Effect Of Injected Noise In Deep Neural Networks

Naresh Nagabushan
Department of ECE
M.S. Ramiah Institute of Technology
Bangalore, India
Email: rnaresh.n@gmail.com

Nishank Satish
Department of ECE
M.S. Ramiah Institute of Technology
Bangalore, India
Email: nishaldo@gmail.com

Raghuram S
Department of ECE
M.S. Ramiah Institute of Technology
Bangalore, India
Email: raghuram@msrit.edu

Abstract—**Deep Neural Networks have become increasingly popular due to their efficient realization in GPU hardware. Problems that were once considered computationally intensive to implement using Neural networks have now become possible due to the vast amount of flexibility and capability offered by the GPU and Deep networks combination. In this work, we attempt to improve the recognition rate for images, using Deep Neural Networks for the classification task. The Artificial Neural Network is modeled on the biological Neural network, where many thousands of neurons and dendrites act synchronously to perform the recognition task. One aspect of this network is the inherent noise in the signals, due to the physical proximity of electrical connections, and hence through inductive and capacitive coupling. In this work, we have used this as a motivation to add a noise component to the values through the network, thereby to create a more realistic model of the biological neural network. We have constructed a neural network architecture in which the output of neurons in each layer is affected by the value of the signal in the neighboring neurons. This method has the predicted effect of reducing overfitting and further increases the Recognition Rate for Deep Networks by up to 8 percent.**

Keywords—Noise; Recognition rate; Neural networks; Deep neural networks.

I. INTRODUCTION

Neural networks can be used to recognize images by taking a large number of training examples each of which has a label assigned to it, the network then uses these examples to modify weights and biases to infer rules for recognizing untrained images. This type of learning that has both the input pattern and the desired target is known as supervised learning. The modification of weights and biases in such a method is carried out using a function known as the cost function. So the main aim of network will be to minimize this cost function. This can be done using an algorithm known as *gradient descent.* The cost function that can be used to improve the learning slowdown is the *cross-entropy* [1] function given in (1):

$$C = -\frac{1}{n}\sum_x [y \ln a + (1-y)\ln(1-a)] \qquad (1)$$

Where n is the total number of training data, x is the training input, y is the desired output and a is the output of the neuron.

The modification of weights and biases for the kth neuron takes place according to (2) and (3) respectively:

$$w_k^{|} = w_k - \eta \frac{\partial C}{\partial w_k} \qquad (2)$$

$$b_l^{|} = b_l - \eta \frac{\partial C}{\partial b_l} \qquad (3)$$

where η is the learning rate, C is the cost, l is the layer, w is the weight and b is the bias.

Unfortunately, since the computation of the gradients ΔC requires the compute of gradients ΔC_X separately for each training example, X, this process can consume a large amount of time when the number of training inputs are large, this then can be overcome by using the stochastic gradient descent [2] method which takes the training images in batches to modify the weights and biases. The modified versions of (2) and (3) using the stochastic gradient descent method is given in (4) and (5):

$$w_k^{|} = w_k - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial w_k} \qquad (4)$$

$$b_l^{|} = b_l - \frac{\eta}{m}\sum_j \frac{\partial C_{X_j}}{\partial b_l} \qquad (5)$$

Where m is the mini-batch size.

The gradient of the cost function can be computed using the backpropagation [3] algorithm. Which is not only a fast algorithm to compute the gradients but also gives an insight as to how the change in weights and biases changes the overall behavior of the network. The output error δ^L, back propagating of error and the gradients can be calculated using the (6), (7), (8) and (9) respectively,

$$\delta^L = \nabla_a C \odot \sigma^{|}(Z^L) \qquad (6)$$

Where L is the layer, $\odot$ is the Hadamard product [4] and $\sigma^{|}$ is the derivative of the sigmoidal activation function used.

$$\delta^L = ((w^{l+1})\delta^{l+1}) \odot \sigma^{|}(Z^l) \qquad (7)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \qquad (8)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{9}$$

Where l = L−1, L−2,…2, w_{jk} is the weight between the kth neuron in the lth layer and the jth neuron in the l−1th layer. Equation (7) is used to find the error at the layer l if the value of the error $\delta(l + 1)$ at layer $l + 1$ is known. Equations (8) and (9) are used to find the value of the gradients if the output error δ^l at the layer l is known.

Since Overfitting [5] is a major issue especially in modern networks that have a very large number of weights and biases, we would want to use techniques that reduce this effect. Increasing the training examples is one such method but it could be difficult and expensive to obtain more training images, we can use other techniques such as *weight decay* also known as *L2 regularization* [6] given by (10):

$$C = -\frac{1}{n}\sum_{x}[y_j \ln a_j^l + (1 - y_j)\ln(1 - a_j^l)] + \frac{\lambda}{2n}\sum_{w} w^2 \tag{10}$$

Where $\lambda > 0$ is known as the regularization parameter and n is the size of the training set.

II. Recent Work

Studies on Artificial Neural Networks have been going on since the 1940's [7] which also developed into very complex architectures. Recently, Artificial Neural Networks with more than 3 hidden layers i.e., Deep Neural Networks have been designed. The first basic idea of a deep neural network was of a *deep multilayer perceptron* [8].

The addition of gradient noise into very Deep Neural Networks was not only found to be a low overhead, easy to implement technique but also was found to have a 72% lesser relative error rate when applied to a very deep network [9]. In this method, a time dependent Gaussian noise was added to the gradient at every training step. From the past two decades, a lot of studies have involved addition of noise into the neural network to improve the efficiency. But the implementation of coupling noise in neural networks has not been ventured into.

Training Neural Networks with additive noise was proposed [10] which required a random additive noise to be added to a desired signal rather than adding random noise to weights which was proposed previously. This technique was found to speed up the backpropagation algorithm which in turn speeds up the supervised learning. This noise is not because of coupling. Therefore, this technique is very different from our proposed technique in which tapered percentage of noise is added due to neighboring neurons.

A calculated noise added to a Convoluted Neural Network [11] was found to enhance the speed of convergence of the backpropagation algorithm which is a special case of the generalized expectation-maximization (EM) algorithm. This method finds the special noise only above the hyperplane in the network's noise space. In this method, the noise added is a certain percentage of the network's noise.

With the advent of very deep neural networks, one problem is the computation speed. This might be a complex task with a CPU but recently, GPU's have been used effectively to train the network in lesser amount of time [12]. Most of the very deep networks being used require GPU's for faster and efficient computation.

III. Proposed Method

Crosstalk is a common phenomenon in high speed VLSI designs - closely placed wires carrying electrical signals are coupled capacitively through the substrate. Hence, a change in one signal has an effect on its immediate neighbors. Similarly, in the biological neural network, more than ten thousand neurons are connected to each other using approximately 10^9 axons. This too will couple and affect neighboring signals whenever transitions occur. Since the evolution of the biological neural network has taken into account this coupling, intuitively, this noise factor should have an effect of improving the recognition rate. We are hence, artificially introducing this noise into our computations, in ways described in the rest of this section.

We have used three simple techniques to add noise to the output of each neuron present in the deep fully connected network. The first one being the addition of a tapered percentage of the input given to each neuron present in its neighborhood which we call the *area of influence noise* (AOI noise), second being the same as the first with randomness considered and the third a small Gaussian noise added to the output of each neuron in the network which can be interpreted as propagation noise which we call *line noise*. These methods where found to be astonishingly fruitful when applied for the image classification for data sets with small number of training and testing images. It was found that these methods coax exploration in the parameter space which was found to be helpful in deep networks.

A. The Architecture:

The architecture is as shown in Fig. 1, which consists of an input layer having 784 neurons because of the 28 × 28 size of the images in MNIST data set, we have considered deep network with 4 fully connected hidden layers and 50 neurons in each layer, the output layer consists of 10 neurons to classify the 10 digits in the data set. The *sigmoid* activation function is used for each of the neurons in the network and the *binary cross-entropy cost function* is used along with *L2 regularization*.

B. Addition of AOI noise :

We have considered the effect of other neurons present in the neighborhood of the neuron when calculating the output of it. The figure illustrating this technique is as shown in Fig. 2.

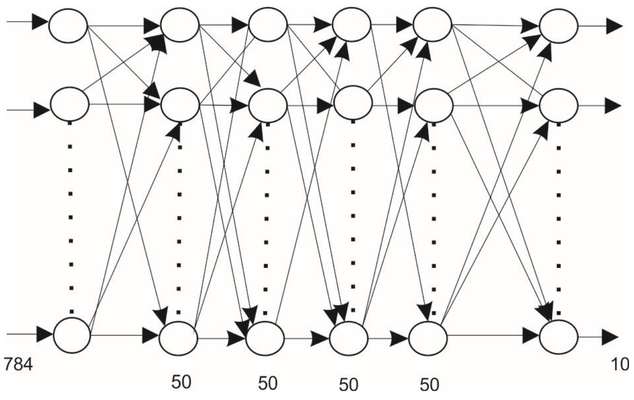

Fig. 1. Architecture of the neural network used.

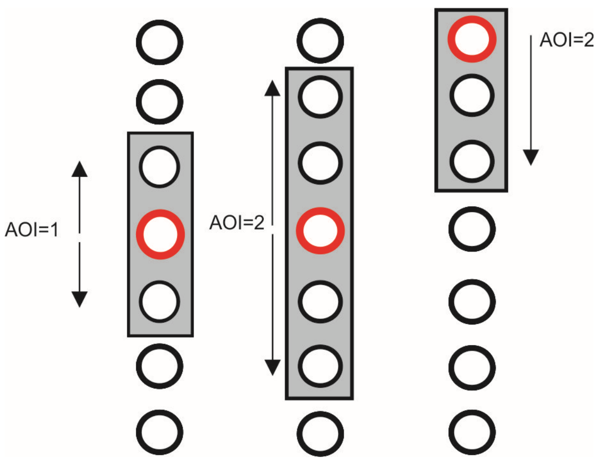

Fig. 2. Effect of different values of AOI.

It can be seen that when calculating the input applied to each neuron in the network a part of the input applied to the neurons in the neighborhood is added to it, the number of neurons effecting the nth neuron is given by the parameter AOI and the percentage of input of other neurons considered is given by the variable *err* as shown in (11).

$$Z_n^l = \frac{Z_{n-2}^l \times err}{10^{-((n-2)-n+1)}} + \frac{Z_{n-1}^l \times err}{10^{-((n-1)-n+1)}} + Z_n^l + \frac{Z_{n+1}^l \times err}{10^{-((n+1)-n+1)}} + \frac{Z_{n+2}^l \times err}{10^{-((n+2)-n+1)}} \tag{11}$$

Where Z is given by (12):

$$Z = w * x + b \tag{12}$$

Where *w* is the weight matrix, *x* is the input and *b* is the bias. From (11) it can be seen that value of err reduces by 1/10th it's previous value as the distance from the neuron increases. AOI 2 was considered while (11) was constructed. (12) is the input applied to the neurons in the network, where w is the weight, x the input applied and b the bias.

$$Z_3^l = \frac{Z_1^l \times err}{10^1} + \frac{Z_2^l \times err}{10^0} + Z_3^l + \frac{Z_4^l \times err}{10^0} + \frac{Z_5^l \times err}{10^1} \tag{13}$$

Equation (13) shows an example wherein the value of input Z applied to the 3rd neuron depends on the values of inputs of the surrounding 2 neurons i.e., for an AOI value of 2.

C. *Addition of AOI noise with randomness :*

Equation (14) is very similar to (13) except for the multiplication of each term in the neighborhood with r which is a Gaussian noise with mean of 0 and standard deviation of 0.1, we call this technique *aoi noise* with randomness.

$$Z_3^l = \frac{Z_1^l \times err \times r}{10^1} + \frac{Z_2^l \times err \times r}{10^0} + Z_3^l + \frac{Z_4^l \times err \times r}{10^0} + \frac{Z_5^l \times err \times r}{10^1} \tag{14}$$

D. *Line noise :*

In the third technique, we have added a small Gaussian noise to the output of each neuron in the network which can be interpreted as propagation noise which we call line noise as shown in Fig. 3., (15) and (16) shows this technique,

$$a = \sigma(z) \tag{15}$$

$$a = N(a, err \times a) \tag{16}$$

The Gaussian noise added has a mean which depends the output of the neuron and standard deviation which is a percentage of the output of that neuron. The percentage of output used as the standard deviation is given by the term *err*.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A series of experiments have been conducted to validate the three methods of adding noise described above which include *AOI noise*, *AOI noise with randomness* and *line noise.*

A. *The Dataset:*

We have used the MNIST data set, which contains 60000 images in the first part and 10000 in the second along with the labels for the images. We have considered only 1000 test and training images from this data set for training and testing purposes.

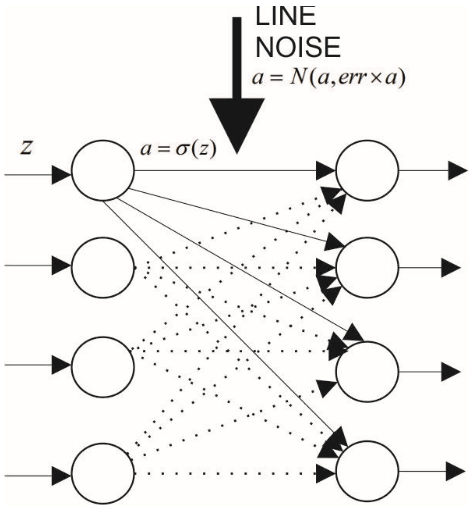

Fig. 3. Addition of line noise

B. *AOI noise:*

In this experiment we have added the *AOI noise* to the neurons in the network architecture which was described above. The learning rate used is 0.9 with batch size of 10 and AOI varying between 0 (no noise added) to 20. The network was trained for 1000 test images with 30 epochs in each trial and the average of 10 trials taken as the average test accuracy.

The result of this experiment is as shown in Table 1. It was found that addition of *AOI noise* to the network improved the average test accuracy form 71.39% for AOI 0 (no noise added) to maximum value of 79.48% for AOI 10. The improvement shown in all the tables is the percent change in the value form 71.39%.

C. *AOI noise with randomness:*

We next applied the *AOI noise with randomness* to the network architecture described above. The same learning rate to 0.9 with batch size 10 and AOI varying from 0 (no noise) to 20 as the previous experiment is carried out and the results are shown in Table 2.

It was found that the average test accuracy increased from 71.39% for AOI 0 (no noise) to 77.7% for AOI 20.

TABLE I. EXPERIMENTAL RESULTS FOR THE ADDITION OF AOI NOISE

AOI	**Results**	
	Average Recognition Rate (%)	***% Improvement***
0	71.39	-
3	72.67	4.4
5	75.89	15.7
10	**79.48**	**28.3**
15	73.49	7.3
20	78.34	24.3

TABLE II. EXPERIMENTAL RESULTS FOR THE ADDITION OF AOI NOISE WITH RANDOMNESS

AOI	**Results**	
	Average Recognition Rate (%)	***% Improvement***
0	71.39	-
3	72.67	4.4
5	76.07	16.4
10	72.5	3.8
15	75.31	13.7
20	**77.7**	**22.1**

TABLE III. EXPERIMENTAL RESULTS FOR THE ADDITION OF LINE NOISE

err %	**Results**	
	Average Recognition Rate (%)	***% Improvement***
Without noise	71.39	-
1	75.67	14.9
5	**77.41**	**21**
10	76.83	19.01
20	71.23	-0.55

D. *Line Noise:*

The *line noise* was next applied to the same network architecture with *err* chosen from {1%, 5%, 10%, 20%}, again the learning rate was fixed at 0.9 with batch size of 10, the network was trained for 20 epochs and the values tabulated by taking the average of 10 trials. The Table 3 shows the variation of the average test accuracy for different values of *err*, the maximum value of 77.41% test accuracy was found for an *err* of 5%.

The same methods where applied for very deep network consisting of 10 hidden layer and using the same 1000 image MNIST training and testing but we could not find any improvement in the recognition rate. The experiments were conducted using the numpy library, the entire source code for the three experiments can be found on the following GitHub link : https:github.com/Naresh1318/Effect_of_injected_noise_in deep_NN

V. Conclusion

The addition of noise due to interference from neighboring neurons in an artificial neural network is found to improve the accuracy of recognition in deep neural networks when the number of training samples are small. The recognition rate was found to be the maximum for AOI noise. This technique is found to be best suited for Deep Neural networks with less than 5 hidden layers and for data set consisting of limited number of test and training examples. The employment of this technique yields an increase in the average recognition rate by almost 8%. Future work would involve including this method for different artificial Neural Network architectures such as Convolution Neural Networks(CNN) [13] and Recurrent Neural Networks [14].

References

[1] De Boer, Pieter-Tjerk, Kroese, Dirk P, Mannor, Shie; Rubinstein, Reuven Y, "A Tutorial on the Cross-Entropy Method," Annals of Operations Research, vol. 134, no. 1, pp. 19–67, Feb. 2005.

[2] Duchi, John; Hazan, Elad; Singer, Yoram,"Adaptive subgradient methods for online learning and stochastic optimization," JMLR, vol. 12, pp. 2121-2159, Jan. 2011.

[3] Yves Chauvin, David E. Rumelhart, "Backpropagation: The Basic Theory," in Backpropogation: Theory, Architectures,and Applications, Hillsdale,New Jersey, 1995, pp. 7-19.

[4] Robert Reams, "Hadamard inverses, square roots and products of almost semidefinite matrices," Linear Algebra and its Applications, vol. 288, pp. 35-43, Feb. 1999.

[5] Q.Ashton Acton, "Machine Leaening," in Issues in Arti?cial Intelligence, Robotics and Machine Learning, 2013 Edition, Atlanta, Georgia, ScholarlyEditions, 2013.

[6] Akaike H, "Information theory and an extension of the maximum likelihood principle," In Second International Symposium on Information Theory, Budapest, pp. 267–281, 1973.

[7] McCulloch, Warren, "A Logical Calculus of Ideas Immanent in Nervous Activity," Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[8] A. G. IVAKHNENKO, "Polynomial Theory of Complex Systems," IEEE Trans. Syst. Man Cybern. , Vol. SMC-1, No. 4, pp. 364-378, Oct. 1971.

[9] Arvind Neelakantan, Luke Vilnis, Quoc V. Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, James Martens,"Adding Gradient Noise Improves Learning for Very Deep Networks," presented at the 4th Internation Conference on Learning Representations, San Juan, Puerto Rico, 2016.

[10] Chuan Wang, J.C. Principe, " Training neural networks with additive noise in the desired signal," IEEE Trans. Neural Netw., Vol. 10, Issue: 6, Nov. 1999.

[11] Kartik Audhkhasi, Osonde Osoba, Bart Kosko, " Noise-enhanced convolutional neural networks," Neural Networks(Elsevier Science Ltd. Oxford, UK), Vol. 78, Issue C, pp. 15-23, June 2016.

[12] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, MarcAurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Andrew Y. Ng, "Large Scale Distributed Deep Networks," in Neural Information Processing Systems 2012, Lake Tahoe, Nevada, 2012.

[13] Michael Nielsen.(2016, Jan).Neural Networks and Deep Learning[Online].Available: http://neuralnetworksanddeeplearning.com/

[14] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber,"A Novel Connectionist System for Improved Unconstrained Handwriting Recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, 2009.