test Error Rate(%)
2-layer NN, 800 HU, ReLU, Cross-entropy, mini-batch GD	2.43
2-layer NN, 1000 HU, ReLU, Cross-entropy, mini-batch GD	2.31

Table - 2

From Table - 1 & 2 we can observe that the model implemented as part of project performs better than the model used by LeCun et.al. The increase in performance can be attributed to use of ReLU activation and initialization of weights as suggested by He et. al. [12]. Also, using mini-batch gradient descent helped in effective learning of weights. The model developed underperforms Simard et. al. The difference in performance can be attributed to distortions in the model proposed by Simard et. al.

6 Conclusion

The paper proposes simple 2-layer neural network model with ReLU activation units, mini-batch gradient descent with varying number of hidden neurons. Test error rate achieved is better than models used in [5]. But, recent advancements in neural networks viz. Convolutional Neural Networks (CNNs) have achieved state of the art performance (0.2% test error rate). Thus there is scope for model improvement. Other techniques that need to be incorporated and tested on the model are parameter updates viz. momentum method, Nestrov momentum and RMSProp. Also, size of layers needs investigation.

7 References

[1] http://deeplearning.net/

[2] https://en.wikipedia.org/wiki/Deep_learning

[3] https://en.wikipedia.org/wiki/Artificial_intelligence#History

[4] http://www.deeplearningbook.org/

[5] http://yann.lecun.com/exdb/mnist/

[6] http://neuralnetworksanddeeplearning.com/chap2.html

[7] https://en.wikipedia.org/wiki/Artificial_neural_network

[8] http://cs231n.github.io/

[9] https://pypi.python.org/pypi/idx2numpy

[10] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

[11] Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis." ICDAR. Vol. 3. 2003.

[12] He, Kaiming, et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." Proceedings of the IEEE international conference on computer vision. 2015.

[13] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

[14] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[15] http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#4d4e495354

[16] https://en.wikipedia.org/wiki/MNIST_database#Classifiers

[17] http://people.idsia.ch/~ciresan/results.htm