Misconception on the Regularization Effect of Noise or Fault Injection : Empirical Evidence

John Sum
Institute of Technology Management, National Chung Hsing Univesity
Taichung 40227, Taiwan {pfsum@nchu.edu.tw}

***Abstract*—Over decades, there is a misconception that the learning objective of adding noise or fault during the gradient descent learning is equivalent the desired measure – the expected mean square error (MSE) of the model with such noise or fault. Subsequently, the desired measure is used to interpret the regularization effect of noise injection. The purpose of this paper, together with an companion paper [1], is to clarify this misconception. It is shown in [1] that equivalency between the learning objective and the desire measure depends on three factor : (i) the model of the neural network, (ii) the noise or fault being injected and (iii) the learning algorithm. This paper presents empirical evidence on two node noise injection during training, one for additive noise and the other for multiplicative noise, to supplement the theoretical analyses presented in [1]. It is found that adding additive (resp. multiplicative) during gradient descent learning is not minimizing the desired measure.**

***Index Terms*—Additive Node Noise, Multiplicative Node Noise, Regularization, Weight Fault.**

I. Introduction

Noise or fault injection is a classical method to improve the generalization of a neural network [2]–[7]. Various researches were then conducted to investigate the effect of such noise/fault on the performance of a neural network [8]–[11]. Learning algorithms were developed to synthesize a neural network that is able to tolerate such noise/fault [12]–[17]. The convergence properties, the learning objective functions and the regularization effects of applying gradient descent learning to train a FNN with such noise/fault were analyzed [18]–[29]. Recently, these ideas have be re-advocated in deep learning. Random node fault (i.e. dropout) [30]–[33], multiplicative node noise [32], [34], gradient noise [35] or input noise [36] is added during training a convolutionary neural network (CNN) or deep neural network (DNN).

A. Misconception

As mentioned in [26], there is a common misconception on the regularization effect of noise injection. It is confused that the objective function being minimized by noise injection-based training (denoted as $\mathcal{L}(\mathbf{w})$) is equivalent to the expected MSE of the model with the same type of noise (denoted as $\mathcal{J}(\mathbf{w})$), i.e. $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$. Accordingly, $\mathcal{J}(\mathbf{w})$ is used to interpret the regularization effect of noise injection-based training [34], [37], [38].

This misconception could be due to early analytical works on noise injection training. For the case of adding input noise, it was shown in [8], [9], [11], [18] that the objective function of training with input noise is given by $\mathcal{L}(\mathbf{w}) = V(\mathbf{w}) + \frac{S_I}{2}\sum_i \frac{\partial^2 V(\mathbf{w})}{\partial x_i^2}$, where $V(\mathbf{w})$ is the MSE of a noise-free model. The second term is the Tikhonov regularizer [18]. Happen to be, by taking expectation of $V(\mathbf{w})$ over the probability space of the input noise, the desired measure is given by $\mathcal{J}(\mathbf{w}) = E[V(\mathbf{w})|\mathcal{D}] = V(\mathbf{w}) + \frac{S_I}{2}\sum_i \frac{\partial^2 V(\mathbf{w})}{\partial x_i^2}$. Hence, $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$. The objective function of adding input noise during gradient descent learning is equivalent to the expected MSE of the model with input noise. Furthermore, the same conclusion is made for the case of additive weight noise [9], [26].

These two equivalency results, one for input noise and the other for additive weight noise, suggest that noise injection could be a cheap trick for implementing regularization. Suppose the noise variance S_I is known, minimizing $\mathcal{J}(\mathbf{w})$ by gradient descent, one needs to solve the following recursive equation :

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \mu_t \left\{ \frac{\partial V(\mathbf{w})}{\partial \mathbf{w}} + \frac{S_I}{2}\sum_i \frac{\partial}{\partial \mathbf{w}} \frac{\partial^2 V(\mathbf{w})}{\partial x_i^2} \right\}.$$

Clearly, the computation of the last term is far more expensive than adding input noise during training.

In sequel, researchers started to confuse that the equivalency property could be applied to other noise models, [37, Section 7.5] and [38, Section 7.4.3]. Noise injection is a computationally cheap trick for minimizing $\mathcal{J}(\mathbf{w})$. On the contrary, the regularization effect of noise injection, like multiplicative Gaussian node noise [34], could be interpreted from $\mathcal{J}(\mathbf{w})$. In fact, it is not always true. Whether $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$ depends on three factors: (i) the model of neural network, (ii) the noise or fault model and (iii) the learning algorithm, as depicted in Table I.

B. Goal of the Paper

The only goal of this paper and the companion paper [1] is to clarify this misconception. This paper presents empirical evidence on two node noise injection during training, one for additive noise and the other for multiplicative noise, to supplement the theoretical analyses presented in [1].

II. Node Noise Injection [1]

Here, we summarize the theoretical results on node noise injection. Reader could refer to [1] for detail derivation.

For a neural network with input $\mathbf{x}$, L hidden layers and one output layer, the output of the network could be defined

The work is supported in part by grants from Taiwan MOST 105-2221-E-005-065-MY2 and 108-2221-E-005-036.

TABLE I
$\mathcal{L}(\mathbf{w})$ VERSUS $\mathcal{J}(\mathbf{w})$

Noise/Fault	NN Model	Learning	Equivalency	Ref.
Input Noise	MLP	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[2], [8], [9], [11], [18]
Random Weight Fault	MLP	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[1]
Additive Weight Noise	MLP	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[9], [24], [26], [27]
Multiplicative Weight Noise	MLP	GD	$\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$	[24], [26], [27]
Random Node Fault	MLP	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[25]
Additive Node Noise	MLP	GD	$\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$	[1]
Multiplicative Node Noise	MLP	GD	$\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$	[1], [29]
Input Noise	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[23]
Random Weight Fault	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[1]
Additive Weight Noise	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[23]
Multiplicative Weight Noise	RBF	GD	$\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$	[23]
Random Node Fault	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[20], [21]
Additive Node Noise	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[1]
Multiplicative Node Noise	RBF	GD	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[1]
Additive Weight Noise	BM	BL	$\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$	[28]

MLP: Multilayer perceptron; RBF: Radial basis function network
BM: Boltzmann machine; GD: Gradient descent; BL: Boltzmann learning

as follows : $\mathbf{f} = \mathbf{h}(\mathbf{z}^L, \mathbf{w}^L)$, $\mathbf{z}^l = \mathbf{h}(\mathbf{z}^{l-1}, \mathbf{w}^l)$ and $\mathbf{z}^1 = \mathbf{h}(\mathbf{x}, \mathbf{w}^1)$ for $l = 2, \cdots, L$, where $\mathbf{z}^l$ and $\mathbf{w}^l$ are respectively the node vector and the weight vector of the l^{th} hidden layer. Besides, $\mathbf{z}^1 = \mathbf{h}(\mathbf{x}, \mathbf{w}^1)$ The element of the vector function $\mathbf{h}(\cdot)$ is nonlinear transfer function. Without loss of generality, we consider that there is one output node. Then, for simplicity, we let $f(\mathbf{x}, \mathbf{z}, \mathbf{w})$ be the model, where $\mathbf{x} \in R^m$ is the input vector, $\mathbf{z} \in R^s$ is the hidden node vector and $\mathbf{w} \in R^n$ is the weight vector, i.e. $\mathbf{z} = (\mathbf{z}^L, \cdots, \mathbf{z}^1)$ and $\mathbf{w} = (\mathbf{z}^L, \cdots, \mathbf{z}^1)$. Thus, $\mathbf{z}$ is a function of $\mathbf{x}$ and $\mathbf{w}$, i.e. $\mathbf{z}(\mathbf{x}, \mathbf{w})$.

Given a set of N samples $\mathcal{D} = \{\mathbf{x}_k, y_k\}_{k=1}^N$, the performance measure is defined as follows :

$$V(\mathbf{z}, \mathbf{w}) = \frac{1}{N} \sum_{k=1}^{N} \ell_k(\mathbf{z}(\mathbf{x}_k, \mathbf{w}), \mathbf{w}), \quad (1)$$

where $\ell_k(\mathbf{z}(\mathbf{x}_k, \mathbf{w}), \mathbf{w})$ is the measure of the network on the k^{th} sample. The desired measure is defined as follows :

$$\mathcal{J}(\mathbf{w}) = E\left[V(\mathbf{z}, \mathbf{w})\right] = \frac{1}{N} \sum_{k=1}^{N} E\left[\ell_k(\mathbf{z}(\mathbf{x}_k, \mathbf{w}), \mathbf{w})\right]. \quad (2)$$

The expectation is taken over the space of $\mathbf{b}_A$ or $\mathbf{b}_M$.

With *node noise,* the weight update is given by

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \mu_t \frac{\partial \ell_t(\tilde{\mathbf{z}}(\mathbf{x}_t, \mathbf{w}(t-1)), \mathbf{w}(t-1))}{\partial \mathbf{w}}, \quad (3)$$

$\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{b}_A$ for additive noise, $\tilde{\mathbf{z}} = \mathbf{z} + \mathbf{z} \otimes \mathbf{b}_M$ for multiplicative noise. Both $\mathbf{b}_A$ and $\mathbf{b}_M$ are mean zero Gaussian noise vectors.

A. Additive Node Noise

With additive node noise, it can be shown that the desired measure is given by

$$\mathcal{J}_A(\mathbf{w}) = E[V(\mathbf{w})] = V(\mathbf{w}) + \frac{S_A}{2} \sum_j \frac{\partial^2 V(\mathbf{w})}{\partial z_j^2} \quad (4)$$

and the objective function of the learning with noise injected is given by

$$\mathcal{L}_A(\mathbf{w}) = V(\mathbf{w}) + \frac{S_A}{2} \sum_i \sum_j \int \frac{\partial^3 V(\mathbf{w})}{\partial z_j^2 \partial w_i} dw_i. \quad (5)$$

Unfortunately, there is no simple close form for (7) expect in some special cases. Thus, it is not easy to interpret the regularization effect multiplicative node noise injection.

B. Multiplicative Node Noise

With multiplicative node noise, the desired measure is given by

$$\mathcal{J}_M(\mathbf{w}) = V(\mathbf{w}) + \frac{S_M}{2N} \sum_{k,j} z_j^2(\mathbf{x}_k, \mathbf{w}) \frac{\partial^2 \ell_k(\mathbf{w})}{\partial z_j^2} \quad (6)$$

and the objective function of the learning with noise injected is given by

$$\begin{aligned} \mathcal{L}_M(\mathbf{w}) &= V(\mathbf{w}) \\ &+ \frac{S_M}{2N} \sum_i \sum_{k,j} \int z_j^2(\mathbf{x}_k, \mathbf{w}) \frac{\partial^3 \ell_k(\mathbf{w})}{\partial z_j^2 \partial w_i} dw_i. \end{aligned} \quad (7)$$

Again, there is no simple close form for (7) expect in some special cases. Thus, it is not easy to interpret the regularization effect multiplicative node noise injection.

III. Empirical Evidence

From (4), (5), (6) and (7), it is clear that $\mathcal{L}_A(\mathbf{w}) \neq \mathcal{J}_A(\mathbf{w})$ and $\mathcal{L}_M(\mathbf{w}) \neq \mathcal{J}_M(\mathbf{w})$. Equivalently, $\mathbf{w}_{\mathcal{L}} \neq \mathbf{w}_{\mathcal{J}}$. To investigate this issue empirically, two methods could be applied. The first one is exhaustive search. However, it is not suitable for large scale network, like deep neural network. We rely on the second method – applying learning curve.

A. Method: Use of Learning Curve

Learning curve is used to illustrate the problem of overfitting [37], [39], [40]. In the course of learning, the testing MSE first decreases together with the training MSE. After some epoches, the testing MSE starts to increase while the training MSE keeps on decreasing until the training is complete.

Here, we use the training MSE to identify if $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$, as $\mathcal{J}(\mathbf{w})$ is essentially the training MSE. These data can easily be collected during training. Using learning curve, three situations will be observed.

- Case 1: $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$. In this case, the trained model $\mathbf{w}_{\mathcal{L}}$ is exactly the same as the desired model $\mathbf{w}_{\mathcal{J}}$, as shown in Figure 1(a). Thus, it is anticipated that $\mathbf{w}$ moves along the path as shown in the figure and eventually reaches the location $\mathbf{w}_{\mathcal{J}}$. The learning curve is a decreasing curve.
- Case 2: $\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$. In this case, the trained model is not the same as the desired model. The trained model $\mathbf{w}_{\mathcal{L}}$ could be located further away from the origin, as shown in Figure 1(b), or located closer to origin Figure 1(c).
  - Case 2(i): For the former case, $\mathbf{w}$ first moves towards $\mathbf{w}_{\mathcal{J}}$. Then, it moves away from $\mathbf{w}_{\mathcal{J}}$ and eventually arrives $\mathbf{w}_{\mathcal{L}}$. So, the learning curve first decreases and then increases after certain epoches.
  - Case 2(ii): For the latter case, $\mathbf{w}$ will not pass through $\mathbf{w}_{\mathcal{J}}$. So, the learning curve is a decreasing curve as shown in the panel of Figure 1(c).

If the learning curve exhibits the shape like Case 2(i), we can thus conclude that $\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$. If the learning curve exhibits the shape like Case 1 or Case 2(ii), no conclusion can be made.

B. Data

To validate the theoretical results obtained in the previous section, the MNIST handwritten digit dataset was downloaded[1]. To convert the dataset in a form that the MATLAB is able to load, two helper functions are used[2]. MNIST dataset consists of ten classes of handwritten digits, from 0 to 9, in the form of images. Each digit image is of size 28×28 pixels. In the dataset, there are 60,000 training images and 10,000 testing images.

C. Network and Noise Models

A MLP of two hidden layers is examined. Each hidden layer consists of 100 hidden nodes and 10 output nodes. For simplicity, this structure is denoted as 784-100-100-10. This model has 89,610 parameters, including weights and biases. The transfer function of both the hidden nodes and the output nodes is defined as a sigmoid function. Gradient descent is applied and the step size is set to 0.1.

In each step, node noise is injected. For the variance of additive noise 0.04 and the variance of multiplicative noise is 0.25. A training sample is selected sequentially from the training set. A testing sample is selected randomly from the testing set. Then, the square error of the model on the training sample and the square error of the model on the testing sample are evaluated. Finally, the weights are updated by gradient descent. After $60,000$ steps (i.e. one epoch), the mean training error and the mean testing error are calculated.

[1] From http://yann.lecun.com/exdb/mnist/.

[2] Downloaded from http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset.

D. Results

The result on additive node noise (resp. multiplicative node noise) is shown in Figure 2(a) (resp. Figure 2(b)). For both cases, the training MSE first decreases and then after certain epoch (around 500 for the case of additive noise and around 200 for the case of multiplicative noise), the training MSE increases. The outcome is exactly the same as what we have anticipated, see Figure 1(b). Thus, we can conclude that the objective function of adding node noise during gradient descent learning is not the desired measure.

IV. Conclusions

In this paper and the companion paper [1], it has been clarified a misconception on the the regularization effect of noise injection. The objective function of adding node noise during gradient descent learning $\mathcal{L}(\mathbf{w})$ is not the desire measure of the model with such noise $\mathcal{J}(\mathbf{w})$. Thus, the regularization effect of noise injection could not simply be interpreted from $\mathcal{J}(\mathbf{w})$. In sequel, training with noise might not able to generate a noise-robust model. It is better said that gradient descent learning might not be able to train a neural network to the desired model if noise exists. The works presented in this paper and in [1], [27]–[29] are still preliminary. A lot more works have to be done in the future. One problem is to search or develop a learning algorithm that is able to train a neural network to the desired model even if noise exists.

References

[1] J. Sum, "Misconception on the regularization effect of noise or fault injection : Theoretical analysis," 2019, in submission.

[2] K. Matsuoka, "Noise injection into inputs in back-propagation learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 3, pp. 436–440, May 1992.

[3] A. Murray, "Analogue noise-enhanced learning in neural network circuits," *Electronics Letters*, vol. 27, no. 17, pp. 1546–1548, 1991.

[4] ——, "Multilayer perceptron learning optimized for on-chip implementation: A noise-robust system," *Neural Computation*, vol. 4, no. 3, pp. 366–381, 1992.

[5] J. S. Judd and P. W. Munro, "Nets with unreliable hidden nodes learn error-correcting codes," in *Advances in Neural Information Processing Systems 5*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 89–96.

[6] A. Murray and P. Edwards, "Synaptic weight noise during multilayer perceptron training: fault tolerance and training improvements," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 722–725, 1993.

[7] ——, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 792–802, 1994.

[8] Y. Grandvalet and S. Canu, "Comments on "Noise injection into inputs in back propagation learning"," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 4, pp. 678–681, 1995.

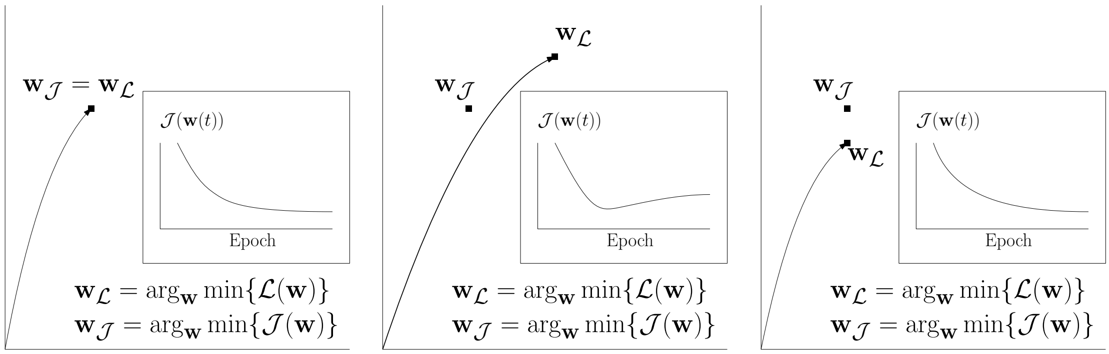
[9] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Computation*, vol. 8, pp. 643–674, 1996.

[10] K. Jim, C. Giles, and B. Horne, "An analysis of noise in recurrent neural networks: Convergence and generalization," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1424–1438, 1996.

[11] Y. Grandvalet, S. Canu, and S. Boucheron, "Noise injection: Theoretical prospects," *Neural Computation*, vol. 9, no. 5, pp. 1093–1108, 1997.

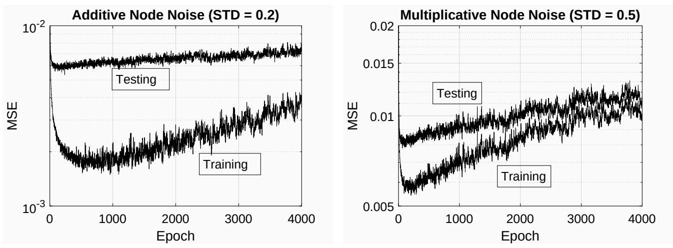
[12] C. Sequin and R. Clay, "Fault tolerance in feedforward artificial neural networks," *Neural Networks*, vol. 4, pp. 111–141, 1991.

[13] G. Bolt, "Fault tolerant in multi-layer perceptrons," Ph.D. dissertation, University of York, UK, 1992.



(a) $\mathcal{L}(\mathbf{w}) = \mathcal{J}(\mathbf{w})$ (b) $\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$ (c) $\mathcal{L}(\mathbf{w}) \neq \mathcal{J}(\mathbf{w})$

Fig. 1. The learning curves (training MSE curves) of three different situations.



(a) ANN, $S_A = 0.04$ (b) MNN, $S_M = 0.25$

Fig. 2. Simulation results on training a MLP (784-100-100-10) with node noise to fit the MNIST dataset.

[14] J. L. Bernier, J. Ortega, E. Ros, I. Rojas, and A. Prieto, "A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs," *Neural Computation*, vol. 12, no. 12, pp. 2941–2964, 2000.

[15] J. L. Bernier, J. Ortega, I. Rojas, and A. Prieto, "Improving the tolerance of multilayer perceptrons by minimizing the statistical sensitivity to weight deviations," *Neurocomputing*, vol. 31, no. 1-4, pp. 87–103, 2000.

[16] J. L. Bernier, A. F. Díaz, F. Fernández, A. Cañas, J. González, P. Martín-Smith, and J. Ortega, "Assessing the noise immunity and generalization of radial basis function networks," *Neural Processing Letters*, vol. 18, no. 1, pp. 35–48, 2003.

[17] C. Leung and J. Sum, "A fault tolerant regularizer for RBF networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 493–507, 2008.

[18] C. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, vol. 7, pp. 108–116, 1995.

[19] R. Reed, R. M. II, and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 529–538, 1995.

[20] J. Sum, "On a multiple nodes fault tolerant training for RBF: Objective function, sensitivity analysis and relation to generalization," in *Proceedings of TAAI'05, Tainan, Taiwan*, 2005.

[21] J. Sum, C.-S. Leung, and K. Ho, "On node-fault-injection training of an RBF network," in *International Conference on Neural Information Processing*. Springer, 2008, pp. 324–331.

[22] K. Ho, C.-S. Leung, and J. Sum, "On weight-noise-injection training," in *International Conference on Neural Information Processing*. Springer, 2008, pp. 919–926.

[23] K. Ho, C. Leung, and J. Sum, "Convergence and objective functions of some fault/noise injection-based online learning algorithms for RBF networks," *IEEE Transactions on Neural Networks*, vol. 21, no. 6, pp. 938–947, June 2010.

[24] ——, "Objective functions of the online weight noise injection training algorithms for MLP," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 317–323, Feb 2011.

[25] J. Sum, C.-S. Leung, and K. Ho, "Convergence analysis of on-line node fault injection-based training algorithms for MLP networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 211–222, Feb 2012.

[26] ——, "Convergence analyses on on-line weight noise injection-based training algorithms for MLPs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 11, pp. 1827–1840, Nov 2012.

[27] ——, "A limitation of gradient descent learning," 2019, accepted for publication in *IEEE Transactions on Neural Networks and Learning Systems*.

[28] J. Sum and C.-S. Leung, "Learning algorithm for Boltzmann machines with additive weight and bias noise," *IEEE Transactions on Neural Networks and Learning Systems*, 2019, accepted for publication.

[29] ——, "Analysis on dropout regularization," 2019, in submission.

[30] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[31] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[33] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," in *Advances in Neural Information Processing Systems*, 2017, pp. 5109–5118.

[34] E. Nalisnick, A. Anandkumar, and P. Smyth, "A scale mixture perspective of multiplicative noise in neural networks," *arXiv preprint arXiv:1506.03208*, 2015.

[35] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv preprint arXiv:1511.06807*, 2015.

[36] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Networks*, vol. 78, pp. 15–23, 2016.

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[38] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: Tutorial," *arXiv preprint arXiv:1905.12787*, 2019.

[39] C. M. Bishop, *Pattern Recognition and Machine Learning*. springer, 2006.

[40] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2009.