Deep Learning and Sensitivity with MNIST Dataset

Ankur Vazirani

The Australian National University

U6483857@anu.edu.au

Abstract

In this paper, I have used the MNIST (Modified National Institute of Standards and Technology database) database to get the idea of how computers can recognize handwritten digits. In this project, an attempt is made to do further research on the working of the deep neural network and extend my work from previous assignment by applying sensitivity a pruning technique. In this paper, a comparison has also been made to check the performance of different classifier algorithms. Consideration is made for various hyperparameters, and the convolutional neural network by taking account of the image length and width to make a stride. The network also comprises pooling layers, and max-pooling is applied to the network. The final results are compared with the research paper of "An Exploratory Study on MNIST Dataset" which was published in University of California, San Diego. [1]

Keywords: Visualisation, pooling, convolutional neural network, sensitivity.

1. Introduction

The growth of deep learning in the field of artificial intelligence has been astounding in the last decade with about 35,800 research papers being published since 2016. [2] With this much amount of research it has been tough to keep up with it for many research organizations and practitioners.

There is much research which has been made in the field of pattern recognition in recent years. Some astonishing results have also been achieved in the area of handwritten recognition. This rapid progress has resulted from a combination of many developments of powerful hardware, and due to the introduction of new algorithms.

In this research, I have extended my previous assignment work by applying sensitivity a pruning technique to a deep learning model for MNIST dataset with backpropagation. I have also looked at accuracy, and I have proposed a visualization of how data is getting trained in the final layer with that a comparison is also made with different machine learning algorithms. The final results are also compared with the previous research paper in which the technique of sensitivity was applied with one hidden layer.

A technique of pooling has also been applied to this dataset. The idea is to make the maximum pool that occurs in each convolution layer with a max value of 2 x 2 area. The objective of pooling was to sample input representation, with reducing the dimensions and allowing some assumptions to be made for features contained in the subregions behind.

2. Dataset

The MNIST database consists of an enormous amount of data in handwritten digits. The original dataset from NIST is comprised of 60,000 examples and has a test set of 10,000 examples. This dataset is a subset of one of the more massive sets from NIST. The digits sizes have been normalized and centred in a fixed size of an image.

The original dataset of NIST was of black and white images, which were normalized to fit in a 20 x 20-pixel box while also preserving the aspect ratio. The resulting dataset contains images with grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28 x 28 image by computing the center of mass of pixels and translating the image as to position this point at the center of the 28 x 28 field.

The MNIST database was constructed from NIST Special database 3 and Special database -1 which has binary images of handwritten digits. The MNIST training set has 30,000 patterns from SD-3 and 30,000 from SD-1, with that the test set has 5000 patterns from SD-3 and the next 5000 from SD-1. The training set contained examples from approximately 250 writers, and 60,000 patterns were drawn from this training set. The creators of this dataset also made sure that both training and test set are disjoint. [3]

3. Deep Neural Network

The network comprises 2 convolutional layers. The first hidden layer takes the input shape of (1, 28, 28) and the second layer takes the input shape of (16, 14, 14). The first layer also adds a filter of size 5 and to keep the length and width of this image as it is, I have added a stride of 1. The first convolutional layer gives us the output shape of (16, 28, 28) and the second layer gives an output shape of (32 x 7 x 7 x 10) which gives me a fully connected architecture with 10 output neurons, with this I have added Relu as an activation function in both the layers.

I have used Adam (Adaptive Moment Estimation) as an optimiser. Adam is an updated version of RMSProp optimizer. It is the algorithm which runs averages of both the gradients and the second moments of the gradient are used.

3.1 Sensitivity

The sensitivity analysis of neural network can be defined by the contribution of a hidden neuron to its output layer. "The information used in approximating the sensitivity uses terms which are often available during training with back-propagation, such as the weight increments, and the partial derivatives of the error surface."[5]

We can achieve sensitivity in multiple ways. During the implementation of this network he has implemented sensitivity by calculating the output of hidden neurons to the output layer, and if any hidden neuron is achieving weight less than 0.1 in more than one epochs, and not contributing to the output layer, then the weights are turned to zero.

This will lead to zero contribution of that particular neuron in the network. Sensitivity helps in network reduction and also reduces the complexity of the network.

The loss can be generalize in a neural network by performing nonlinear differentiable mapping: I → K, from input x = (x1, x2... xI) to output o = (o1, o2... oK). Suppose x (n) ∈ Δ, where Δ is an open set. Since o is differentiable at x (n) we have o(x + Δ x) = o(x(n)) + J(x(n)) Δ x + g(Δ x), where J(x(n)) is the Jacobson matrix.

There are multiple purposes of using sensitivity analysis in neural networks (Engelbrecht, 1999):

Optimization: "In neural networks, derivatives of the objective function concerning the weights are computed to locate minima by driving these derivatives to 0. Second order derivatives have also been used to develop more sophisticated optimization techniques to improve convergence and accuracy. Koda (1995, 1997) employed stochastic sensitivity analysis to compute the gradient for time-dependent networks such as recurrent neural networks."

Selective Learning: Hunt and Deller in 1995 used weights to determine its impact on each pattern when the data changes while training the dataset. They used a highly effective method in which they had chosen patterns which had a maximum effect on weight changes. [6]

3.2 Pooling layers

Pooling layers are usually described immediately after the convolutional layer. A pooling layer takes each feature map and outputs it from the convolutional layer and prepared a condensed feature map. During this research, I have applied a technique called max-pooling. The pooling region is set to 2 x 2, and a pooling unit outputs the maximum activation in this region.

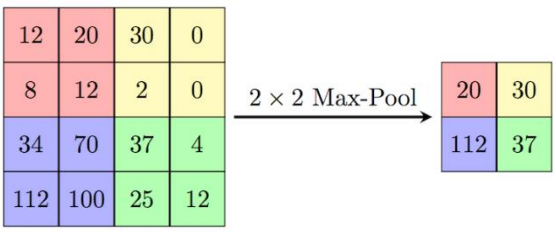

Fig 1. Pooling image. [4]

I have applied three feature maps separately and then I have combined them for max pooling with the final output architecture of the network. This gives us 10 possible values from MNIST dataset with digits (“0”, ‘’1”, ‘’2” ... ”9”).

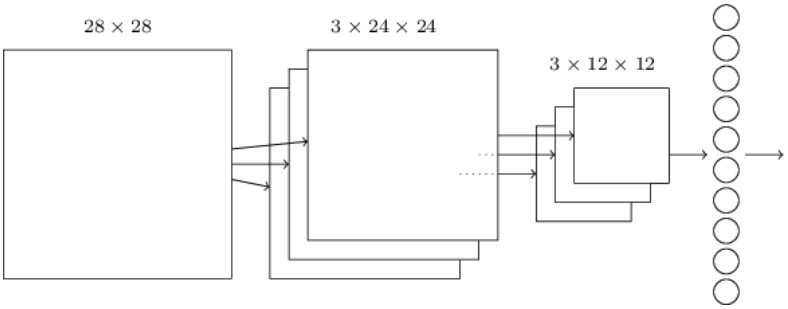

Fig 2. Final output image of the network

4. Results

When the network is getting trained at last layer it produces results something like in the Fig 3.

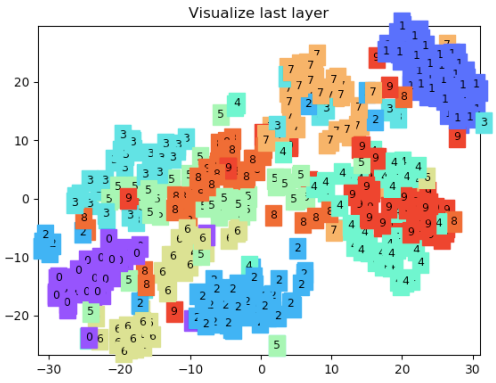

Fig 3. Visualization of data in the last layer

The final accuracy of the network was 98% after applying sensitivity to the data, whereas compared to the network before sensitivity the accuracy was of 97.15%. It seems to be a good result, but the comparison of the results with research paper mentioned above shows that when the network gets trained with the combination of softmax regression and linear SVM, it produces an accuracy of 91%. However, the best accuracy one can get is of 99.17%, and this result is obtained when one trains network with SVM and KNN algorithm.

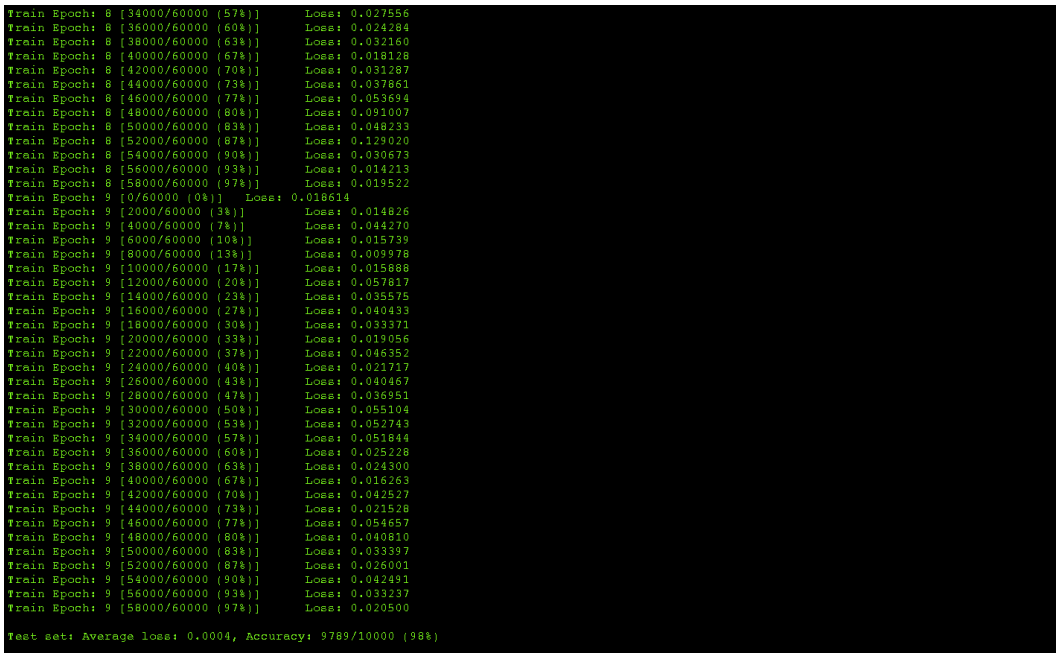

Fig 4. Final output

However, it took some time to train the data because the network wasn't made compellingly. The CNN is most effective and takes the least amount of time when it takes with Softmax regression as compared with KNN. The short timing is also one of the advantages of using softmax model.

The model in the first epoch starts with 3%, and it increases with 4 % on every iteration. By the time it reaches the last iteration of the training process, the accuracy hits its highest point of 97%. Finally, in the testing set, the network produces an accuracy of 98%.

5. Conclusion

This paper has produced an excellent score, and it also shows that sensitivity is essential to train the data, even though after applying sensitivity the result obtained made a minute difference, but these differences can make much difference when applied in the real world application.

The paper also focuses on different techniques and algorithms which can be used to improve the efficiency of the network. I found that network would give a better result if used with SVM with the combination of KNN (K Nearest Neighbour).

The paper also shows that max pooling is just not enough to get the results to improve the performance of the network we need to use better choices of pooling such as stochastic pooling to get the best results.

It also shows that when plenty of data is available then only it makes sense to apply deep neural network. Deep network is applied because of the complexity and to give better results.

There are also certain methods which can only be applied in deep neural network which also helps us in improving performance. These techniques has also saved a lot of time and it reduces the time complexity.

6. References

1. B. Zhang, M. Yin, X. Lin and Z. Zhu, *An Exploratory Study on MNIST Dataset*.
2. LeCun, Y., Cortes, C. and J.C. Burges, C. (n.d.). *THE MNIST DATABASE of handwritten digits*. [online] http://yann.lecun.com. Available at: http://yann.lecun.com.
3. Show and Tell (Vinyals et al. 2015): Original implementation available in Theano; https://github.com/ kelvinxu/arctic-captions.
4. *https://computersciencewiki.org*. [Online]. Available: https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.
5. T. Gedeon and D. Harris, Network Reduction Techniques, 1st ed. Brunel University, p 3.
6. D. Yeung, Principles of Sensitivity Analysis, 8th ed. Springer, 2010.