Whiteout: Gaussian Adaptive Regularization Noise in Deep Neural Networks

Yinan Li[1*], Ruoyi Xu[2*], and Fang Liu[1†]

[1] Department of Applied and Computational Mathematics and Statistics

University of Notre Dame, Notre Dame, IN 46556, U.S.A.

[2] School of Computer Science and Technology,

University of Science and Technology of China, Hefei, 230027, P.R.China

Dec 3, 2016

Abstract

Noise injection is an off-the-shelf method to mitigate over-fitting in neural networks (NNs). The recent developments in Bernoulli noise injection as implemented in the dropout and shakeout procedures demonstrates the efficiency and feasibility of noise injection in regularizing deep NNs. We propose whiteout, a new regularization technique via injection of adaptive Gaussian noises into a deep NN. Whiteout offers three tuning parameters, offering flexibility during training of NNs. We show that whiteout is associated with a deterministic optimization objective function in the context of generalized linear models with a closed-form penalty term and includes lasso, ridge regression, adaptive lasso, and elastic net as special cases. We also demonstrate that whiteout can be viewed as robust learning of NN model in the presence of small and insignificant perturbations in input and hidden nodes. Compared to dropout, whiteout has better performance when training data of relatively small sizes with the sparsity introduced through the l_1 regularization. Compared to shakeout, the penalized objective function in whiteout has better convergence behaviors and has a tighter bound for tail probabilities. We establish theoretically that the noise-perturbed empirical loss function with whiteout converges almost surely to the ideal loss function, and the estimates of NN parameters obtained from minimizing the former loss function are consistent with those obtained from minimizing the ideal loss function. Computationally, whiteout can be incorporated in the back-propagation algorithm and is computationally efficient. The superiority of whiteout over dropout and shakeout in training NNs in classification is demonstrated using the MNIST data.

keywords: (adaptive) lasso; elastic net; regularization; robustness; consistency; backpropagation

*Co-first authors

†Corresponding author email: fang.liu.131@nd.ed

1 Introduction

An artificial neural network (NN) is a simplified computational model of how the neurons in our brains operate to solve certain kinds of problems (Mcculloch and Pitts, 1943). A NN comprises of an input layer, an output layer, and one or more hidden layers in between with nonlinear processing units for feature extraction and transformation. When there are more than one hidden layer in a NN, the NN is often referred to as a deep NN. Deep learning is often regarded as a re-branding of NNs with a class of machine learning algorithms (Hinton and Salakhutdinov, 2006). In the deep learning of a NN, each successive layer uses the output from the previous layer as input, representing a higher level set features derived from the lower-level features in the previous layer, and thus formulating a hierarchical structure of feature extraction. The work by Hinton and Salakhutdinov (2006) marked a reactivation in the research and applications of deep learning of NN, which has seen many successful applications in pattern recognition and classification such as image an speech recognition, natural language processing, drug discovery and toxicology, among others.

Deep NNs are prone to overfitting given the high-dimensionality of the parameters involved in the multiplicity of layers and the often large number of nodes. Regularization methods such as early stopping, max-norm, Ivakhnenko's unit pruning Ivakhnenko (1971), l_2-regularization (weight decay) and l_1-regularization (sparsity) can be used to impose smoothness constraints on the learned NN model to help combat overfitting. A recent regularization method is dropout (Hinton et al., 2012; Srivastava et al., 2014), where some number of units are randomly dropped from the hidden layers (with recommended proportion 0.5) and the input layer (with recommended proportion 0.2) during the training period of a deep NN. The dropout regularization can be viewed a stochastic version of model averaging and prevents the nodes from co-adapting too much. As such, dropout can significantly reduce the generalization errors in a trained deep NN. From an implementation perspective, dropout is noise injection (NI), by adding Bernoulli noise to hidden and inputs units in a NN (Wager et al., 2013; Wang and Manning, 2013a; Srivastava et al., 2014). In the simple case of linear regression models $\mathbf{E}(\mathbf{y}) = \mathbf{X}\mathbf{w}$, dropping a predictor $\mathbf{x}_j$ from the model dropped out with a probability $1-p$ is equivalent, to the following Tikhonov regularization in expectation: minimization of $||\mathbf{y}-p\mathbf{X}\mathbf{w}||^2+p(1-p)||\Gamma\mathbf{w}$, where $\Gamma = (\text{diag}(\mathbf{X}^t\mathbf{X}))^{1/2}$. In general, NI helps to improve the generalization ability of a trained NN, especially in fully connected NN layers (Zhuo et al., 2015; Grandvalet, 2000; An, 1996), , and has been shown to be related to kernel smoothing (Holmstrom and Koistinen, 1992) and heat equation (Grandvalet, 2000).

Due to the optimistic performance of dropout, various algorithmic and computational extensions and improvements have been proposed. The maxout technique is designed to facilitate optimization by dropout and improve the accuracy of dropout with a new activation function (Goodfellow et al., 2013). Fast dropout speeds up the dropout algorithm by sampling from or integrating a Gaussian approximation instead of randomly dropping nodes (Wang and Manning, 2013b). DropConnect sets a random subset of weights in a NN to zero (Wan et al., 2013). Standout or adaptive dropout selectively (rather than randomly) sets nodes to zero and chooses to enhance or inverse the contributions of a node to the next layer (Ba and Frey, 2013). In all these regularization techniques (dropout, DropConnect,

standout, shakeout), noises being injected to a deep NN are Bernoulli noises. Shakeout also leads to a sparse model and has better performance when the training data set is not large in size given that the Bernoulli noises in shakeout are designed to allow both l_1 and l_2 regularization (elastic-net type of regularization) on the weights of a NN.

On the other hand, there is still room to improve on the current available set of regularization techniques through Bernoulli NI. First, the final NN trained with Bernoulli NI is an average of a finite number of sub-models that may not exhibit concentration patterns in sub-models in large p-small n settings. Therefore, the "average" final trained NN can be biased. Second, in shakeout, updates of weights through the backpropagation (BP) algorithm includes derivative of the sign function which can only be approximated at best. Third, due to the restriction of Bernoulli's noise (two possible noise values), dropout corresponds to the l_2 regularization and the elastic-net-type regularization is the only form of extension from the l_2 regularization in shakeout. Srivastava et al. (2014) hinted that Gaussian noise works just as well as, or perhaps better than, Bernoulli noise.

In this paper, we propose *whiteout*, a *Gaussian adaptive regularization NI* technique, to regularize learning of deep NNs. whiteout is named so because it is a NI technique as dropout, and the injected Gaussian noise is a "white" noise. whiteout perturbs the input and hidden nodes in a deep NN with Gaussian noise, the variance of which adapts to the weights during the iterations of a deep learning algorithm. whiteout has a corresponding deterministic optimization objective function with a closed-from penalty term that includes lasso, ridge regression, elastic net, adaptive lasso, and adaptive elastic net as special cases. The involvement of the l_1 penalty enable whiteout to train a NN model in data sets of relatively small sizes. whiteout offers 3 tuning parameters, and is more flexible in terms of training than the above mentioned Bernoullis NI injection. In addition to being reviewed as regularization technique to combat overfitting, whiteout can also be regarded as a technique to improve robustness of the learned model to small and insignificant perturbations in data. We also show that the empirical loss function with whiteout is consistent for the ideal loss function and has a tighter bound, and empirically has a higher convergence rate than those of Bernoulli noise based methods. Computationally, whiteout can be easily incorporated in the back-propagation procedure and is computationally efficient.

The rest of the paper is organized as follows. The concept of whiteout is introduced in Section 2. The connection between whiteout and regularization is established in Section 2.1. whiteout as a way to improve the robustness of a trained NN is presented in Section 2.2. We define multiple types of loss function in NNs in Section 2.3, and prove that the noise-perturbed empirical loss function with whiteout converges almost surely to the ideal loss function. Section 2.4 presents the step-by-step whiteout-augmented BP algorithm and proposes a weights updating algorithm without having to approximate the derivative of the sign function (an improvement over the shakeout algorithm). The whiteout technique is applied to the classification problem in the MNIST data, and compared to the regular BP algorithm, dropout and shakeout in Section 3. Concluding remarks are presented in Section 4. Most of the technical proofs are provided in the Appendix.

2 Whiteout

Whiteout adds independent noises drawn from the Gaussian distribution in Eqn (1) to each node $i = 1, \ldots, p^{(l)}$ in layer l of a NN in each case of the training data in an epoch cycle of the deep learning algorithm,

$$\tilde{\mathbf{X}}_i^{(l)} = \mathbf{X}_i^{(l)} + \mathbf{e}_i = \begin{pmatrix} X_{i1}^{(l)} \\ \vdots \\ X_{ip^{(l)}}^{(l)} \end{pmatrix} + \begin{pmatrix} e_{i1} \\ \vdots \\ e_{ip^{(l)}} \end{pmatrix}, \text{ where } e_{ij} \overset{\text{iid}}{\sim} N\left(0, \frac{\sigma^2}{|w_{ij}^{(l+1)}|^\gamma} + \lambda\right), \qquad (1)$$

where j is a node in layer $l+1$, $w_{ij}^{(l+1)}$ is the weight between nodes i and j, $\sigma^2 > 0$, $\lambda \geq 0$, and $\gamma \in [0, 2]$ are tuning parameters. In the practical implementation of whiteout, all three tuning parameters can be user-specified or chosen by cross validation (CV). Intuitively, σ should be inversely proportional to n; that is, given the same NN model, overfitting is less a serious problem with a large n thus less regularization is required. Whiteout can be regarded as a process of generating noisy versions of the training data set, and the auxiliary noise is then averaged out in the final trained NN (Seghouane et al., 2004). Some convenient choices and special cases of whiteout noise are Gaussian lasso noise (gala), Gaussian adaptive lasso noise (gaala), Gaussian ridge noise (gar), Gaussian elastic-net noise (gen), and Gaussian group noise (gag) presented in Definition 1 (for notation simplicity, w_{ij} is used in place of $w_{ij}^{(l+1)}$).

Definition 1. In whiteout,

a) gala draws Gaussian noise from $e_{ij} \sim N\left(0, \frac{\sigma^2}{|w_{ij}|}\right)$

b) gaala draws noises from $e_{ij} \sim N\left(0, \frac{\sigma^2}{|w_{ij}|^\gamma}\right), 1 < \gamma < 2$

c) gar draws Gaussian noise from $e_{ij} \sim N\left(0, \frac{\sigma^2}{|w_{ij}|^2}\right)$.

d) gala, gaala and gar noises are collectively referred to as *gallr* noise, covering the scenarios when $\gamma \in [1, 2]$ and $\lambda = 0$.

e) gen draws noises from $e_{ij} \sim N\left(0, \frac{\sigma^2}{|w_{ij}|} + \lambda\right), \lambda > 0$

f) gag penalizes a group of input variables $(X_1, \cdots, X_g) \in \mathcal{G}$ (e.g.,indicator variables generated from the same categorical attribute) rather than singly, by drawing noises from $e_{ij} \sim N\left(0, \sigma^2 \frac{C\sqrt{\sum_{k=1}^g w_{ik}^2}}{w_{ij}^2}\right), j = 1, \cdots, g$, whereas $e_{ij} \sim N\left(0, \frac{\sigma^2}{|w_{ij}|}\right)$ for $X_j \notin \mathcal{G}$, where C is the scale constant concerning the group structure.

Noise e_{ij} can be injected to the nodes in both the input and the hidden layers in gallr and gen, whereas gag makes the most sense in perturbing the nodes in the input layer, since grouping of hidden nodes, which are abstract extracted features that do not necessarily have any physical meanings, are hard to justify.

Remark 1. Whiteout with gen noise is equivalent to adding the sum of gala noise and the regular Gaussian noise with a constant variance σ^2. Therefore, all theoretical properties established in the framework gallr noise (Sections 2.1, 2.2, 2.3) also hold in case of gen noise.

Injecting additive Gaussian noise of mean = 0 as given in Eqn (1) is equivalent to injecting multiplicative Gaussian noise to node h as in $h_i + h_i\epsilon_{ij}$, where $\epsilon_{ij} \sim N\left(1, \frac{\sigma^2}{h_i^2|w_{ij}^{(l+1)}|^\gamma} + \frac{\lambda}{h_i^2}\right)$.

Remark 2. The minor difference between additive and multiplicative Gaussian NI does not affect the theoretical properties concerning whiteout (Sections 2.1, 2.2, 2.3) given the inherent properties of activation functions and the structures of deep NNs.

Without loss of generality, we will establish the theoretical properties of whiteout in the framework of additive gallr noises, which will be extended to other noises types and to the multiplicative noise case, given the statements in Remarks 1 and 2.

2.1 Whiteout as a regularization method

A common framework where NI as a regularization technique is established is the the exponential family, where generalized linear models (GLMs) are based on (Bishop, 1995; An, 1996; Kang et al., 2016; Wager et al., 2013). In a GLM, the conditional distribution of output Y given input $\mathbf{X} \in \mathcal{R}^p$ and parameters $\mathbf{w}$ is modelled with the exponential family distribution with parameter w that is often expressed as

$$\Pr(Y|\mathbf{X}, \mathbf{w}) = h(Y) \exp\left(T(Y)\mathbf{X}\mathbf{w} - A(\mathbf{X}\mathbf{w})\right),$$

where $\boldsymbol{\eta} = \mathbf{X}\mathbf{w}$ is the natural parameter and $A(\mathbf{X}\mathbf{w})$ is the log-partition function. The corresponding negative log-likelihood function is $l(\mathbf{w}|\mathbf{X}, Y) = -\log(\Pr(Y|\mathbf{X}, \mathbf{w})) = -\mathbf{X}\mathbf{w} + A(\mathbf{X}\mathbf{w})T(Y) + C$. Given training data $(\mathbf{x}_i, y_i)$ for $i = 1, \cdots, n$, where n is the sample size of the training data, the maximum likelihood estimator (MLE) $\hat{\mathbf{w}}$ is

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \textstyle\sum_{i=1}^n l(\mathbf{w}|\mathbf{x}_i, y_i) \tag{2}$$

Whiteout substitutes the observed $\mathbf{x}_i$ with its noisy version $\tilde{\mathbf{x}}_i$ as defined in Eqn (1) in $\sum_{i=1}^n l(\mathbf{w}|\mathbf{x}_i, y_i)$. That is, the noise perturbed version of the negative log-likelihood function is $\sum_{i=1}^n l(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i)$. Minimization of the expectation $\sum_{i=1}^n l(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i)$ over the distribution of noise $\mathbf{e} = (e_1, \ldots, e_n)^t$ leads to

$$\hat{\mathbf{w}}_p = \arg\min_{\mathbf{w}} \textstyle\sum_{i=1}^n \mathrm{E}_{\mathbf{e}}(l(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i)) \tag{3}$$

Lemma 1. The expected negative log-likelihood function over the distribution of noise $\mathbf{e}$ $\mathrm{E}_{\mathbf{e}}(l_p(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i))$ can be expressed as

$$\mathrm{E}_{\mathbf{e}}(l(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i)) = \sum_{i=1}^n l(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i) + R(\mathbf{w}), \tag{4}$$

$$\text{where} \qquad R(\mathbf{w}) \triangleq \sum_{i=1}^{n} \mathrm{E}_{\mathbf{e}}(A(\tilde{\mathbf{x}}_i\mathbf{w})) - A(\mathbf{x}_i\mathbf{w}) \approx \tfrac{1}{2}\sum_{i=1}^{n} A''(\mathbf{x}_i\mathbf{w})\mathrm{Var}(\tilde{\mathbf{x}}_i\mathbf{w}) \tag{5}$$

The proof of Eqns (4) and (5) in Lemma 1 is given in Appendix A. Eqn (4) suggests that $R(\mathbf{w})$ is a regularization term that penalizes the minimization of negative log-likelihood function. In GLMs, $A(\tilde{\mathbf{x}}_i\mathbf{w})$ is a convex and smooth function of $\mathbf{w}$ (Wainwright and Jordan, 2008), and $R(\mathbf{w})$ is always positive by Jensen's inequality (Wager et al., 2013). The actual analytical form of $R(\mathbf{w})$ can be readily derived. For example, in linear regression, $\mathrm{Var}(\tilde{\mathbf{x}}_i\mathbf{w}) = \sigma^2\sum_{j=1}^{p}|\mathbf{w}_j|^{2-r}$ and $A''(\mathbf{x}_i\mathbf{w}) = 1$, thus

$$R(\mathbf{w}) \approx \sigma^2 \sum_{i=1}^{n}\sum_{j=1}^{p} |\mathbf{w}_j|^{2-r} \triangleq \frac{n\sigma^2}{2}\big|\big||\mathbf{w}|^{2-r}\big|\big|_1;$$

in logistic regression, $\mathrm{Var}(\tilde{\mathbf{x}}_i\mathbf{w}) = \sigma^2\sum_{j=1}^{p}|w_j|^{2-r}$ and $A''(\mathbf{x}_i\mathbf{w}) = p_i(1-p_i)$, where $p_i = \Pr(y_i = 1|\mathbf{x}_i) = (1+\exp(-\mathbf{x}_i\mathbf{w}))^{-1}$, thus

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\sum_{i=1}^{n} p_i(1-p_i)\sum_{j=1}^{p}|w_j|^{2-r} \triangleq \frac{\sigma^2}{2}\sum_{i=1}^{n} p_i(1-p_i)\big|\big||\mathbf{w}|^{2-r}\big|\big|_1$$

Theorem 1. In the framework of GLMs, in expectation,

a). the whiteout procedure with additive gallr noise in Definition 1 is approximately equivalent to the adaptive lasso penalization (Zou, 2006), with the regularization term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\mathbf{1}^t\mathbf{\Lambda}(\mathbf{w})\mathbf{1}\big|\big||\mathbf{w}|^{2-r}\big|\big|_1, \tag{6}$$

where $\mathbf{\Lambda}(\mathbf{w}) = \mathrm{diag}(A''(\mathbf{x}_1\mathbf{w}), \cdots, A''(\mathbf{x}_n\mathbf{w}))$, $\mathbf{1}$ is a column vector of dimension n.

b). the whiteout procedure with gen noise in Definition 1 is approximately equivalent to the elastic net penalization (Zou and Hastie, 2005), with the regularization term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\mathbf{1}^t\mathbf{\Lambda}(\mathbf{w})\mathbf{1}||\mathbf{w}||_1 + \frac{\lambda}{2}\mathbf{1}^t\mathbf{\Lambda}(\mathbf{w})\mathbf{1}||\mathbf{w}||_2^2 \tag{7}$$

c). the whiteout procedure with gag noise in Definition 1 is approximately equivalent to the grouped lasso penalization (Yuan and Lin, 2006), with the regularization term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\mathbf{1}^t\mathbf{\Lambda}(\mathbf{w})\mathbf{1}\left(gC||\mathbf{w}_j I(\mathbf{X}_j \in \mathcal{G})||_2 + ||\mathbf{w}_j I(\mathbf{X}_j \notin \mathcal{G}))||_1\right), \tag{8}$$

where I is an indicator function.

The results presented in Theorem 1 with additive noises can be easily extended to the case of multiplicative noises.

Corollary 1. In the framework of GLMs, in expectation,

a). the whiteout procedure with multiplicative gallr noise leads to penalty term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\big|\big|\Gamma(\mathbf{w})|\mathbf{w}|^{2-r}\big|\big|_1, \text{ where } \Gamma(\mathbf{w}) = \mathrm{diag}(\mathbf{X}^{\mathbf{T}}\mathbf{\Lambda}(\mathbf{w})\mathbf{X})) \tag{9}$$

b). the whiteout procedure with multiplicative gen noises leads to penalty term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\big|\big|\Gamma(\mathbf{w})|\mathbf{w}|^2\big|\big|_1 + \frac{\lambda}{2}\big|\big|\Gamma(\mathbf{w})|\mathbf{w}|\big|\big|_1 \tag{10}$$

c). the whiteout procedure with multiplicative gag noises leads to penalty term

$$R(\mathbf{w}) \approx \frac{\sigma^2}{2}\big|\big|C\Gamma_1||\mathbf{w}_j I(\mathbf{X}_j \in \mathcal{G})||_2\big|\big|_1 + \sigma^2\big|\big|\Gamma_2|\mathbf{w}_j I(\mathbf{X}_j \notin \mathcal{G})|\big|\big|_1, \tag{11}$$

where Γ_1 is the subset of $\Gamma(\mathbf{w})$ corresponding to $X_j \in \mathcal{G}$, Γ_2 is the subset of $\Gamma(\mathbf{w})$ corresponding to $X_j \notin \mathcal{G}$.

2.2 Whiteout as a robust learning method

We have shown in Section 2.1 that the whiteout procedure can be regarded as a regularization approach in the maximization of the likelihood function in GLMs. In this section, we will examine the whiteout procedure from the perspective of stabilizing a learned deep NN (robustness to noisy small perturbation, and thus generalization of the learned NN). The mathematical framework, which is established below, can be used to justify NNs of multiple hidden layers. It is motivated by the work in Matsuoka (1992) who examined the stability of a shallow NN with noise injection to the input layer only. We explore the more general case where NI can occur in the input and the hidden layers in deep NNs. For the purposes of presentation clarity, we demonstrate the framework in NNs with a single hidden layer and a single binary output node. The proof can be easily extended to a deep NN with multiple hidden layers and output with multiple outcomes (Remark 3).

Denote the training data by $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ where $\mathbf{x}_i = (x_{i1}, \cdots, x_{ip})$ and $y_i \in \{0, 1\}$ for $i = 1, \ldots, n$, and the hidden nodes by h_{ij} for $j = 1, \ldots, m$. Inputs $\mathbf{x}_i$ and hidden node h_{ij} are connected through the activation function $f_j^{(1)}(\mathbf{x}_i) \triangleq f_j^{(1)}(\mathbf{x}_i\mathbf{w}_j^{(1)} + b^{(1)})$ with weights $\mathbf{w}_j^{(1)} = (w_{j1}^{(1)}, \cdots, w_{jp}^{(1)})^t$ and bias $b^{(1)}$. Denote the injected noises via whiteout to $\mathbf{x}_i$ during the training of $\mathbf{w}_j^{(1)}$ by $\mathbf{e}_{ij}^{(1)} = (e_{ij1}^{(1)}, \cdots, e_{ijp}^{(1)})^t$. Similarly, the hidden and output layers are connected through the activation function $f^{(2)}(\mathbf{h}_i) \triangleq f^{(2)}(\mathbf{h}_i\mathbf{w}^{(2)} + b^{(2)})$ with bias $b^{(2)}$ and weights $\mathbf{w}^{(2)} = (w_1^{(2)}, \cdots, w_m^{(2)})^t$, where $\mathbf{h}_i = (h_{i1}, \ldots, h_{im})$. Denote the noises injected to hidden node $\mathbf{h}_i$ during the training of $\mathbf{w}^{(2)}$ by $\mathbf{e}_i^{(2)} = (e_1^{(2)}, \cdots, e_m^{(2)})^t$. Given an input vector $\mathbf{x}_i$, the predicted output from the NN is

$$\hat{y}_i = f^{(2)}(f_1^{(1)}(\mathbf{x}_i), \cdots, f_m^{(1)}(\mathbf{x}_i)|\mathbf{w}, \mathbf{b}) \tag{12}$$

where $\mathbf{w} = \{\mathbf{w}_1^{(1)}, \cdots, \mathbf{w}_m^{(1)}, \mathbf{w}^{(2)}\}, \mathbf{b} = \{b^{(1)}, b^{(2)}\}$. $\mathbf{w}$ and $\mathbf{b}$ are estimated by minimizing the empirical loss that measures the distance between the observed and predicted outcomes

$$l(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) = \textstyle\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n \Big(y_i - f^{(2)}(f_1^{(1)}(x_i), \cdots, f_m^{(1)}(x_i)|\mathbf{w}, \mathbf{b})\Big)^2 \tag{13}$$

NI in the input and hidden layers will lead to a change in the predicted output, which is

$$\Delta\hat{y}_i = f^{(2)}(f_1^{(1)}(\mathbf{x}_i + \mathbf{e}_i^{(1)}) + e_1^{(2)}, \cdots, f_m^{(1)}(\mathbf{x}_i + \mathbf{e}_i^{(1)}) + e_m^{(2)}) - f^{(2)}(f_1^{(1)}(\mathbf{x}_i), \cdots, f_m^{(1)}(\mathbf{x}_i)), \tag{14}$$

that is approximated, through the first-order Taylor expansion at $\mathbf{x}_i$, by

$$\Delta\hat{y}_i \approx \Psi_i \cdot \mathbf{e}_i = \left(\frac{\partial f^{(2)}}{\partial f_1^{(1)}} \frac{\partial f_1^{(1)}}{\partial \mathbf{x}_i}, \cdots, \frac{\partial f^{(2)}}{\partial f_m^{(1)}} \frac{\partial f_m^{(1)}}{\partial \mathbf{x}_i}, \frac{\partial f^{(2)}}{\partial f_1^{(1)}}, \cdots, \frac{\partial f^{(2)}}{\partial f_m^{(1)}} \right) \cdot \mathbf{e}_i, \tag{15}$$

where $\frac{\partial f_j^{(1)}}{\partial \mathbf{x}_i} = \left(\frac{\partial f_j^{(1)}}{\partial x_{i1}}, \cdots, \frac{\partial f_j^{(1)}}{\partial x_{ip}} \right)$ and $\mathbf{e}_i = (e_{ij1}^{(1)}, \cdots, e_{ijp}^{(1)}, e_1^{(2)}, \cdots, e_m^{(2)})^t$.

Definition 2. Sensitivity of a NN model $S(\mathbf{w}, \mathbf{b})$ is defined as the ratio between the variance of $\Delta\hat{y}_i$ and the variance of the total noise injected per case during the training of the NN,

$$S(\mathbf{w}, \mathbf{b}) = \frac{\sum_{i=1}^n \mathrm{Var}(\Delta\hat{y}_i)}{\mathrm{Var}(\mathbf{1}^t \mathbf{e}_i)} \approx \frac{\sum_{i=1}^n \mathrm{Var}\left(\Psi_i \mathbf{e}_i\right)}{\mathrm{Var}(\mathbf{1}^t \mathbf{e}_i)} = \frac{\sum_{i=1}^n \Psi_i^t D \Psi_i}{\mathbf{1}^t D \mathbf{1}}$$

where $D = \mathrm{diag}\left(\left|w_{11}^{(1)}\right|^{-\gamma}, \cdots, \left|w_{mp}^{(1)}\right|^{-\gamma}, \left|w_1^{(2)}\right|^{-\gamma}, \cdots, \left|w_m^{(2)}\right|^{-\gamma} \right)$, Ψ_i is given in Eqn (15) and $\mathbf{1}$ is a column vector with $m + mp$ 1's.

Remark 3. In a deep NN that contains multiple hidden layers and more than one output nodes, the only modification to Eqns (14) and (15) and Definition 2 is the inclusion of more partial derivatives terms of those layers' activation functions and expressing $\Delta\hat{y}_i$ as a vector. As such, Theorem 2 holds in the general framework of deep NNs.

Theorem 2. Minimizing the loss function in Eqn (13) in a deep NN coupled with the whiteout procedure is first-order equivalent to minimizing simultaneously the loss function with and the sensitivity of the NN model.

$$\tilde{l}(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) = l(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) + \lambda S(\mathbf{w}, \mathbf{b}) \approx \sum_{i=1}^n \left((y_i - \hat{y}_i)^2 + \lambda \frac{\Psi^t D \Psi}{\mathbf{1}^t D \mathbf{1}} \right).$$

The proof of Theorem 2 is given in Appendix B.

2.3 Consistency of the whiteout procedure

As demonstrated in Sections 2.1 and 2.2, whiteout can be regarded as a technique to mitigate the over-fitting problem and to improve the generalization of a trained NN model with an additional penalization term. In this section, We establish theoretically that the noise-perturbed empirical loss function with whiteout converges almost surely to the ideal loss function, and the estimates of NN parameters obtained from minimizing the former loss function are consistent with those obtained from minimizing the ideal loss function. We also investigate the boundary properties of the empirical loss functions with a finite n, which is important from a practical implementation perspective when whiteout is incorporated in the BP algorithm.

Denote the true and unknown underlying distribution of $(\mathbf{X}, \mathbf{Y})$ by $p(\mathbf{X}, \mathbf{Y})$ from which training data $(\mathbf{x}_i, y_i)$ are sampled. Denote the assumed relation between input $\mathbf{x}$ and output $\mathbf{y}$ by f. In the framework of NNs, $f(\mathbf{y}|\mathbf{x}, \mathbf{w}, \mathbf{b})$ is a composition of activation functions connecting the nodes between layers, with parameters $\mathbf{w}$ and $\mathbf{b}$. Per the universal

approximation theorem (Hornik, 1991), a feed-forward NN is an universal approximator for any function under some mild assumptions. We start with defining several types of loss functions, and assume the NN model f is the same across the loss functions.

Definition 3.

a). The *ideal loss function (ilf)* is $l(\mathbf{w}, \mathbf{b}) = \mathrm{E}_{\mathbf{x},\mathbf{y}}(f(\mathbf{x}) - \mathbf{y})^2$. $l(\mathbf{w}, \mathbf{b})$ is not computable with unknown $p(\mathbf{X}, \mathbf{Y})$.

b). The *empirical loss function (elf)* is the realized version of ilf in data $(\mathbf{x}, \mathbf{y})$: $l(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) = n^{-1} \sum_{i=1}^{n} (f(\mathbf{x}_i) - y_i)^2$.

c). The *perturbed empirical loss function (pelf)* in data $(\mathbf{x}, \mathbf{y})$ is $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) = (kn)^{-1} \sum_{j=1}^{k} \sum_{i=1}^{n} (f(\mathbf{x}_i, \mathbf{e}_{ij}) - y_i)^2$, where k is the number of epochs in a deep learning algorithm (e.g., BP) and $\mathbf{e}_{ij}$ represents the collective noises added to case i (in both input and hidden layers) in the j^{th} epoch during training.

d). The *noise-marginalized empirical loss function (nm-pelf)* is the expectation of pelf over the distribution of noise: $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) = \mathrm{E}_{\mathbf{e}}(l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}))$. nm-pelf can be interpreted as training a NN model by minimizing the perturbed empirical loss function with a finite n but with infinite number of epochs ($k \to \infty$).

e). The *marginalized perturb empirical loss function (m-pelf)* is the expectation of nm-pelf over the distribution of data: $l_p(\mathbf{w}, \mathbf{b}) = \mathrm{E}_{\mathbf{x},\mathbf{y}}(l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})) = \mathrm{E}_{\mathbf{x},\mathbf{y},\mathbf{e}}(l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})))$. Since $p(\mathbf{x}, \mathbf{y})$ is often unknown, either assumed parameter distribution, or nonparametric density estimation can be used to approximate $p(\mathbf{x}, \mathbf{y})$.

Minimizes nm-pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ is a regularized minimization of elf with an added penalty term to mitigate overfitting (Sections 2.1 and 2.2). In the practical implementation of the deep learning algorithm, since the number of epochs k is finite, the whiteout procedure, eventually leads to minimization of pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ rather than nm-pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ (requires $k \to \infty$). In what follows, we establish the convergence of $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ to $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ (Lemmas 2 and 3), the convergence of $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ to $l_p(\mathbf{w}, \mathbf{b})$ (Lemma Lemma 4), and eventually $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ to $l_p(\mathbf{w}, \mathbf{b})$ (5). The proof of the lemmas are provided in the Appendix C to Appendix E.

Lemma 2. Pointwise convergence of pelf to nm-pelf: There exists an upper bound dimension-free Lipschtz constant B ($B > 0$) on $\frac{\partial f}{\partial e}$, that is independent of NN model parameters $(\mathbf{w}, \mathbf{b})$, such that

$$\Pr(|l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) - l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})| > t) \leq 2 \exp\left(-\frac{knt^2}{2B}\right) \text{ for any } t > 0 \qquad (16)$$

Since B is independent of the values and dimension of $(\mathbf{w}, \mathbf{b})$, the upper bound given in Lemma 2 is uniform with regard to $(\mathbf{w}, \mathbf{b})$, and thus guarantees that as $k \to \infty$, the difference between pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ and nm-pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ approaches 0 with probability 1. Note that a dimension-free Lipschtz constant B only exists for strictly log-concave distributions such as Gaussian distribution, hence the tail bound of in the pelf is much tighter

than that of shakeout where Bernoulli noise in injected, given that the variance of the noise terms between the two are comparable.

Lemma 3. Almost sure convergence of pelf to nm-pelf: $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ converges to $l_p(\mathbf{w}, \mathbf{b}|\mathbf{e})$ almost surely as $k \to \infty$.

$$|l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) - l_p(\mathbf{x}, \mathbf{y}|\mathbf{x}, \mathbf{y})| < \delta \text{ as } k \to \infty \text{ for every } \delta > 0 \text{ and all } (\mathbf{x}, \mathbf{y}) \in R^{p+1}$$

Lemma 4. Almost sure convergence of nm-pelf to m-pelf: $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ converges to $l_p(\mathbf{w}, \mathbf{b})$ uniformly as $n \to \infty$.

$$|l_p(\mathbf{w}, \mathbf{b}) - l_p(\mathbf{w}, \mathbf{b})| < \delta \text{ as } n \to \infty \text{ for every } \delta > 0 \text{ and all } (\mathbf{w}, \mathbf{b}) \in R^K$$

where K is the dimension of the NN model (dimension of $(\mathbf{w}, \mathbf{b})$).

By the the triangle inequality, $|l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) - l_p(\mathbf{w}, \mathbf{b})| \leq |l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) - l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})| + |l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}) - l_p(\mathbf{w}, \mathbf{b})|$. With Lemma 3 and Lemma 4 established, we can easily obtain the almost sure convergence of $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ to $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$.

Lemma 5. Almost sure convergence of pelf to m-pelf: $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ converges to $l_p(\mathbf{w}, \mathbf{b})$ uniformly as $k \to \infty$ and $n \to \infty$,

$$\begin{aligned} &l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e}) - l_p(\mathbf{w}, \mathbf{b})| < \delta \text{ as } k \to \infty, n \to \infty, \\ &\text{for every } \delta > 0 \text{ and all } (\mathbf{x}, \mathbf{y}) \in R^{p+1}, (\mathbf{w}, \mathbf{b}) \in R^K. \end{aligned}$$

Estimates of $(\mathbf{w}, \mathbf{b})$ and thus predicted Y obtained from minimizing m-pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ will always be different from those obtained by minimizing elf $l(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y})$ regardless of n and k, due to the regularization effect. Given this, the variance σ^2 of the Gaussian distribution in whiteout should kept small so that the estimates of $(\mathbf{w}, \mathbf{b})$ won't deviate much from an ideal local optimum. We present below a sufficient but not necessary condition in Lemma 6, a bound on σ^2, to achieve consistency between pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ and ilf $l(\mathbf{w}, \mathbf{b})$ as $n \to \infty$ and $k \to \infty$. The proof is provided in Appendix F.

Lemma 6. Suppose that K is a class-s kernel ($s \geq 1$). Let g denote be the true density of random variable $\mathbf{x}$ (of dimension d) whose weak partial derivatives $D^\alpha g$ are integrable ($|\alpha| = s$), and $g_{n,\sigma}$ is the kernel smoothed density estimate of $\mathbf{x}$. Assume that for some $\varepsilon > 0$, $\int_{R^d}(1 + ||\mathbf{x}||^{d+\varepsilon})K(\mathbf{x})^2 < \infty$ and $\int_{R^d} ||\mathbf{x}||^{d+\varepsilon} g(\mathbf{x}) < \infty$. When s is even and

$$\sigma(n) = n^{-\frac{1}{d+2s}} \left(\frac{d\sqrt{\int_{R^d} K(\mathbf{x})^2} \int_{R^d} \sqrt{g(\mathbf{x})}}{2s \int_{R^d} \left| \sum_{|\alpha| \leq s} \frac{1}{\alpha!} D^\alpha g(\mathbf{x}) \int_{R^d} \mathbf{x}^\alpha K(\mathbf{x}) \right|} \right)^{-\frac{2}{d+2s}}, \tag{17}$$

$$\text{then} \quad E\left(\int_{R^d} \| g_{n,\sigma}(\mathbf{x}) - g(\mathbf{x}) \| \right) \leq$$

$$(1 + \delta(n))\left(\sigma^s \int_{R^d} \left| \sum_\alpha \frac{1}{\alpha!} D^\alpha g(\mathbf{x}) \int_{R^d} \mathbf{x}^\alpha + K(\mathbf{x}) \right| + (n\sigma^d)^{-\frac{1}{2}} \sqrt{\int_{R^d} K(\mathbf{x})^2} \int_{R^d} \sqrt{g(\mathbf{x})} \right),$$

where $\delta(n) \to 0$ as $n \to \infty$.

Theorem 3. If σ^2 of the whiteout Gaussian noise satisfies Eqn (17), and Lemma5 holds, then $(\mathbf{w}, \mathbf{b})$ estimated by minimizing $l_p((\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$ (pelf) are consistent with the estimates from minimizing $l(\mathbf{w}, \mathbf{b})$ (ilf) as $n \to \infty$ and $k \to \infty$.

2.4 back-propagation with whiteout

We present in this section the algorithmic realization of whiteout in the context of the BP procedure in deep NNs. Let $p^{(l)}$ and $p^{(l+1)}$ denote the number nodes in layer l and layer $l+1$, respectively, where $l = 1, \ldots, L-1$ and L is the total number of fully-connected layers. Weight $w_{ij}^{(l+1)}$ connects the i^{th} node in layer l and the j^{th} node in layer $l+1$. The *training loss* is $D = \frac{1}{2}\sum_{k=1}^{n}(\hat{y}_k - y_k)^2$, where $\hat{y}_k$ is the predicted label via the learned NN and y_k is the observed label in case k; and the activation function between two layers is denoted by f. The detailed steps in the BP algorithm with whiteout are illustrated with additive gen noise $e_{ij} \sim N(0, \frac{\sigma^2}{|w_{ij}|^\gamma} + \lambda)$ in Table 1. The steps are similar with multiplicative noise, except for the calculations of u_j^{l+1} and $\frac{\partial u_j^{(l+1)}}{\partial w_{ij}^{(l+1)}}$, which are provided in the table footnote.

input: learning step: η; tuning parameters in whiteout Gaussian noise: $(\sigma^2, \gamma, \lambda)$

1. **feed forward (FF)**:
   - sample $e_{1j}, \ldots, e_{pj}$ from $N(0,1)$:
   - calculate $u_j^{(l+1)} = b_i^{(l+1)} + \sum_{i=1}^{p^{(l)}} \left(x_i^{(l)} + e_{ij}\sqrt{\frac{\sigma^2}{|w_{ij}^{(l+1)}|^\gamma} + \lambda} \right) w_{ij}^{(l+1)}; x_j^{(l+1)} = f(u_j^{(l+1)})$
2. **back propagation**:
   - $\frac{\partial D}{\partial x_i^{(l)}} = \frac{\partial D}{\partial u_i^{(l+1)}}\frac{\partial u_i^{(l+1)}}{\partial x_i^{(l)}} = \frac{\partial D}{\partial u_i^{(l+1)}} w_{ij}^{(l+1)}$, where $\frac{\partial D}{\partial u_j^{(l+1)}} = -2(y_i - x_i^L) f'\left(u_j^{(l+1)}\right)$ for $l = L-1$, and $\frac{\partial D}{\partial u_j^{(l+1)}} = \frac{\partial D}{\partial x_i^{l+1}} f'\left(u_j^{(l+1)}\right)$ for $l < L-1$
   - weight update: $w_{ij}^{(l+1)} = w_{ij}^{(l+1)} + \eta \frac{\partial D}{\partial w_{ij}^{(l+1)}}$, where $\frac{\partial D}{\partial w_{ij}^{(l+1)}} = \frac{\partial D}{\partial u_i^{(l+1)}}\frac{\partial u_i^{(l+1)}}{\partial w_{ij}^{(l+1)}} = \frac{\partial D}{\partial u_i^{(l+1)}}\left(x_i^{(l)} + e_{ij}\frac{(2-r)\sigma^2\left|w_{ij}^{(l+1)}\right|^{1-r} + 2\lambda\left|w_{ij}^{(l+1)}\right|}{2\sqrt{\sigma^2\left|w_{ij}^{(l+1)}\right|^{2-\gamma} + \lambda\left|w_{ij}^{(l+1)}\right|^2}} \right)$
   - bias update: $b_j^{(l+1)} = b_j^{(l+1)} + \eta \frac{\partial D}{\partial b_j^{(l+1)}}$, where $\frac{\partial D}{\partial b_j^{(l+1)}} = \frac{\partial D}{\partial u_j^{(l+1)}}\frac{\partial u_j^{(l+1)}}{\partial b_j^{(l+1)}} = \frac{\partial D}{\partial u_j^{(l+1)}}$

output: final estimates of $(\mathbf{w}, \mathbf{b})$ after k epochs of FF/BP in all k cases

With multiplicative noise, all steps above are the same except for the calculations of $u_j^{(l+1)}$ and $\frac{\partial u_j^{(l+1)}}{\partial w_{ij}^{(l+1)}}$, that is, **FF:** $u_j^{(l+1)} = b_j^{(l+1)} + \sum_{i=1}^{p^{(l)}} x_i^{(l)} \left(1 + e_{ij}\sqrt{\frac{\sigma^2}{|w_{ij}^{(l+1)}|^\gamma} + \lambda}\right) w_{ij}^{(l+1)}$, and **BP:** $\frac{\partial u_j^{(l+1)}}{\partial w_{ij}^{(l+1)}} = x_j^{(l)} + \quad x_i^{(l)} e_{ij}\frac{(2-\gamma)\sigma^2\left|w_{ij}^{(l+1)}\right|^{1-\gamma} + 2\lambda|w_{ij}^{(l+1)}|}{2\sqrt{\sigma^2|w_{ij}^{(l+1)}|^{2-\gamma} + \lambda|w_{ij}^{(l+1)}|^2}}$

Table 1: Backpropagation with whiteout

As can be seen in Table 1, the BP algorithm with whiteout involves calculation of many derivatives during training, which can be time consuming and prone to large rounding errors. A computationally more efficient way, accompanied with improved accuracy, is to calculate the regular BP derivatives and the noise related partial derivatives separately,

suggested by Proposition 1 (the proof is provided in Appendix G). In other words, we can think of whiteout NI as superimposing a "noise NN" onto the "data NN". During the BP step in an epoch iteration, the two superimposed NNs receive the same values passed down from the higher layers, and calculate the partial derivatives in their respective cases, which are then combined only to lead to the same derivatives as those listed in Table 1.

Proposition 1. Let the left superscript below denote the two "superimposed" NNs: 1 is the the "data' 'NN and 2 refers to the "noise" NN, then the derivatives involved in the computation of $u_j^{(l+1)}$ and updating of weights and bias in whiteout-augmented BP algorithm can be achieved by computation by part in the two "superimposed" NNs.

$$
\begin{aligned}
u_j^{(l+1)} &= {}^1u_j^{(l+1)} + {}^2u_j^{(l+1)} \\
\frac{\partial D}{\partial w_{ij}^{(l+1)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{\partial {}^1u_j^{(l+1)}}{\partial {}^1w_{ij}^{(l+1)}} \frac{\partial {}^1w_{ij}^{(l+1)}}{\partial w_{ij}^{(l+1)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{d^2u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 w_{ij}^{(l+1)}} \frac{\partial^2 w_{ij}^{(l+1)}}{\partial w_{ij}^{(l+1)}} \\
\frac{\partial D}{\partial b_i^{(l+1)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{d^1u_j^{(l+1)}}{\partial {}^1b_i^{(l+1)}} \frac{\partial {}^1b_i^{(l+1)}}{\partial b_i^{(l+1)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial^2 u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 b_i^{(l+1)}} \frac{\partial^2 b_i^{(l+1)}}{\partial b_i^{(l+1)}} \\
\frac{\partial D}{\partial x_j^{(l)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{\partial {}^1u_j^{(l+1)}}{\partial {}^1x_i^{(l)}} \frac{\partial {}^1x_i^{(l)}}{dx_i^{(l)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{d^2u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 x_i^{(l)}} \frac{\partial^2 x_i^{(l)}}{\partial x_i^{(l)}}.
\end{aligned}
$$

3 Experiment

In this section, we compare whiteout with the regular BP, dropout and shakeout using the hand written digits data MNIST. We employed the same NN structure in Kang et al. (2016) (Table 2) to facilitate the comparisons across the different algorithms. Layer 1 and layer 2 were convolutionart layers followed by ReLU nonlinear activation and max-pooling; Layer 3 and layer 4 were fully-connected layers. Dropout, shakeout, and whiteout were applied to hidden nodes in the fully connected layers (layers 3 and 4).

Layer	1	2	3	4
Type	convolutional	convolutional	fully-conn.	fully-conn.
Channels/Nodes	20	50	500	10
Filter Size	5×5	5×5	-	-
Convolutional Stride	1	1	-	-
Pooling Size	2×2	2×2	-	-
Pooling Stride	2	2	-	-
Activation Function	ReLU	ReLU	ReLU	Softmax

Table 2: Adopted NN model in the MNIST data (reproduced from Kang et al. (2016))

In whiteout, we applied the multiplicative gen noise with three tuning parameters (σ^2, λ and γ) with a 6-fold CV. The 6-fold CV procedure was repeated 10 times with different random seeds. The classification errors on the testing data set were summarized over the 10 repetitions to obtain the mean and standard deviation of the error rate in each algorithm. The results are given in Table 3. Whiteout had the best performance (lowest misclassification rate) when the size of training data was relatively small with its efficient regularization

training size n	regular BP	dropout	shakeout	whiteout
500	6.70±0.24%	7.10±0.20%	4.84±0.27%	4.73±0.19%
1,000	4.76±0.13%	5.01±0.13%	3.33±0.14%	3.25±0.18%
3,000	2.50±0.11%	2.40±0.12%	2.00±0.16%	1.89±0.13%
8,000	1.78±0.10%	1.68±0.11%	1.47±0.10%	1.38±0.07%
20,000	1.34±0.05%	1.07±0.06%	1.25±0.08%	1.13±0.06%
50,000	0.90±0.06%	0.88±0.07%	0.97±0.13%	0.95±0.11%

Table 3: Misclassification rates (mean±SD) in regular BP, dropout, shakeout and whiteout in training data of different training size

on over-fitting. The misclassification rate decreased by almost 50% in whiteout compared to the regular dropout procedure, and also improved over the shakeout procedure though not as dramatically. The improvement continued until when training size was 8,000. When the training set was large (20,000 and 50,000), dropout procedure performed the best, followed by whiteout and then shakeout; but the differences were small. Note that a constant σ^2 was employed in all scenarios of n. The difference might disappear if σ^2 had been specified as a function n (smaller σ^2 with larger n).

The evolvement of the classification accuracy in the training data set during the iterations of the regular BP, dropout, shakeout, and whiteout algorithms are depicted in Figure 1. The error stabilized in all algorithm after a certain number of epochs, but to different error rate level. Whiteout yielded the smallest error rate among all algorithms. The training loss was bounded in all algorithms, serving as empirical evidence on the feasibility of minimizing the pelf $l_p(\mathbf{w}, \mathbf{b}|\mathbf{x}, \mathbf{y}, \mathbf{e})$. Since the magnitude of noise injected in shakeout and whiteout depended on the weights being updated in each iteration (adaptive), there was more fluctuation around error rates and training loss compared to the regular BP (no NI) and dropout (dropping nodes at a constant rate); whiteout nevertheless fluctuated much less than shakeout because of the tighter bound (Lemma 2).

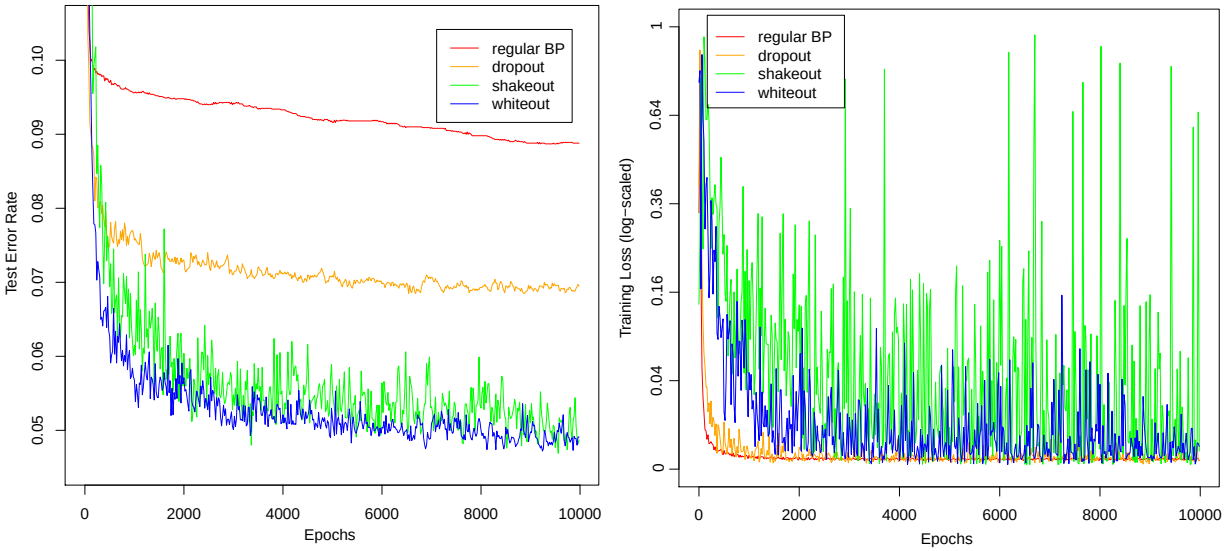

Figure 1: Misclassification error and training loss of the learned NNs in the MNIST data

4 Conclusion

Whiteout is a flexible and efficient approach to regulate overfitting and improve generalization and prediction in deep NNs. Whiteout is associated with an optimization objective function in the context of GLMs with a closed-form penalty term that includes lasso, ridge regression, adaptive lasso, and elastic net as special cases, and can also incorporate group structures among the features. Whiteout can also be viewed as robust learning of NN models in the presence of small and insignificant perturbations in the input and hidden nodes. Computationally, whiteout can be incorporated in the popular iterative BP algorithm in deep learning, where whiteout noises are sampled from Gaussian distributions with variance terms adaptive to the weights trained up to the latest epoch. Contrast to shakeout where the derivatives of sign functions are approximated, our proposed algorithm with whiteout does not involve such as approximation. In terms of performance, whiteout has better prediction performance than dropout, when training data are relatively small in size; compared to shakeout, the penalized objective function in whiteout is more stable and has better convergence behaviors during training. We established the almost sure convergence of noise-perturbed empirical loss function to the ideal loss function as the number of epochs and the size of the training data approach infinity, and the consistency of estimated parameters in a trained NN under mild assumptions. Future work include examination of convergence rates of the noise-perturbed empirical loss function to the ideal loss function, and applications of whiteout in more real-life data sets.

Appendix

A Proof of Lemma 1

$$\begin{aligned}\sum_{i=1}^{n} \mathrm{E}_{\mathbf{e}}(l_p(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i, \mathbf{e}_i)) &= \sum_{i=1}^{n} -(y_i \mathrm{E}_{\mathbf{e}}(\tilde{\mathbf{x}}_i)\mathbf{w} - \mathrm{E}_{\mathbf{e}}(A(\tilde{\mathbf{x}}_i\mathbf{w}))) = \sum_{i=1}^{n} -(y_i\mathbf{x}_i\mathbf{w} - \mathrm{E}_{\mathbf{e}}(A(\tilde{x}_i\mathbf{w})))\\ &= \sum_{i=1}^{n} l_p(\mathbf{w}|\tilde{\mathbf{x}}_i, y_i) + R(\mathbf{w}), \text{ where } R(\mathbf{w}) \triangleq \sum_{i=1}^{n} \mathrm{E}_{\mathbf{e}}(A(\tilde{\mathbf{x}}_i\mathbf{w})) - A(\mathbf{x}_i\mathbf{w}).\end{aligned}$$

A second -order Taylor expansion of $A(\tilde{\mathbf{x}}_i\mathbf{w})$ around $\mathbf{x}_i\mathbf{w}$ and taking expectation on the approximation with regardless to the distribution of noise leads to

$$\begin{aligned}\mathrm{E}_{\mathbf{e}}(A(\tilde{\mathbf{x}}_i\mathbf{w})) &\approx A(\mathbf{x}_i\mathbf{w}) + A'(\mathbf{x}_i\mathbf{w})\mathrm{E}_{\mathbf{e}}(\tilde{\mathbf{x}}_i\mathbf{w} - \mathbf{x}_i\mathbf{w}) + \frac{1}{2}A''(\mathbf{x}_i\mathbf{w})\mathrm{Var}_{\mathbf{e}}(\tilde{\mathbf{x}}_i\mathbf{w} - \mathbf{x}_i\mathbf{w})\\ &= A(\mathbf{x}_i\mathbf{w}) + \frac{1}{2}A''(\mathbf{x}_i\mathbf{w})\mathrm{Var}_{\mathbf{e}}(\tilde{\mathbf{x}}_i\mathbf{w})\\ \text{thus, } R(\mathbf{w}) &\approx \frac{1}{2}\sum_{i=1}^{n} A''(\mathbf{x}_i\mathbf{w})\mathrm{V}_{\mathbf{e}}(\tilde{\mathbf{x}}_i\mathbf{w})\end{aligned}$$

B Proof of Theorem 2

$$\tilde{l}(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y}) \approx \sum_{i=1}^{n}\left((y_i-\hat{y}_i)^2+\mathrm{Var}_{\mathbf{e}}(\Psi_i\mathbf{e}_i)\right)=\sum_{i=1}^{n}\left((y_i-\hat{y}_i)^2+\mathrm{E}_{\mathbf{e}}(\Psi_i\mathbf{e}_i)^2-(\mathrm{E}_{\mathbf{e}}(\Psi_i\mathbf{e}_i))^2\right)$$
$$=\sum_{i=1}^{n}\mathrm{E}_{\mathbf{e}}((y_i-\hat{y}_i)^2-(\Psi_i\mathbf{e})^2)=\mathrm{E}_{\mathbf{e}}\left(\sum_{i=1}^{n}((y_i-\hat{y}_i)^2-(\Psi_i\mathbf{e})^2)\right)$$

C Proof of Lemma 2

Proof of Lemma 2 employs concentration inequality of independent variables (Pollard, 1984). Let $\boldsymbol{\epsilon}=(\epsilon_1,\ldots,\epsilon_n)$ be a vector of n independent standard Gaussian variables, and $f:R^n\to R$ be L-Lipschitz continuous with respect to the Euclidean norm. Then $f(\boldsymbol{\epsilon})-\mathrm{E}(f(\boldsymbol{\epsilon}))$ is sub-Gaussian with parameter at most L, and for any $t\geq 0$

$$\Pr(|f(\boldsymbol{\epsilon})-E(f(\boldsymbol{\epsilon}))|\geq t)\leq 2\exp\left(\frac{-t^2}{2L^2}\right).$$

The Gaussian noise injected through whiteout per case is $\mathbf{e}=(\mathbf{e}_1^{(1)},\ldots,\mathbf{e}_m^{(1)},\mathbf{e}^{(2)})^t$ of dimension $m+pm$ (p is the number of nodes in layer l and m is the number of nodes in layer $l+1$) with mean $\mathbf{0}$ and covariance $\Sigma=\sigma^2\mathrm{diag}\left\{\left|w_{11}^{(1)}\right|^{-\gamma},\ldots,\left|w_{mp}^{(1)}\right|^{-\gamma},\left|w_1^{(2)}\right|^{-\gamma},\ldots,\left|w_m^{(2)}\right|^{-\gamma}\right\}$. To utilize concentration inequality, we re-write $\mathbf{e}=\sigma\Sigma^{1/2}\boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a standard Gaussian vector containing $pm+m$ independent elements. In other words, $\mathbf{e}$ is linear function of ϵ (Since $\mathbf{e}$ appears inside f in the inner product $\mathbf{e}_j^t\mathbf{w}_j$, working with ϵ won't change Lipschitz continuity of the NN model f). f considered in the context of whiteout is pelf $l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y},\mathbf{e})$, a composition of Lipschitz continuous activation functions, with the output function uniformly bounded. To determine the Lipschitz constant L, it suffices to bound weights $\mathbf{w}$ such as gradient $\frac{\partial f}{\partial e}$ is bounded. In the case of gallr noise, weights are naturally bounded due to the constraints imposed by the l_1 and l_1 regularization. We denote upper bound of $\frac{\partial f}{\partial e}$ as B, which is independent the dimension and values of $(\mathbf{w},\mathbf{b})$, and thus $\Pr(|l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y},\mathbf{e})-l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y})|>t)\leq 2\exp(-\frac{knt^2}{2B})$ per concentration inequality.

D Proof of Lemma 3

Since $\mathrm{E}_{\mathbf{e}}(\inf_{\mathbf{w},\mathbf{b}} l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y},\mathbf{e}))\leq\inf_{\mathbf{w},\mathbf{b}} l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y})$ and $\inf_{\mathbf{w},\mathbf{b}} l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y},\mathbf{e})$ is a concave function of the empirical measure defined by $\mathbf{e}$, hence it is a Backward Super-Martingale, and thus converges almost surely toward a random variable as $k\to\infty$. By the law of large numbers, $l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y},\mathbf{e})\to l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y})$.

E Proof of Lemma 4

$$|l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y}) - l_p(\mathbf{w},\mathbf{b})| = \left|\mathrm{E}_{\mathbf{e}}\left(n^{-1}(f(x_i,\mathbf{e})-y_i)^2\right) - \mathrm{E}_{\mathbf{e}}\left(\mathrm{E}_{\mathbf{x},\mathbf{y}}(f(\mathbf{x},\mathbf{e})-\mathbf{y})^2\right)\right|$$
$$\leq \mathrm{E}_{\mathbf{e}}\left|n^{-1}(f(\mathbf{x}_i,\mathbf{e}_i)-y_i)^2 - \mathrm{E}_{\mathbf{x},\mathbf{y}}(f(\mathbf{x},\mathbf{e})-\mathbf{y})^2\right| \text{ by Jensen's Inequality}$$

By the law of large numbers, $n^{-1}\sum_{i=1}^{n}(f(\mathbf{x}_i,\mathbf{e}_i)-\mathbf{y}_i)^2 \to \mathrm{E}_{\mathbf{x},\mathbf{y}}(f(\mathbf{x},\mathbf{e})-\mathbf{y})^2$ almost surely. Taken together, $\lim_{n\to\infty}|l_p(\mathbf{w},\mathbf{b}|\mathbf{x},\mathbf{y}) - l_p(\mathbf{w},\mathbf{b})|$
$\leq \mathrm{E}_{\mathbf{e}}\left(\lim_{n\to\infty}|n^{-1}\sum_{i=1}^{n}(f(\mathbf{x}_i,\mathbf{e})-\mathbf{y}_i)^2 - \mathrm{E}_{\mathbf{x},\mathbf{y}}(f(\mathbf{x},\mathbf{e})-\mathbf{y})^2|\right) \to 0$

F Proof of Lemma 6

Expectation of the loss function with regard to random variable $(\mathbf{X},Y)$ can be seen as averaging over infinitely many samples of $(\mathbf{X}_i,Y_i)$ where $i=1,\ldots,\infty$. The average becomes smoothed bootstrap with each noise injected as a kernel. In the feed forward process between layers l and $l+1$, each iteration is thus a smoothed bootstrap sampling from layer l. Holmström and Klemelä (1992) showed that in order for the difference in the predicted outcomes from minimizing two loss functions asymptotically approaching 0, we only need the difference between the true density g of $(\mathbf{X},Y)$ and its kernel smoothed density $g_{n,\sigma}$ asymptotically approaching 0. Under the mild assumption that the $g(\mathbf{X},Y)$ belongs to Sobolev Space, the weak partial derivatives $D^\alpha g, |\alpha| \leq s = 2$ of which are integrable.

$$\sigma(n) = n^{-\frac{1}{p+4}}\left(\frac{p\sqrt{\int_{R^p}K(x)^2\partial x}\int_{R^p}\sqrt{g}}{4\int_{R^p}|\sum_{|\alpha|=2}\frac{1}{\alpha!}D^\alpha g\int_{R^p}x^\alpha K(x)\partial x|}\right)^{-\frac{2}{p+4}}, \tag{F.1}$$

where p is the dimension of p. By the basic properties of Gaussian kernels ($s=2$), it is trivial to show that all integrations in Eqn (F.1) exist and with an upper bound that is a function of p.

G Proof of Proposition 1

As mentioned in Section 2.4, we can view whiteout NI as a "noise" NN superimposed over the "data" NN. Let the left superscript below denote the two NNs (1 is the the data NN and 2 refers to the noise NN), then

$$\text{input: } {}^1x_j^{(l)} = x_i^{(l)}, {}^2x_j^{(l)} = e_{ij}^{(l)}$$
$$\text{bias: } {}^1b_j^{(l+1)} = b_j^{(l+1)}, {}^2b_j^{(l+1)} = 0$$
$$\text{weight: } {}^1w_{ij}^{(l+1)} = w_{ij}^{(l+1)}, {}^2w_{ij}^{(l+1)} = w_{ij}^{(l+1)}\sqrt{\frac{\sigma^2}{|w_j|^\gamma}+\lambda}$$

The feed forward process calculates ${}^1u_j^{(l+1)}$ and ${}^2u_j^{(l+1)}$

$$
\begin{aligned}
{}^1u_j^{(l+1)} &= \sum_{i=1}^{p^{(l)}} {}^1w_{ij}^{(l+1)} {}^1x_i^{(l)} + {}^1b_i^{(l+1)};\ {}^2u_j^{(l+1)} = \sum_{j=1}^{p^{(l)}} {}^2w_{ij}^{(l+1)} {}^2x_i^{(l)} + {}^2b_i^{(l+1)} \\
u_j^{(l+1)} &= {}^1u_j^{(l+1)} + {}^2u_j^{(l+1)} = \sum_{i=1}^{p^{(l)}} w_{ij}^{(l+1)} x_i^{(l)} + b_i^{(l+1)} + \sum_{i=1}^{p^{(l)}} w_{ij}^{(l+1)} \sqrt{\frac{\sigma^2}{|w_j|^\gamma} + \lambda} \\
&= \sum_{i=1}^{p^{(l)}} \left(x_{j^{(l)}} + e_{ij} \sqrt{\frac{\sigma^2}{|w_{ij}|^\gamma} + \lambda} \right) w_{ij}^{(l+1)} + b_i^{(l+1)}
\end{aligned}
$$

After the calculation of ${}^1u_j^{(l+1)}$ and ${}^2u_j^{(l+1)}$, the BP process updates weights as in

$$
\begin{aligned}
\frac{\partial D}{\partial w_{ij}^{(l+1)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{\partial {}^1u_j^{(l+1)}}{\partial {}^1w_{ij}^{(l+1)}} \frac{\partial {}^1w_{ij}^{(l+1)}}{\partial w_{ij}^{(l+1)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{d^2u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 w_{ij}^{(l+1)}} \frac{\partial^2 w_{ij}^{(l+1)}}{\partial w_{ij}^{(l+1)}} \\
&= \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot x_i^{(l)} \cdot 1 + \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot e_{ij} \frac{(2-r)\sigma^2 |w_{ij}^{(l+1)}|^{1-r} + 2\lambda |w_{ij}^{(l+1)}|}{2\sqrt{\sigma^2 |w_{ij}^{(l+1)}|^{2-r} + \lambda |w_{ij}^{(l+1)}|^2}} \\
&= \frac{\partial D}{\partial u_j^{(l+1)}} (x_i^{(l)} + e_{ij} \frac{(2-r)\sigma^2 |w_{ij}^{(l+1)}|^{1-r} + 2\lambda |w_{ij}^{(l+1)}|}{2\sqrt{\sigma^2 |w_{ij}^{(l+1)}|^{2-r} + \lambda |w_{ij}^{(l+1)}|^2}})
\end{aligned}
$$

bias as in

$$
\begin{aligned}
\frac{\partial D}{\partial b_i^{(l+1)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{d^1 u_j^{(l+1)}}{\partial {}^1b_i^{(l+1)}} \frac{\partial {}^1b_i^{(l+1)}}{\partial b_i^{(l+1)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial^2 u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 b_i^{(l+1)}} \frac{\partial^2 b_i^{(l+1)}}{\partial b_i^{(l+1)}} \\
&= \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot 1 \cdot 1 + \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot 1 \cdot 0 = \frac{\partial D}{\partial u_j^{(l+1)}}
\end{aligned}
$$

and continues to the lower layer as in

$$
\begin{aligned}
\frac{\partial D}{\partial x_i^{(l)}} &= \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial {}^1u_j^{(l+1)}} \frac{\partial {}^1u_j^{(l+1)}}{\partial {}^1x_i^{(l)}} \frac{d^1 x_i^{(l)}}{dx_i^{(l)}} + \frac{\partial D}{\partial u_j^{(l+1)}} \frac{\partial u_j^{(l+1)}}{\partial^2 u_j^{(l+1)}} \frac{\partial^2 u_j^{(l+1)}}{\partial^2 x_i^{(l)}} \frac{d^2 x_i^{(l)}}{dx_i^{(l)}} \\
&= \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot {}^1w_{ij}^{(l+1)} \cdot 1 + \frac{\partial D}{\partial u_j^{(l+1)}} \cdot 1 \cdot {}^2w_{ij}^{(l+1)} \cdot 0 \\
&= \frac{\partial D}{\partial u_j^{(l+1)}} \cdot w_{ij}^{(l+1)}
\end{aligned}
$$

References

An, G. (1996). The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, 8:643–674.

Ba, J. and Frey, B. (2013). Adaptive dropout for training deep neural networks. *Advances in Neural Information Processing Systems*, pages 1–9.

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout Networks. *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 28:1319–1327.

Grandvalet, Y. (2000). Anisotropic noise injection for input variables relevance determination. *IEEE Transactions on Neural Networks*, 11(6):1201–1212.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313:504–507.

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

Holmström, L. and Klemelä, J. (1992). Asymptotic bounds for the expected L1 error of a multivariate kernel density estimator. *Journal of Multivariate Analysis*, 42(2):245–266.

Holmstrom, L. and Koistinen, P. (1992). Using additive noise in back-propagation training. *IEEE transactions on neural networks*, 3(1):24–38.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Ivakhnenko, A. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, 4:364–378.

Kang, G., Li, J., and Tao, D. (2016). Shakeout: A New Regularized Deep Neural Network Training Scheme. *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 1751–1757.

Matsuoka, K. (1992). Noise injection into inputs in back-propagation learning. *IEEE Tranactions on Systems, Man, and Cybernetics*, 22(3):436–440.

Mcculloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115.

Pollard, D. (1984). *Convergence of stochastic processes*. Springer-Verlag.

Seghouane, A. K., Moudden, Y., and Fleury, G. (2004). Regularizing the effect of input noise injection in feedforward neural networks training. *Neural Computing and Applications*, 13(3):248–254.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929−1958.

Wager, S., Wang, S., and Liang, P. (2013). Dropout training as adaptive regularization. *Advances in Neural Information Processing Systems (NIPS)*, 26:351–359.

Wainwright, M. J. and Jordan, M. I. (2008). *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. *Icml*, (1):109–111.

Wang, S. and Manning, C. D. (2013a). Fast dropout training. *Proceedings of the 30th International Conference on Machine Learning*, pages 118–126.

Wang, S. I. and Manning, C. D. (2013b). Fast dropout training. *Proceedings of the 30th International Conference on Machine Learning*, 28:118–126.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B*, 68(1):49–67.

Zhuo, J., Zhu, J., and Zhang, B. (2015). Adaptive dropout rates for learning with corrupted features. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua(Ijcai):4126–4133.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association: Theory and Methods*, 101(476):1418–1429.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2):301–320.