Development of Neural Networks for Noise Reduction

Lubna Badri
Faculty of Engineering, Philadelphia University, Jordan

Abstract: *This paper describes the development of neural network models for noise reduction. The networks used to enhance the performance of modeling captured signals by reducing the effect of noise. Both recurrent and multi-layer Backpropagation neural networks models are examined and compared with different training algorithms. The paper presented is to illustrate the effect of training algorithms and network architecture on neural network performance for a given application.*

Keywords: *Noise reduction, recurrent neural networks, multi-layer backpropagation.*

Received January 3, 2009; accepted February 25, 2009

1. Introduction

In physical systems, transmitted signals are usually distributed partially, or sometimes almost completely, by an additive noise from the transmitter, channel, and receiver. The approach investigated in this work is to consider noise reduction as an essentially required process to enhance the estimation process of image reconstruction of the captured signal. Noise reduction is considered as a continuous mapping process of the noisy input data to a noise free output data. The resulted enhanced signal can be applied to the holographic imaging process and improves the performance of the estimated model. Artificial Neural Networks (ANNs) are finding increasing use in noise reduction problems [1, 2, 3, 4, 7, 8, 12, 13, 16, 17], and the main design goal of these Neural Networks (NNs) was to obtain a good approximation for some input-output mapping. In addition to obtaining a conventional approximation, NNs are expected to generalize from the given training data. The generalization is to use information that NN learned during training phase in order to synthesize, similar but not identical, input-output mapping [11].

In this paper, two different NN architectures are employed. These are Recurrent Neural Networks (RNNs) and MultiLayer Neural Networks (MLNNs). Both networks are trained with five training algorithms. The training functions used are: Gradient descent backpropagation (traingd), gradient descent with momentum backpropagation (traingdm), gradient descent with adaptive lr (learning rate) backprobagation (traingda), gradient descent w/momentum and adaptive lr backpropagation (traingdx), and Leverberg Marquardt backpropagation (trainlm).

The designed NNs are trained with input sequences that are assumed to be a composition of the desired signal plus an additive white Gaussian noise. The networks are expected to learn the noisy training data with the corresponding desired output and generalize the model. This research is an attempt to employ ANN for the enhancement of the measured corrupted signal and reduce the noise. The main contribution includes the following:

- The input training sequences to the designed NNs are assumed to be a composition of the desired signal plus an additive white Gaussian noise. This assumption speeds up the learning process and improves the approximation of the desired model [15].
- The development and comparison of NN architectures for use in noise reduction applications.
- A comparison of modeling performance using multi-layer and recurrent NNs.
- An examination of the relationship between training performance and training speed with the training algorithm used for a given NN architecture.

2. Artificial Neural Networks

There are two main phases in the operation of ANN: learning and testing. Learning is the process of adapting or modifying the NN weights in response to the training input patterns being presented at the input layer. How weights adapt in response to a learning example is controlled by a training algorithm. Testing is the application mode where the network processes a tested input pattern presented at its input layer and creates a response at the output layer.

Designing an ANN for a given application requires determining the NN architecture, the optimal size for the network (the total number of layers, the number of hidden units in the middle layers, and number of units in the input and output layers) in terms of accuracy on a test set, and the training algorithm used during the

learning phase. Two types of neural networks are used to perform the required extraction of the knowledge from a noisy training set to achieve better signal enhancement. These networks are RNN and MLNN. The architectures of both networks are presented in the following section.

2.1. Architecture

2.1.1. Recurrent Neural Network Architecture

The designed RNN is called Elman network. Elman networks are two-layer backpropagation networks, with the addition of a feedback connection from the output of the hidden layer to its input. This feedback path allows Elman networks to learn to recognize and generate temporal patterns, as well as spatial patterns [6]. A two-layer Elman network is shown in Figure 1.

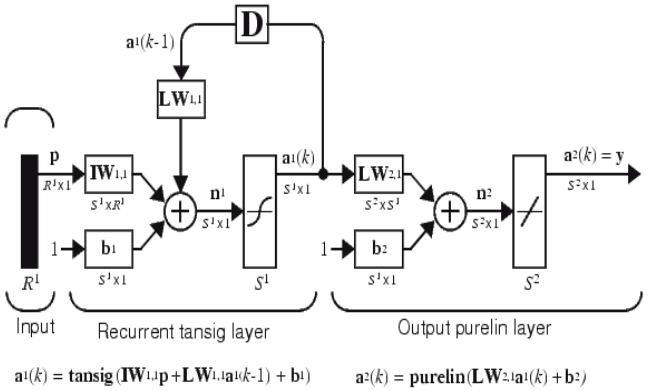

Figure 1. The architecture of Elman Network [10].

The Elman network constructed has tansig neurons in its hidden (recurrent) layer, and purelin neurons in its output layer, shown in Figures 2 and 3, respectively. The numbers of neurons in the hidden and output layers are 10 and 1, respectively. The hidden units and the output unit also have biases. These bias terms act like weights on connections from units whose output is always 1. The bias gives the network an extra variable, and so the network with bias is expected to be more powerful than those without [10]. This combination is special in that two-layer networks with these transfer functions can approximate any function (with a finite number of discontinuities) with arbitrary accuracy. The only requirement is that the hidden layer must have enough neurons. More hidden neurons are needed as the function being fitted increases in complexity. Note that the Elman network differs from conventional two-layer networks in that the first layer has a recurrent connection. The delay in this connection stores values from the previous time step, which can be used in the current time step. Thus, even if two Elman networks, with the same weights and biases, are given identical inputs at a given time step, their outputs can be different due to different feedback states [5].

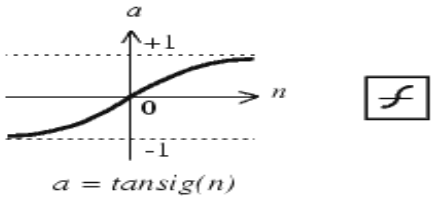

Figure 2. Tansig transfer function.

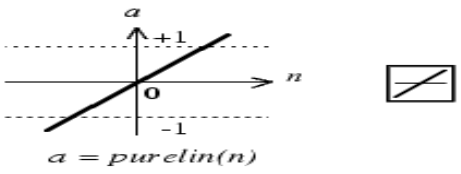

Figure 3. Purline transfer function.

Elman network performs the following:

1. The input units receive the first input.
2. Both the input units and context units (group of units that receives feedback signals from the previous time step [8]) activate the hidden units.
3. The hidden units also feedback to activate the context units (copying the content of the hidden unit).
4. The output units is compared with a teacher input (desired output) and backpropogation of error is used to incrementally adjust the connection strength.

The recurrent connections allow the network's hidden units to see its own previous output, so that the subsequent behaviour can be shaped by previous responses. These recurrent connections are what give the network memory. The context units are also "hidden" in the sense that they interact exclusively with other nodes internal to the network, and not the outside world [7].

2.1.2. Multilayer Neural Network Architecture

A MLNN is designed with three layers as shown in Figure 4. The feedforward network has two hidden layers of tansig neurons (f^1 and f^2) followed by an output layer of purelin neurons (f^3). The numbers of neurons in the first and second hidden layers are 7 and 3 respectively. The hidden units and the output unit also have biases. These bias terms act like weights on connections from units whose output is always 1. Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationship between input and output vectors. The backpropagation model is multilayered since it has distinct layers. The neurons within each layer are connected with the neurons of the adjacent layers through directed edges. There are no connections among the neurons within the same layer.

Only the direction of information flow for the feedforward phase of operation is shown. During the backpropagation phase of learning, signals are sent in the reverse direction.

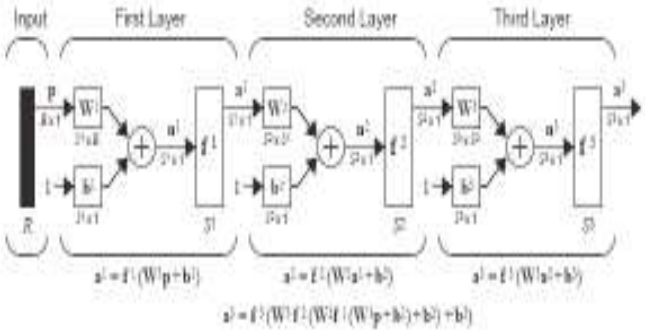

Figure 4. The Architecture of three-layer neural network [10].

Three-layer neural network performs the following: During feedforward phase, the input unit receives an input signal and broadcasts this signal to the each of the hidden units in the first hidden layer. Each of the hidden units then computes its activation and sends its signal to the hidden units in the second layer. Each hidden unit in the second layer computes its activation and sends its signal to the output units. Finally, the output unit computes its activation to form the response of the net for the given input pattern. During training phase, each output unit compares its computed activation with its target value to determine the error associated with that unit. The error is distributed from output layer back to all units in the next lower layer, and also used to update the weights between the output and the second hidden layer. The computed error in the second layer is distributed to all units in the previous layer and used to update the weights between the second hidden layer and the first hidden layer. The computed error in the first hidden layer is used to update the weights between the first hidden layer and the input layer.

2.2. Training Neural Networks

The neural networks are trained using backpropagation algorithm. There are several variations to the training algorithm of backpropagation NN. These variations are the basis of test procedures evaluation the overall most effective way to model the system. The distinction between training and generalization accuracies lies in the test patterns adopted. Good training accuracy can be achieved by forming complex decision boundaries, which in turn requires a large network size. Also, good generalization accuracy needs not to push too hard on the training accuracy; the overtraining may result in degraded generalization. This may occur if too many hidden units are used [12]. A number of network architectures have been designed and tested with different noisy data samples. The aim was to have good training process, to avoid overtraining problem, and to have better Mean Square Error (MSE) goal during the training process. It has been proven [10] that the addition of random noise to the desired signal during the training process of the neural network can improve the generalization of the network and can avoid the learning process from being trapped into local minimum. Assume *xk* denotes the k^{th} element of an input vector; *yi* is the i^{th} output of the output layer. Let *di(t)* denote the desired response for output neuron *i* at time *t*, where *t* is the discrete time index. The error signal *ei(t)* is defined as the difference between the target response *di(t)* and the actual response *yi(t)*.

$$e_i(t) = d_i(t) - y_i(t) \tag{1}$$

The aim of learning is to minimize a cost function based on the error signal *ei(t)*, with respect to network parameters (weights), such that the actual response of each output neuron in the network approaches the target response [6]. A criterion commonly used for the cost function is the MSE criterion, defined as the mean-square value of the sum squared error:

$$J = E\left[\frac{1}{2}\sum_i (e_i(t))^2\right] \tag{2}$$

$$= E\left[\frac{1}{2}\sum_i (d_i(t) - y_i(t))^2\right] \tag{3}$$

where E is the statistical expectation operator and the summation is over all the neurons of the output layer. Usually the adaptation of weights is performed by using the desired signal *di(t)* only. In [6] it is stated that a new signal $d_i(t) + n_i(t)$ can be used as a desired signal for output neuron i instead of using the original desired signal *di(t)*, where *ni(t)* is a noise term. This noise term is assumed to be white Gaussian noise, independent of both the input signals *xk(t)* and the desired signals *di(t)*. With the new desired signals, the MSE of equation 3 can be written as:

$$J = E[\frac{1}{2}\sum_i (d_i(t) + n_i(t) - y_i(t))^2] \tag{4}$$

It is shown in [6] that Eq.4 is equal to:

$$J = \frac{1}{2} E\left[\sum_i (y_i(t) - E\{d_i(t) + n_i(t) | x(t)\})^2\right]$$

$$+ \frac{1}{2} E\left[\sum_i \mathrm{var}((d_i(t) + n_i(t)) | x(t))\right] \tag{5}$$

where the symbol means conditional probabilities and 'var' is an abbreviation of variance. The second term in the right hand side of equation 5 will contribute to the total error J and as learning progresses, but it does not affect the final value of the weights because it is not a

function of the network weights, while the first term will decide the optimal value of the weights [6]. Since the noise is zero mean and it is independent of both desired and the input signals, thus:

$$\{E\{d_i(t) + n_i(t) \mid x(t)\} = \{E\{d_i(t) \mid x(t)\} \quad (6)$$

It is clear from equations 5 and 6 that the final weight values can be determined without the existence of noise in the equation. The training of NN was made to follow the model described by the following equation for holograph image process:

$$d(x) = A_r^2 + S^2 + 2A_r S Cos\,\theta \quad (7)$$

where θ is a phase difference between A_r and S is reflected signal from the object under imaging process, and A_r is reference signal due to the requirement of inline holography.

3. Experimental Results

In this section, the experimental results obtained using RNN and MLNN architectures on the recorded signal from a test object. A test object consists of two steel rods of 2.5cm diameter was used. The separation between the two rods was 7cm. The distance Zo between the object and recording planes was 90cm. The object was covered by two different opaque materials: a sheet of paper and a Styrofoam. The object was illuminated by ultrasound waves using ultrasonic transmitting transducers [2].

The received signals from the object due to the reflection are recorded and in order to enhance them, or in other words, to decrease the effect the environments such as the relatively high background that caused by the opaque material; the neural network is used to increase the SNR of holographic data. Designing neural network architecture for a given application requires determining the optimal size for the network in terms of accuracy on a test set, usually by increasing its size until there is no longer significant decrease in error. The analysis was performed through programs implemented on MATLAB software.

Both networks performed the required reduction of the noise in the captured signal, i.e., the RNN and MLNN. The MLNN performs better than RNN in terms of run time and also the Mean Square Error (MSE) performance. Tables 1 and 2 summarize the results of NN's training by comparing the Elapsed time, Epochs, and the MSE of RNN and MLNN with five training algorithms. These algorithms are Traingd, Traingdm, Traingda, Trainlm, and Traingdx [5]. All of these algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance (i.e., MSE). As can be seen in Tables 1 and 2, the MLNN trained with TRAINLM backpropagation function results in a fastest algorithm implementation (90 epochs) with a best performance (MSE equals 0.0112285).

Table 1. Elapsed time and epochs of RNN and MLNN with five training algorithms

	Recurrent Neural Network (RNN)		Multilayer Neural Network (MLNN)	
Algorithm	**Elapsed Time (Sec)**	**Epochs**	**Elapsed Time (Sec)**	**Epochs**
Traingd	7.563	1800	24.047	7000
Traingdm	8.328	2000	75.235	20000
Traingda	8.656	2000	7.813	2000
Trainlm	4.781	500	1.484	90
Traingdx	4.563	1000	12.593	3500

Table 2. Mean square error of RNN and MLNN with five training algorithms.

	Mean Square Error (MSE)	
Algorithm	**Recurrent Neural Network (RNN)**	**Multilayer Neural Network (MLNN)**
Traingd	0.0151797	0.0153542
Traingdm	0.0151767	0.014463
Traingda	0.016874	0.0157362
Trainlm	0.0111192	0.0112285
Traingdx	0.0150799	0.0126875

The trainlm function works according to levenberg-marquardz optimization technique [5]. Figure 5 shows MLNN performance during the training process with TRAINLM algorithm. Extensive testing is made to improve the performance of MLNN. Table 3 shows the MLNN performance with two experiments cases: sheet of paper and Styrofoam isolated materials. The designed MLNN was able to decrease the effect of concealing medium and the noise in the captured signal. Figure 6 shows the behavior of the neural network output after applying the captured signal h(n) in the sheet of paper and styrofoam cases, respectively.

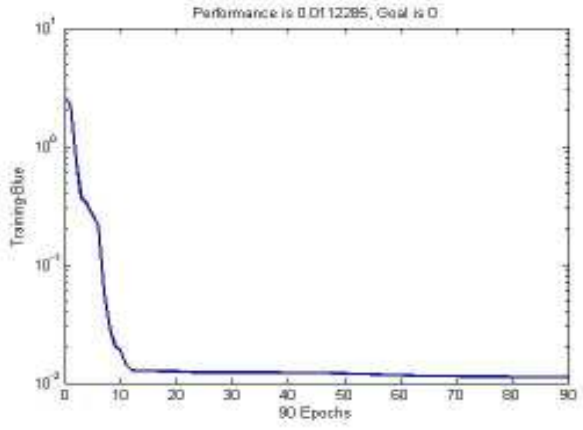

Figure 5. Multilayer neural network MLNN performance during the training process with TRAINLM training algorithm.

4. Conclusions

In this paper, two different neural networks have been compared to minimize the effect of noise in the model of a concealed object and increase the SNR of holographic data. Experimental results show that using

neural network to enhance the captured signal can improve the tracking of the model parameters. The RNN and MLNN architecture have been studied and tested to obtain the optimal architecture in terms of number of hidden layers and neurons in each layer. The results obtained show that during the pre-processing stage, the RNN and MLNN were able to enhance the tested recorded signal and produce an output signal that follows the desired model with minimum MSE (0.0112285). The effect of adding white Gaussian noise to the desired signal when training the neural network with backpropagation has been discussed. Both analytically and experimentally it has been demonstrated that the additive noise improves the network generalization on the tested patterns and the training trajectory. Similar results have been obtained when training both RNN and MLNN.

Table 3. Multilayer neural network performance with two experiments cases: sheet of paper and Styrofoam isolating materials.

Training Algorithm	Mean Square Error (MSE)	
	Sheet of Paper Case	Styrofoam Case
Traingd	0.053542	0.177533
Traingdm	0.014463	0.170649
Traingda	0.0157362	0.15886
Trainlm	0.0112285	0.149683
Traingdx	0.0126875	0.164845

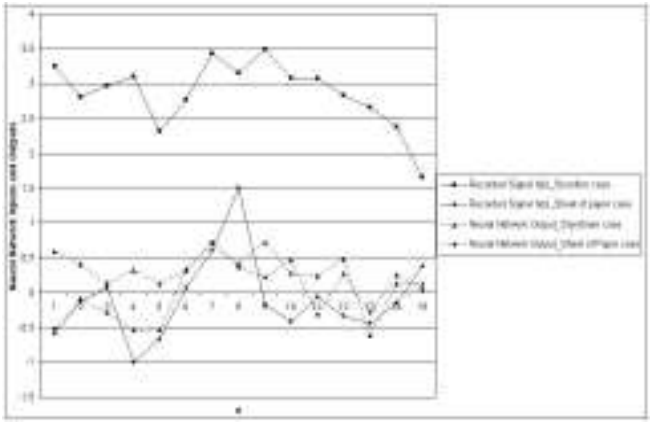

Figure 6. Multilayer neural network MLNN output after applying the captured signal h(n) in two cases: sheet of paper and Styrofoam opaque materials.

References

[1] Badri L. and Al-Azzo M., "Burg-Neural Network Based Holographic Source Localization," *WSEAS Transactions on Signal Processing*, vol. 2, no. 14, pp. 414-422, 2006.

[2] Badri L. and Al-Azzo M., "Modelling of Long Wavelength Detection of Objects Using Elman Network Modified Covariance Combination," *International Arab Journal of Information Technology (IAJIT)*, vol. 5, no. 3, pp. 265-272, 2008.

[3] Brueckmann R., Scheidig A., and Gross H., "Adaptive Noise Reduction and Voice Activity Detection for improved Verbal Human-Robot Interaction using Binaural Data," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Italy, pp. 10-14, 2007.

[4] Chuan W. and Jose P., "Training Neural Networks with Additive Noise in the Desired Signal," *Transactions on Neural Networks*, vol. 10, no. 6, pp. 1511-1517, 1999.

[5] Demuth H. and Beale M., Neural Network Toolbox-for Use with MATLAB User's Guide, *The Mathworks*, Massachusetts, 2002.

[6] Dorronsoro J., López V., Cruz C., and Sigüenza J., "Autoassociative Neural Networks and Noise Filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 5, pp. 1431-1438, 2003.

[7] Elman J., "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp.179-211, 1990.

[8] Fausett L., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice Hall International, New Jersey, 1994.

[9] Giles C., Lawrence S., and Tsoi A., "Noisy Time Series Prediction Using a Recurrent Neural Network and Grammatical Inference," *Machine Learning*, vol. 44, no. 1-2, pp. 161-183, 2001.

[10] Hagan M., Demuth H., and Beale M., *Neural Network Design*, PWS Publishing Company and Thomson Asia, 2002.

[11] Khairnar D., Merchant S., and Desai U., "Radar Signal Detection In Non-Gaussian Noise Using RBF Neural Network," *Journal of Computers*, vol. 3, no. 1, pp. 32-39, 2008.

[12] Kung S., *Digital Neural Network*, Printice Hall, 1993.

[13] Mastriani M. and Giraldez A., "Neural Shrinkage for Wavelet-Based SAR Despeckling," *International Journal of Intelligent Technology*, vol. 1, no. 3, pp. 211-222, 2006.

[14] Parveen S. and Green P., "Speech Enhancement with Missing Data Techniques Using Recurrent Neural Networks," *in Proceedings of the IEEE International Conference on Acoustics*, *Speech and Signal Processing*, Canada , pp. 1733-1736, 2004.

[15] Radonja P., "Neural Networks Based Model of A Highly Nonlinear Process," *in Proceedings of the IX Telekomunikacioni Forum Telfor'2001*, Beograd, 2001.

[16] Yoshimura H., Shimizu T., Matumura T., Kimoto M., and Isu N., "Adaptive Noise Reduction Filter for Speech Using Cascaded Sandglass-Type Neural Network," *in Proceedings of the IEEE International Conference on Robotics and Automation*, Italy, pp. 10-14, 2007.

[17] Zhang X., "Thresholding Neural Network for Adaptive Noise Reduction," *IEEE Transactions on Neural Networks*, vol. 12, no. 3, 2001.

Lubna Badri received her BSc and MSc degree in computer and control engineering, from the University of Technology, Baghdad in 1994 and 1996, respectively, and the PhD degree in computer engineering, from University of Technology in Baghdad, Iraq, in 1999. Her research interest includes neural network and fuzzy logic, knowledge acquisition systems, and embedded system design. She has one book and more than 15 publications in reputed journals and conferences.