Empirical studies on weight noise injection based online learning algorithms

John Sum, Yen-lun Liang
ITM, National Chung Hsing University
Taichung, Taiwan.
Email:pfsum@nchu.edu.tw

Chi-sing Leung
EE, City University of Hong Kong
Kowloon Tong, Hong Kong.
Email:eeleungc@cityu.edu.hk

Kevin Ho
CSCE, Providence University
Sha-Lu, Taichung, Taiwan.
Email:ho@pu.edu.tw

***Abstract*—While weight noise injection during training has been adopted in attaining fault tolerant neural networks (NNs), theoretical and empirical studies on the online algorithms developed based on these strategies have yet to be complete. In this paper, we present results on two important aspects in online learning algorithms based on combining weight noise injection and weight decay. Through intensive computer simulations, the convergence behaviors of those algorithms and the performance of the NNs generated by these algorithms are elucidated. It is found that (i) the online learning algorithm based on purely multiplicative weight noise injection does not converge, (ii) the algorithms combining weight noise injection and weight decay exhibit better convergence behaviors than their pure weight noise injection counterparts, and (iii) the neural networks attained by those algorithms combining weight noise injection and weight decay show better fault tolerance abilities than the neural networks attained by the pure weight noise injection-based algorithms. These empirical results not just can supplement our recent work done on the convergence behaviors of the weight noise injection-based learning algorithms [16], but also provide new information on effect of weight noise and weight decay on the neural networks that are generated by these algorithms.**

***Keywords*-Cross Entropy Error, MLP, Mean Square Errors, Regression, Weight Decay, Weight Noise Injection**

I. Introduction

Many methods have been developed throughout the last two decades to improve tolerance of a neural network towards random node fault, stuck-at node fault and weight noise. Known methods include injecting random or stuck-at node fault during training [26], [4], injecting (synaptic) weight noise during training (specially for multilayer perceptrons (MLP) [22], [23], a recurrent neural network (RNN) [17], or a pulse-coupled neural networks (PCNN) [11], injecting node noise (response variability) during training [2] (specifically for a model of PCNN) applying weight decay training [8], introducing network redundancy [25], formulating the training algorithm as a nonlinear constraint optimization problem [9], [24], bounding weight magnitude during training [6], [12], [18], and adding fault tolerant regularizer [3], [19], [28]. Amongst all, the weight/node noise-injection-based on-line training algorithms are of least theoretical studied [1], [2], [22], [23]. Especially, the objective functions and the convergence properties of these algorithms for multilayer perceptron (MLP) have not been derived and proved.

Murray & Edward in [10], [23] have derived the prediction error of a (trained) MLP if multiplicative weight noise is injected after training (see Section II.A and II.B in [23]). For the dynamics of the weight vector during training, only a qualitative analysis has been presented (see Section II.C in [23]). Convergence proof and objective function have not been analyzed. An has attempted to derive an objective function for this weight-noise injection training algorithm (see Section 4 in [1]). However, An has not succeeded to a convergence proof. The objective function derived is not true objective function for *training with weight noise injection*. It is again the prediction error of a trained MLP if weight noise is injected after training. Until recently, Ho *et al* [13] have showed that the convergence of output weight noise injection-based training a radial basis function (RBF) network is almost sure. For MLP, theoretical study on the effect of injecting weight noise during training is still missing. The performance of the neural network generated by these algorithms is unknown.

Table I
ALGORITHMS TO BE INVESTIGATED IN THIS PAPER. BPA STANDS FOR THE STANDARD BACKPROPAGATION ALGORITHM.

Algorithm	Objective	Structure
BPA	MSE	Linear output
BPA+WN	(8) & (10)	Linear output
BPA+WD	MSE + WD	Linear output
BPA+WN+WD	(8) & (10)	Linear output
BPA	CEE	Sigmoid output
BPA+WD	CEE + WD	Sigmoid output
BPA+WN	Unknown	Sigmoid output
BPA+WN+WD	Unknown	Sigmoid output

MSE: Mean Square Errors, CEE: Cross entropy error
WN: Weight Noise, WD: Weight Decay

In this regard, the convergence behaviors of the weight noise injection-based learning algorithms are investigated in this paper. Besides, the performance of the neural networks that are generated by these algorithms are studied. The main contribution of this paper is to provide intensive simulation results to supplement our theoretical work done in other papers [16], [27]. In the next section, the algorithms to be studied in the paper will be defined. The datasets and the simulation results are elucidated in Section 3. Section 4 gives the conclusions.

II. Weight noise injection during training

The learning algorithms to be investigated in this paper are depicted in Table I. In which four of them are originated from the conventional BPA based on minimizing mean square errors (MSE). While the other four are originated from the BPA based on minimizing cross entropy error (CEE).

A. MSE-based

Let $\mathbf{f}(\cdot,\cdot) \in R^l$ be a single output multilayer perceptron (MLP) consisting of m hidden nodes, n input nodes and l linear output nodes.

$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \mathbf{D}^T\mathbf{z}(\mathbf{A}^T\mathbf{x}+\mathbf{c}), \tag{1}$$

where $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_l] \in R^{m\times l}$ is the hidden to output weight vector, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m] \in R^{n\times m}$ is the input to hidden weight matrix, $\mathbf{a}_i \in R^n$ is the input weight vector of the i^{th} hidden node and $\mathbf{c} \in R^m$ is the input to hidden bias vector. $\mathbf{w}$ in (1) is a vector augmenting all the parameters, i.e. $\mathbf{w} = (\mathbf{d}_1^T, \mathbf{d}_2^T, \cdots, \mathbf{d}_l^T, \mathbf{a}_1^T, \mathbf{a}_2^T, \cdots, \mathbf{a}_m^T, \mathbf{c}^T)^T$. $\mathbf{z} = (z_1, z_2, \cdots, z_m)^T \in R^m$ is the hidden output vector. For $i = 1, 2, \cdots, m$,

$$z_i(s) = \frac{1}{1+\exp(-s)}. \tag{2}$$

Training dataset is denoted by $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^N$. The MSE is defined as follows :

$$MSE = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2. \tag{3}$$

For simplicity, we assume that there is only one output node, i.e. $l = 1$. In such case, the gradient of $f(\mathbf{x},\mathbf{w})$ with respect to $\mathbf{w}$ is denoted by $g(\mathbf{x}_t, \mathbf{w}(t))$. The Hessian matrix of $f(\mathbf{x},\mathbf{w})$ is denoted by $g_{\mathbf{w}}(\mathbf{x}_t, \mathbf{w}(t))$.

The online **weight noise injection** training for $f(\mathbf{x},\mathbf{w})$ given a dataset $\mathcal{D}$ can be written as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t(y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)), \tag{4}$$

where

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b}\otimes\mathbf{w}(t). \quad \text{(multi. noise)} \tag{5}$$
$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b}. \quad \text{(additive noise)} \tag{6}$$

Here $\mathbf{b}\otimes\mathbf{w} = (b_1w_1, b_2w_2, \cdots, b_Mw_M)^T$ and b_i, for all i, is a mean zero Gaussian distribution with variance S_b.

For **simultaneous weight noise injection and weight decay**, the update equations are similar except the decay term is added.

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t\left\{(y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) - \alpha\mathbf{w}(t)\right\}. \tag{7}$$

It has been shown in that the objective function being minimized by injecting multiplicative weight noise with weight decay during training is given by [14]

$$\mathbf{V}(\mathbf{w}) = \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M}w_j^2 g_j(\mathbf{x}_k,\mathbf{w})^2\right\} - \frac{S_b}{N}\sum_{k=1}^{N}\int_{\mathbf{w}_0}^{\mathbf{w}}\mathbf{u}(\mathbf{x}_k,\mathbf{r})\mathbf{d}r + \frac{\alpha}{2}\|\mathbf{w}\|^2. \tag{8}$$

where

$$\mathbf{u}(\mathbf{x}_k,\mathbf{w}) = \left[\begin{array}{ccc} w_1g_1(\mathbf{x}_k,\mathbf{w})^2 & \cdots & w_Mg_M(\mathbf{x}_k,\mathbf{w})^2 \end{array}\right] \tag{9}$$

The objective function being minimized by injecting additive weight noise with weight decay is given by [14]

$$\mathbf{V}(\mathbf{w}) = \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M}g_j(\mathbf{x}_k,\mathbf{w})^2 + \alpha\|\mathbf{w}\|^2\right\}. \tag{10}$$

While the parameter α in (8) and (10) is zero, the functions are the objective functions of the pure weight noise injection-based learning algorithms.

B. CEE-based

Cross entropy error (CEE)-based algorithms applied mainly to the MLP with sigmoid output neurons. Let us denote the output by $\mathbf{h}(\mathbf{x},\mathbf{w})$. The output is defined as follows :

$$\mathbf{h}(\mathbf{x},\mathbf{w}) = \mathbf{z}(\mathbf{f}(\mathbf{x},\mathbf{w})) \tag{11}$$
$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \mathbf{D}^T\mathbf{z}(\mathbf{A}^T\mathbf{x}+\mathbf{c}), \tag{12}$$

where the parameters $\mathbf{D}$, $\mathbf{z}$, $\mathbf{A}$, $\mathbf{a}_i$s and $\mathbf{c}$ are defined in the same way as for the linear output MLP. The CEE is defined as follows :

$$CEE = \frac{1}{N}\sum_{k=1}^{N} y_k \ln h(\mathbf{x}_k,\mathbf{w}) + (1-y_k)\ln(1-h(\mathbf{x}_k,\mathbf{w})). \tag{13}$$

The online **weight noise injection** training for $h(\mathbf{x},\mathbf{w})$ given a dataset $\mathcal{D}$ can be written as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t(y_t - h(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)), \tag{14}$$

where $\tilde{\mathbf{w}}(t)$ is defined in the same way as in (5) and (6) depended on the noise type.

For **simultaneous weight noise injection and weight decay**, the update equations are given by

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t\left\{(y_t - h(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) - \alpha\mathbf{w}(t)\right\}, \tag{15}$$

where $\tilde{\mathbf{w}}(t)$ is defined in the same way as in (5) and (6) depended on the noise type.

While the update equations for the learning algorithms (14) and (15) are similar to the MSE-based algorithms (4) and (7), their objective functions have not been discovered.

III. Simulations

In this section, we will first present the datasets being used for studies and then methodology to evaluate the performance of the networks will be followed.

Table II
TRAINING DATA AND TESTING DATA FOR THE SIMULATIONS.

	2D Map	MG	NAR	XOR	Char. Recog.
Train data	100	500	500	100	800
Test data	100	500	500	100	793

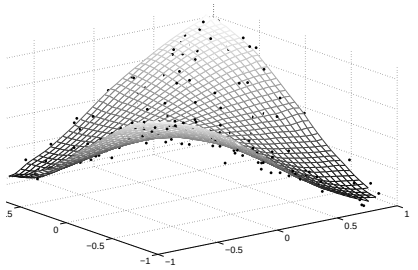

Figure 1. 2D mapping data.

A. Datasets

Five datasets are used to examine the convergence properties of the algorithms and the performance of the neural networks being generated. The datasets are used for (i) 2D mapping, (ii) Mackey-Glass, (iii) NAR, (iv) XOR and (v) Character recognition, in which three of them are for regression and two are for classification. The numbers of the training data and testing data for the simulations are depicted in Table II.

1) 2D Mapping: It is as artificial dataset consisting of 200 data points, which is generated from the following equation.

$$y_k = \sin(x_{k1})\sin(x_{k2}) + e_k, \tag{16}$$

where x_{k1}, x_{k2} are the k^{th} sample inputs and y_k be its target output, e_k is a mean zero Gaussian noise with variance 0.01. Amongst these 200 data points, 100 of them are randomly selected to be the training dataset, and the other 100 data points are the testing dataset. Fig. 1 shows the data set.

2) Mackey-Glass: It is a benchmark time-series dataset available on the Internet. The data is generated by the differential equation given by

$$\frac{dx(t)}{dt} = 0.2\frac{x(t-\tau)}{1+x(t-\tau)^{10}} - 0.1x(t), \tag{17}$$

with $x(0) = 1.2$ and $\tau = 17$. In our simulations, we use 1000 points in the time series as shown in Fig. 2. The first 500 points (from $k = 5$ to 504) are picked for training and others are for testing. A MLP is trained to predict the current value $y(k)$ based on the past observations $y(k-1), y(k-2), y(k-3), y(k-4)$. In other words, $x_k = [y(k-1), y(k-2), y(k-3), y(k-4)]^T$.

3) NAR: We consider the following nonlinear autoregressive (NAR) time series [7], given by

$$\begin{aligned} y(k) &= (0.8 - 0.5\exp(-y^2(k-1)))y(k-1) \\ &\quad -(0.3 + 0.9\exp(-y^2(k-1)))y(k-2) \\ &\quad +0.1\sin(\pi y(k-1)) + e(k), \end{aligned} \tag{18}$$

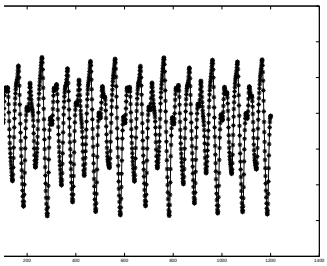

Figure 2. Mackey-Glass time series.

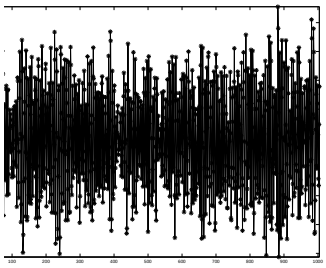

Figure 3. NAR time series.

where $e(k)$ is a mean zero Gaussian random variable with variance equals to 0.09. One thousand samples were generated given $y(0) = y(1) = 0.1$, as shown in Fig. 3. The first 500 data points were used for training and the other 500 points were used for testing. The task is used to predict $y(k)$ based on the past observation $y(k-1)$ and $y(k-2)$. In other words, $x_k = [y(k-1), y(k-2)]^T$.

4) XOR: It is an artificial data set with two inputs and one output. Let x_{k1}, x_{k2} be the input and y_k is the target output.

$$y_k = sign(x_{k1})sign(x_{k2}). \tag{19}$$

Amongst these 200 data points, 100 of them are randomly selected as the training dataset, and the other 100 data points are the testing dataset.

5) Character Recognition: Character recognition data set [5] [1] consists of 1593 handwritten digits collected from around 80 people. Each person wrote on a paper all the digits from 0 to 9, twice. The digits were scanned and stretched in a 16x16 rectangular box in a gray scale of 256. Then each pixel of each image was scaled into a binary (1/0) value using a fixed threshold. Fig. 4 shows the sample data set. The 800 data set is as training data and the rest 793 data as testing data.

B. Methodology

Table III summarizes the structure of the networks for simulations. For the first three simulations, we use linear output MLPs. For the last two simulations, we use sigmoid output MLPs.

[1] archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit

Figure 4. Samples of character recognition data.

Table III
NODE NUMBERS AND NODE TYPES WHICH ARE DEFINED FOR SIMULATIONS.

	2D Map	MG	NAR	XOR	Char. Recog.
Input	2	4	2	2	256
Hidden(*)	10	10	10	6	20
Output	1 LN	1 LN	1 LN	1 SN	10 SN

Hidden nodes are all sigmoid nodes.
SN : Sigmoid node; LN : Linear node

C. Noise variance and weight decay

Five different weight noise variance $(0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2})$, and the four different weight decay constants $(0, 10^{-5}, 10^{-4}, 10^{-3})$ are set during training. Altogether, 20 different settings are investigated for each dataset. The step size for all simulations are set to 0.1 and the total number of epoches is 10^5.

D. Results

We study the change of weights during training so as to identify the convergence behaviors of the learning algorithms. Besides, the fault tolerances of the neural networks being generated are investigated.

Let $f(\mathbf{x}, \mathbf{w})$ be the trained MLP with linear output node and $h(\mathbf{x}, \mathbf{w})$ be the trained MLP with sigmoid output node. For the regression problems, we measure the performance of a MLP by its prediction errors:

$$MSE = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} (y_k - f(\mathbf{x}_k, \mathbf{w} + \triangle\mathbf{w}))^2, \qquad (20)$$

where $\triangle\mathbf{w}$ is the weight noise added during testing. For the classification problems, we measure the performance of a MLP by its classification errors:

$$CE = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} |(y_k - sign(h(\mathbf{x}_k, \mathbf{w} + \triangle\mathbf{w}) - 0.5))|, \quad (21)$$

where $\triangle\mathbf{w}$ is the weight noise added during testing.

For a neural network that is generated by multiplicative weight noise injection during training, multiplicative weight noise with different variance will be added during testing. The weight noise injected during testing is with variances $[0, 0.002, 0.004, ..., 0.04]$, a total of 21 cases. For each noise variance, 100 networks are generated and their testing MSEs are recorded. Then, we evaluate the fault tolerance ability of a neural network by the average testing MSEs. For a neural network that is generated by additive weight noise injection during training, the evaluation method is the same except that the noise injected is additive instead of multiplicative. Here, we only present the key results. For the results of all settings, readers can refer to [20].

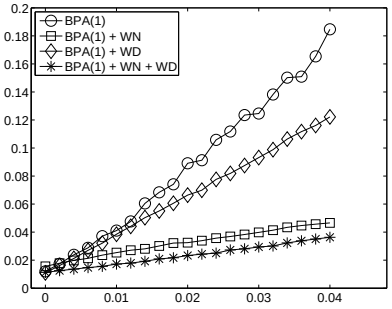

Figure 5. Testing MSE of 2D mapping-MWN.

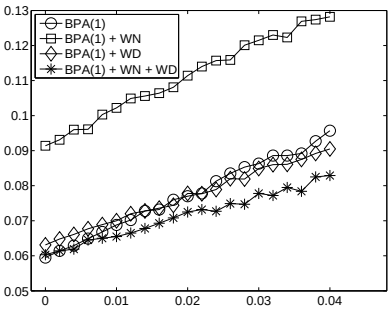

Figure 6. Testing MSE of 2D mapping-AWN.

1) 2D mapping: For MWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training gives the best performance. The MLP generated by the algorithm based on pure back-propagation gives the worst performance, as shown in Fig 5. For AWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training also gives the best performance. The MLP generated by the algorithm based on only weight noise injection gives the worst performance, as shown in Fig 6.

For both cases of MWN and AWN, it is clear from Fig 7 and Fig 8 that the MSE value converges during training. However, the weights do not converge when $\alpha = 0$. The weights diverge when $\alpha = 0$ and $S_b = 10^{-2}$. Once a small weight decay has been added, say $\alpha = 10^{-5}$, the weights converge.

2) Mackey-Glass: For MWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training gives the best performance. The MLP generated by the algorithm based on pure back-propagation gives the worst performance, as shown in Fig 9. For AWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training also gives the best performance. The MLP generated by the algorithm based on only weight noise injection gives the worst performance, as shown in Fig 10.

For both cases of MWN and AWN, it is clear from Fig 11 and Fig 12 that the MSE value converges during training. However, the weights do not converge when $\alpha = 0$. The

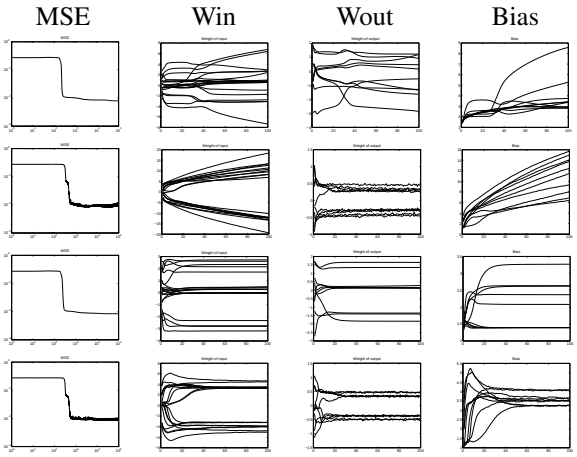

Figure 7. Change of the MSE and weights against time for the 2D mapping problem with multiplicative weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

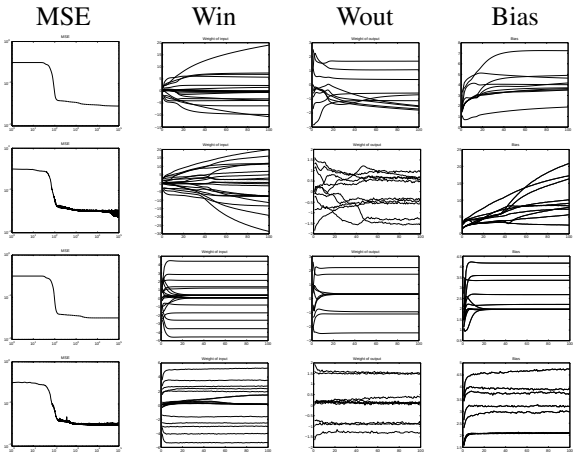

Figure 8. Change of the MSE and weights against time for the 2D mapping problem with additive weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

weights diverge when $\alpha = 0$ and $S_b = 10^{-2}$. Once a small weight decay has been added, say $\alpha = 10^{-5}$, the weights converge.

3) NAR: For MWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training gives the best performance. The MLP generated by the algorithm based on pure back-propagation gives the worst performance, as shown in Fig 13. For AWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training also gives the best performance. The MLP generated by the algorithm based on only weight noise injection gives the worst performance, as shown in Fig 14.

For both cases of MWN and AWN, it is clear from Fig 15 and Fig 16 that the MSE value converges during training. However, the weights do not converge when $\alpha = 0$. The weights diverge when $\alpha = 0$ and $S_b = 10^{-2}$. Once a small weight decay has been added, say $\alpha = 10^{-5}$, the weights converge.

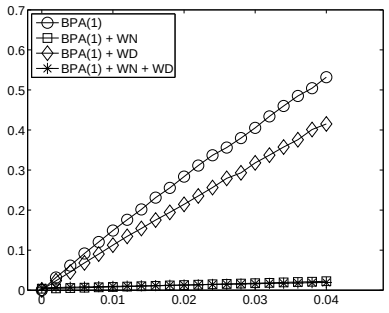

Figure 9. Testing MSE of Mackey-Glass-MWN.

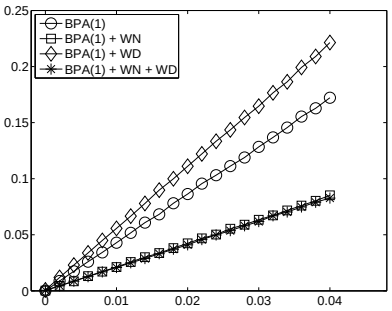

Figure 10. Testing MSE of Mackey-Glass-AWN.

4) XOR: For MWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training gives the best performance. The MLP generated by the algorithm based on pure back-propagation gives the worst performance, as shown in Fig 17. For AWN, the MLP generated by the algorithm based on combining weight noise injection with weight decay during training also gives the best performance. The MLP generated by the algorithm based on only weight noise injection gives the worst performance, as shown in Fig 18.

For both cases of MWN and AWN, it is clear from Fig 19 and Fig 20 that the MSE value converges during training. However, the weights do not converge when $\alpha = 0$. The weights diverge when $\alpha = 0$ and $S_b = 10^{-2}$. Once a small weight decay has been added, say $\alpha = 10^{-5}$, the weights converge.

5) Characters recognition: For the case of multiplicative weight noise injection during training, we have the following findings as shown Fig 21. (1) When the noise variance is less than 0.02, the MLP generated by adding weight decay during training gives the best performance. (2) When the noise variance is greater than 0.02, the MLP generated by the algorithm based combining weight noise injection with weight decay during training gives the best performance. The MLP generated by the algorithm based on weight noise injection gives the worst performance.

For the case of additive weight noise injection during training, we have the following findings as shown Fig 22. (1) When the noise variance is less than 0.01, the MLP generated by adding weight decay during training gives the best performance. (2) When the noise variance is in between

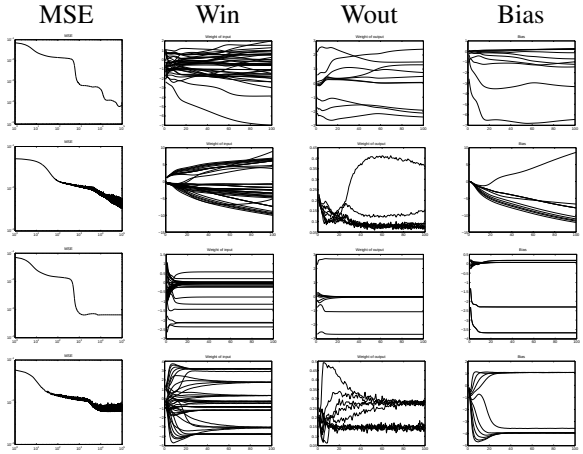

Figure 11. Change of the MSE and weights against time for the Mackey-Glass problem with multiplicative weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

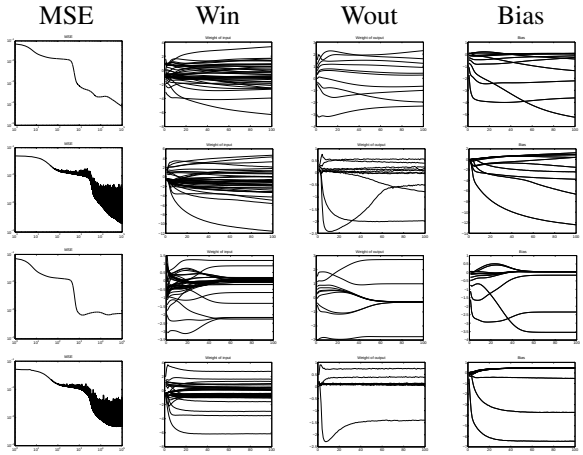

Figure 12. Change of the MSE and weights against time for the Mackey-Glass problem with additive weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

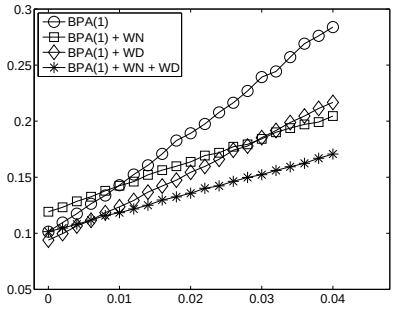

Figure 13. Testing MSE of NAR-MWN.

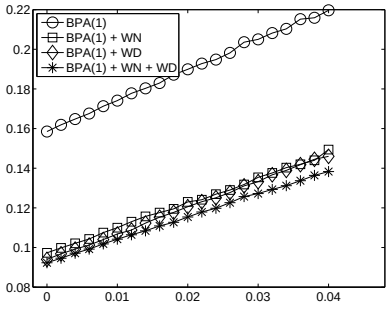

Figure 14. Testing MSE of NAR-AWN.

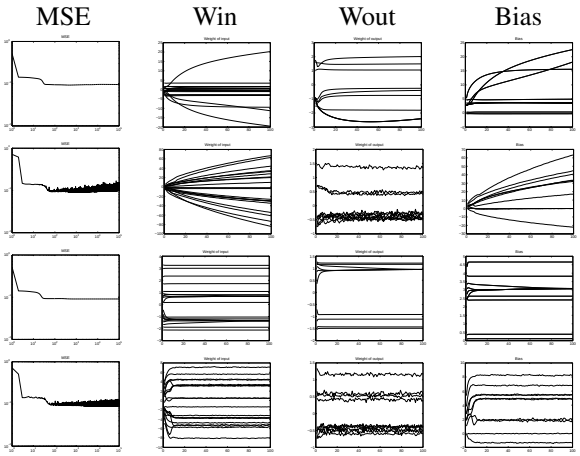

Figure 15. Change of the MSE and weights against time for the NAR problem with multiplicative weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

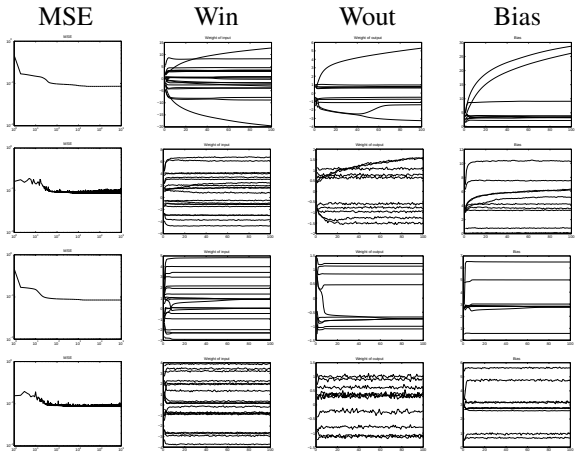

Figure 16. Change of the MSE and weights against time for the NAR problem with additive weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-5}, 0)$ and $(10^{-5}, 10^{-2})$.

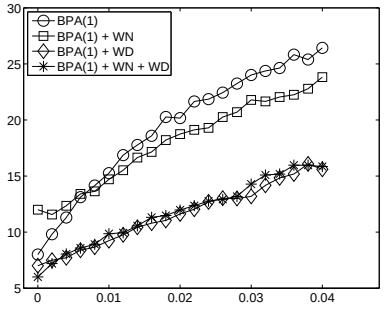

Figure 17. Classification error of XOR-MWN.

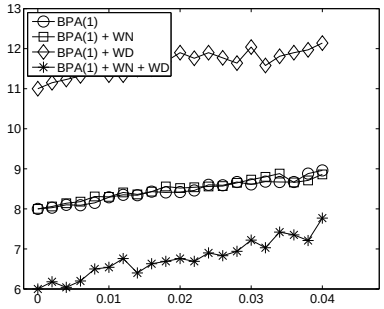

Figure 18. Classification error of XOR-AWN.

0.01 and 0.03, the MLP generated by combining weight noise injection with weight decay during training gives the best performance. (3) When the noise variance is greater than 0.03, the MLP generated by weight noise injection during training gives the best performance.

For both cases of MWN and AWN, it is clear from Fig 19 and Fig 20 that the MSE value converges during training. However, the weights do not converge when $\alpha = 0$. The weights diverge when $\alpha = 0$ and $S_b = 10^{-2}$. Once a small weight decay has been added, say $\alpha = 10^{-5}$, the weights converge.

IV. Conclusions

In this paper, we have altogether presented eight different types of online weight noise injection-based algorithms. In which, four of them are extended from the original back-propagation algorithm based on minimizing mean square errors (MSE). While the other four are extended from the original back-propagation algorithm based on minimizing cross entropy error (CEE). For those algorithms based on MSE, the output neuron is defined as a linear neuron. For those algorithms based on CEE, the output neuron is defined as a sigmoid neuron. To study the convergence behaviors of these algorithms and the fault tolerance abilities of the multilayer perceptron (MLP) models attained by these learning algorithms, extensive computer simulations are conducted based on five datasets. Simulation results show that **the online learning algorithm based on purely multiplicative weight noise injection does not converge**. These results complement a recent analysis by us [16], [27] on the objective functions of the weight noise injection-based algorithms. Besides, simulation results show that **the**

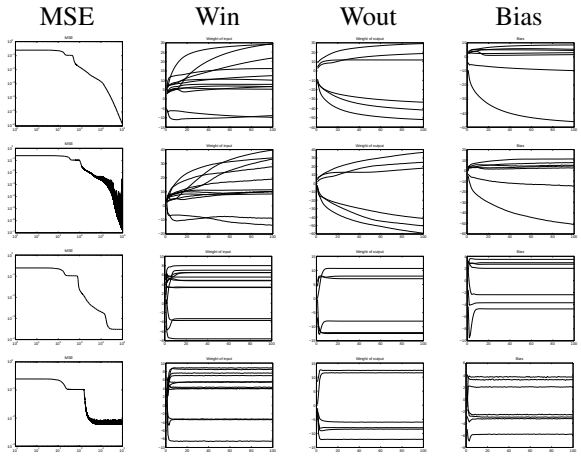

Figure 19. Change of the MSE and weights against time for the XOR problem with multiplicative weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-3})$, $(10^{-4}, 0)$ and $(10^{-4}, 10^{-3})$.

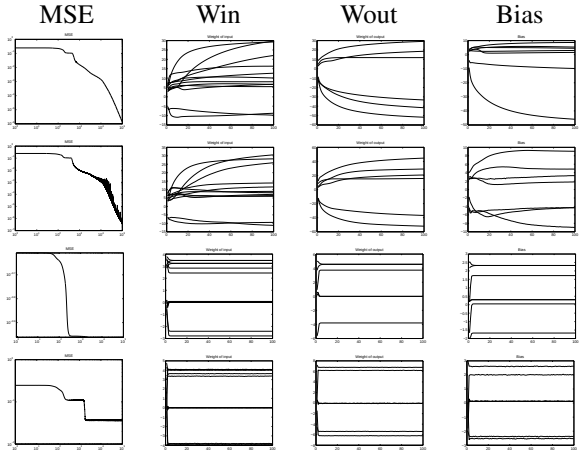

Figure 20. Change of the MSE and weights against time for the XOR problem with additive weight noise injection. From top to bottom, the parameters (α, S_b) are $(0, 0)$, $(0, 10^{-2})$, $(10^{-3}, 0)$ and $(10^{-3}, 10^{-2})$.

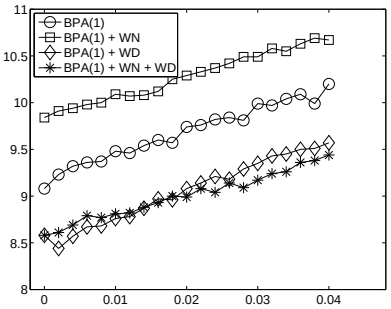

Figure 21. Classification error of Handwritten recognition-MWN.

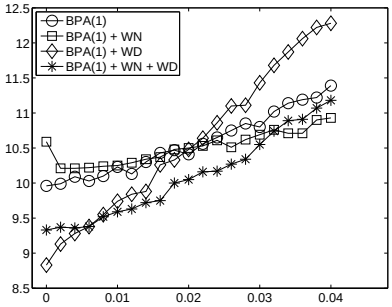

Figure 22. Classification error of Handwritten recognition-AWN.

algorithms combining weight noise injection and weight decay exhibit better convergence behaviors than their pure weight noise injection counterparts. For multiplicative weight noise injection-based algorithms, the benefit of adding weight decay during training is much clear. Adding weight decay during training is able to alleviate the divergence effect due to multiplicative weight noise injection. Finally, our simulation results have shown that **the neural networks attained by the algorithms combining weight noise injection and weight decay could have better fault tolerance abilities than the neural networks attained by the pure weight noise injection-based algorithms**.

ACKNOWLEDGEMENT

The research work reported in this paper is supported in part by Taiwan National Science Council (NSC) Research Grant 97-2221-E-005-050 and 98-2221-E-005-048.

REFERENCES

[1] An G. The effects of adding noise during backpropagation training on a generalization performance, *Neural Computation*, Vol.8, 643-674, 1996.

[2] Basalyga G. and E. Salinas, When response variability increases neural network robustness to synaptic noise, *Neural Computation*, Vol.18, 1349-1379, 2006.

[3] Bernier J.L. *et al*, Obtaining fault tolerance multilayer perceptrons using an explicit regularization, *Neural Processing Letters*, Vol.12, 107-113, 2000.

[4] Bolt G., *Fault tolerant in multi-layer Perceptrons*. PhD Thesis, University of York, UK, 1992.

[5] M. Buscema, Metanet: The theory of independent judges, *Substance Use and Misuse*, Vol.33(2), 439-461, Jan. 1998.

[6] Cavalieri S. and O. Mirabella, A novel learning algorithm which improves the partial fault tolerance of multilayer NNs, *Neural Networks*, Vol.12, 91-106, 1999.

[7] S. Chen, Local regularization assisted orthogonal least squares regression, *Neurocomputing*, pp. 559–585, 2006.

[8] Chiu C.T. *et al.*, Modifying training algorithms for improved fault tolerance, ICNN'94 Vol.I, 333-338, 1994.

[9] Deodhare D., M. Vidyasagar and S. Sathiya Keerthi, Synthesis of fault-tolerant feedforward neural networks using minimax optimization, *IEEE Transactions on Neural Networks*, Vol.9(5), 891-900, 1998.

[10] Edwards P.J. and A.F. Murray, Can deterministic penalty terms model the effects of synaptic weight noise on network fault-tolerance? *International Journal of Neural Systems*, 6(4):401-16, 1995.

[11] Edwards P.J. and A.F. Murray, Fault tolerant via weight noise in analog VLSI implementations of MLP's – A case study with EPSILON, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol.45, No.9, p.1255-1262, Sep 1998.

[12] Hammadi N.C. and I. Hideo, A learning algorithm for fault tolerant feedforward neural networks, *IEICE Transactions on Information & Systems*, Vol. E80-D, No.1, 1997.

[13] Ho K., C.S. Leung, and J. Sum, On weight-noise-injection training, M.Koeppen, N.Kasabov and G.Coghill (Eds.), *Advances in Neuro-Information Processing*, Springer LNCS 5507, pp. 919V926, 2009.

[14] Kevin Ho and John Sum, Note on Weight Noise Injection During Training a MLP, in *Proc. TAAI'2009*, Taichung, 2009.

[15] Ho K., C.S. Leung, J. Sum, Analysis on the Convergence and Objective Functions of Some Fault/Noise Injection-Based Online Learning Algorithms for RBF Networks. *IEEE Transactions on Neural Networks*, Vol.21(6), 938-947, June, 2010.

[16] Ho K., C.S. Leung, J. Sum, Objective functions of online weight noise injection training algorithms for MLP, in submission.

[17] Jim K.C., C.L. Giles and B.G. Horne, An analysis of noise in recurrent neural networks: Convergence and generalization, *IEEE Transactions on Neural Networks*, Vol.7, 1424-1438, 1996.

[18] Kamiura N., *et al*, On a weight limit approach for enhancing fault tolerance of feedforward neural networks, *IEICE Transactions on Information & Systems*, Vol. E83-D, No.11, 2000.

[19] Leung C.S., J. Sum, A fault tolerant regularizer for RBF networks, *IEEE Transactions on Neural Networks*, Vol. 19 (3), pp.493-507, 2008.

[20] Liang Y.L., *Empirical studies on the online learning algorithms based on combining weight noise injection and weight decay*, Master Thesis, Institute of Technology Management, National Chung Hsing University, Taiwan, 2010. (web.nchu.edu.tw/∼pfsum/papers/Liang-WNI-Thesis-20100707.pdf)

[21] Ljung L., Analysis of recursive stochastic algorithms, *IEEE Transactions on Automatic Control*, Vol.AC-22, No.4, pp.551-575, August, 1977.

[22] Murray A.F. and P.J. Edwards, Synaptic weight noise during multilayer perceptron training: fault tolerance and training improvements, *IEEE Transactions on Neural Networks*, Vol.4(4), 722-725, 1993.

[23] Murray A.F. and P.J. Edwards, Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training, *IEEE Transactions on Neural Networks*, Vol.5(5), 792-802, 1994.

[24] Neti C. M.H. Schneider and E.D. Young, Maximally fault tolerance neural networks, *IEEE Transactions on Neural Networks*, Vol.3(1), 14-23, 1992.

[25] Phatak D.S. and I. Koren, Complete and partial fault tolerance of feedforward neural nets., *IEEE Transactions on Neural Networks*, Vol.6, 446-456, 1995.

[26] Sequin C.H. and R.D. Clay, Fault tolerance in feedforward artificial neural networks, *Neural Networks*, Vol.4, 111-141, 1991.

[27] Sum J., K. Ho, SNIWD: Simultaneous weight/node noise injection with weight decay for MLP training, *Proc. ICONIP'2009*.

[28] Sum J., C.S. Leung and K. Ho, On objective function, regularizer and prediction error of a learning algorithm for dealing with multiplicative weight noise, *IEEE Transactions on Neural Networks* Vol.20(1), Jan, 2009.