Robust Neural Networks Learning via a Minimization of Stochastic Output Sensitivity

JINCHENG LI\(^1\) AND WING W. Y. NG\(^2\), (Senior Member, IEEE)

\(^1\)College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China
\(^2\)School of Computing Science and Engineering, South China University of Technology, Guangzhou 510006, China

Corresponding author: Wing W. Y. Ng (wingng@ieee.org)

This work was supported in part by the Natural Science Foundation of Guangdong Province Project under Grant 2015A030310483 and in part by the National Natural Science Foundation of China under Grant 61876066.

ABSTRACT In this article, we propose Sensitivity Minimization Learning (SML) to overcome the performance degradation problem caused by features corruption at the testing phase by using the stochastic sensitivity measure (STSM) as a regularizer. The STSM measures output deviations between each training sample and its noisy versions generated by a feature perturbation strategy. The feature perturbation strategy is user defined to simulate noises that our model target to defense at the testing phase. The SML is computational efficiency in training and testing phase and minimizes the generalization error on both training samples and its noise version set with feature perturbations. Models regularized by the STSM can be trained by the stochastic gradient descent algorithm efficiently and applied to very large scale applications. Experiment results on eight grayscale image, one color image, and two face databases show that the SML significantly outperforms several regularization techniques and yields much lower classification error when testing sets are contaminated with noises.

INDEX TERMS Feature perturbation, neural networks, stochastic output sensitivity.

I. INTRODUCTION

The goal of neural network training is to generalize future testing samples well. However, due to the limited amount of training samples or heavily noise-contaminated training samples, the learned model over-fit to training data easily. Regularization is a well-studied method in neural network training to improve the generalization performance. Commonly used regularization technique including weight decay (WD) [1], [2], output smoothing [3]–[5], and noise injection to the input during training [6], [7]. These regularization techniques either assume the distribution of model parameter is Gaussian (WD) or the data noise is small (output smoothing and noise injection). However, due to the noisy and uncertain conditions in the real world, new samples may have a large deviation from the limited training samples due to feature perturbation. Feature perturbation due to noises (such as white noise, illumination, rotation or occlusion of part of the object) may degrade the performance of classifiers significantly in applications like face recognition [8]–[12], Pedestrian Detection [13]–[15], object tracking [16], and hand written recognition [17], [18]. Particular, in image recognition applications based on bag of visual words features (BoW) [19]–[21], the same object may have very different BoW features due to large varieties of background and other factors. For examples, BoW feature vectors may be different for the same object due to changes of angles, illumination, or occlusion of parts of the object. In these cases, the commonly used regularization methods may fail.

To overcome this problem, we propose a general robust model learning method based on the minimization of the output stochastic sensitivity measure (STSM) in this article. The STSM measures the output deviations between each training sample and its noise versions. Noise versions of training samples are generated by a user defined rule to simulate the noise that our trained model targets to defense. The objective function of our method consists of a training error term and a STSM term. The training error term measures how well the model fit to the training data while the STSM term measures the robustness of the trained model with respect to noise. The
learned model not only learns the training data well but also is robust to feature perturbations (i.e. noise). These two terms are traded-off by a regularization parameter. We name the proposed method as the Sensitivity Minimization Learning (SML). Advantages of SML are listed as follows:

- applicability to any type of noise that the model targets to defense
- applicability to any parametric models for which is differentiable with respect to model parameters
- only two hyper-parameters
- computational efficient

We should emphasize that different from current methods that assuming small perturbation of inputs, our method trains classifier to tolerant for feature perturbations in real applications, e.g. random feature corruption (missing), random gaussian noise. The SML minimizes training errors of clean training samples and STSMs of clean training samples and their noise versions. The change of perturbation type does not need to modify the training algorithm. This is very useful in situations that the user has prior knowledge of the noise type that the model needs to defense to avoid performance degradation.

The second advantage is that the proposed learning framework yields specific training algorithms for many kinds of models (e.g. multilayer perceptron neural networks, convolution neural network) with bounded output activation functions (e.g. sigmoid and softmax). For models being differentiable with respect to model parameters, the training error term, the STSM term, and their derivatives can be computed efficiently. The STSM can be used as a general regularization term for various settings and model architectures. In this work, we explore a special case of a multilayer perceptron neural networks.

Finally, The STSM can be approximated by a set of finite noise versions of training samples and can be optimized by the stochastic gradient descend algorithm efficiently. This makes the SML readily applicable for very large models and datasets. This is different from the L-GEM [22] for model selection and [23] because it suffers from a high computation complexity.

The rest of the paper is organized as follows. In Section II, we briefly review some works that closely related to this article. We propose our method and its optimization algorithm in Section III and Section IV, respectively. Experiment results on eight grayscale datasets with different difficulties (including six handwritten digit datasets), one natural image dataset (CIFAR-10), and two face image datasets are presented in Section V. We conclude this work in Section VI.

II. RELATED WORKS

In this section, we review commonly used and newly proposed methods to improve generalization capability of machine learning. The weight decay (WD) may be the most commonly used method [1] which is based on the Bayesian theory with the prior that network parameters follow Gaussian distributions. Other regularization technique can be derived with different priors base on the Bayesian theory, e.g. $L_1$ norm ($|W|$) of network parameters with Laplace distributions of network parameters. The $L_1$ norm of network parameters prefers most of network weight values being zero. More regularizers and their corresponding prior distribution functions can be found in [24], [25]. Network parameters regularized by WD or $L_1$ norm easily have large values in some weights but near zero for most of network parameters. Such networks easily fail to recognize testing samples correctly and may not be robust to noise.

In contrast, the noise injection [6], [7], [26] is proposed based on the principle that output of the network after learning should be insensitive to small perturbations of inputs. The output smoothing [3], [5], [7] formulates this principle mathematically as a minimization of the Frobenius norm of the Jacobian matrix (partial derivatives of outputs with respect to inputs). These mathematical formulations assume that amplitudes of noise are small. Moreover, adding noises to inputs introduces the trace of the Hessian matrix (second order partial derivatives of outputs with respect to inputs) to the objective function which shows down the optimization processes [3]. Another output smoothing method is convolution target smoothing (CTS) [4]. The application of CTS is limited by its rigid assumptions of both training data is continuously and uniformly distributed over the entire input space and the magnitude of noise is small. Under the assumption that input noise is small and training data is uniformly distributed, authors in [4] show that the CTS is approximately equal to noise injection. Authors in [26] use multiple noise samples per training example in a stochastic gradient descent iteration to achieve better performance than the simple noise injection. Instead of injecting noise to inputs, authors in [27] propose to add gaussian noise on hidden units of autoencoders to generate augmented samples for learning which achieves a significant performance improvement. On the other hand, authors in [28] propose noisy activation functions for injecting noises to nonlinear functions to allow the stochastic gradient descent to perform more diversified explorations in the search space. Noise injected to activations are easier to optimize and regularize the model to achieve a better generalization capability. Furthermore, authors in [29] propose to normalize layer inputs to have zero mean and unit variance (i.e. batch normalization) to address the internal covariate shift problem of neural networks. Experiment results show that the batch normalization prevents over-fitting and improves generalization capability.

In [30], [31], authors propose the random dropout to prevent neural networks over-fitting and achieve promising results on several image datasets. The idea of random dropout is to randomly omit a hidden unit (set output value to zero) with a probability, e.g. 50%, during the neural network training. Neural networks trained with random dropout can prevent complex co-adaptations on the training data. For a neural network with $M$ hidden neurons, random dropout can be seen as averaging over $2^M$ of neural networks with
shared weights. The major limitation of random dropout is its slow convergence which needs much larger number of weight updates to reach a local minima in comparison to other regularization techniques. To overcome the slow convergence problem of Random dropout, authors in [32] propose a fast dropout training algorithm by using an approximation of the geometric mean of outputs of 2 neural networks with shared weights. The error of this approximation be very large because its geometric mean is unbounded.

Initializing connection weights of Restricted Boltzmann Machines (RBM) [33], [34], autoencoders [35], and variants of autoencoders [36]–[39] using unsupervised pre-training improves their generalization capabilities. Unsupervised pre-training helps to prevent over-fitting in supervised fine tuning by learning the distribution information of data. Take the RBM as an example, a two-layer RBM is trained to maximize the log likelihood of raw inputs. Connection weights of the learned RBM is used to initialize the first layer of the neural network while outputs of the RBM are used as inputs to train the next two-layer RBM to initialize the second layer of the neural network. These procedures are repeated to initialize a neural network with multiple hidden layers. After pre-training, all connection weights are fine-tuned by a standard back-propagation (BP) algorithm. Unsupervised pre-training algorithms have achieved promising results on variants of applications, e.g. image classification [40]–[42], speech recognition [43], [44], and natural language processing [25], [45]. However, they usually need to train multiple shallow networks and perform hyper-parameter selection, which are time consuming.

The Localized Generalization Error Model (L-GEM) [22] was proposed to measure classifiers’ generalization capabilities for unseen samples near the training samples. The L-GEM is applied to architecture selection of Radial Basis Function Neural Network (RBFNN) [22] and feature selection [46] successfully. In [23], authors found that the sensitivity term is usually small and dominated by the training error in Multi-Layer Perceptron Neural Networks (MLPNN) training. So, the Pareto-based multi-objective training algorithm (MO-RSM) combining with a single objective local search (SO-RSM) is proposed to select a set of models that minimizing both the training error and the output sensitivity. The SO-RSM and the MO-RSM cannot be applied to large datasets due to the high computational complexity of genetic algorithm and may fail to recognize testing samples that are very different from training samples due to the assumption that input perturbation are small. The LiSSA minimizes the output sensitivity and the training error to train robust autoencoders [39].

In this article, we generalize the L-GEM to a more general case. We adopt output stochastic sensitivity measure (STSM) as a regularization term which does not limit types and magnitudes of feature perturbations. User can define task specific feature perturbation type to simulate the noise that the learned model target to defense. The learned model that training regularized by the STSM can generalize to training samples and their noisy neighborhoods.

### III. SENSITIVITY MINIMIZING LEARNING (SML)

Our proposed method is closely related to the L-GEM proposed by Yang et al. [22], [23]. We therefore formulate L-GEM (section III-A) before introducing our method (Section III-B). Feature perturbation types that used in the training in this article are introduced in Section III-C. The relationship between the SML and other regularization methods are discussed in Section III-D.

#### A. L-GEM

For a pattern classification problem, a training data set \( D \) consisting of \( N \) input-output pairs \( \{x_b, g(x_b)\}, b = 1, \ldots, N, x_b \in \mathbb{R}^n \), is drawn from the problem domain. \( g(\cdot) \) denotes the true input-output mapping that we target to learn for a given pattern classification problem. The training mean square error (MSE) of an trained model is defined as follows:

\[
R_{\text{emp}} = \frac{1}{N} \sum_{b=1}^{N} (g(x_b) - g_\theta(x_b))^2
\]

where \( g_\theta(x_b) \) denotes the output of the learned classifier with model parameter set \( \theta \) for input \( x_b \). The localized generalization error bound \( R_{\text{SM}}(Q) \) is an upper bound of the MSE of those unseen samples (noisy version samples of training samples) that have features similar to the training samples, i.e. having a distance smaller than a constant \( Q \) in the input space [22]. The \( Q \)-neighborhood of a training sample \( x_b \) is defined as

\[
S_Q(x_b) = \{ \tilde{x}_b | x_b + \Delta x; |\Delta x_i| \leq Q, \forall i = 1, \ldots, n \}
\]

where, \( Q \) is a given tiny real number larger than zero and \( \Delta x_i \) can be considered as an input perturbation (noise) which is a random variable following a zero mean uniform distribution. So, unseen (noisy version) samples in the \( Q \)-neighborhood appear uniformly with the same probability. The \( Q \)-union of the whole training dataset \( S_Q \) is defined to be the union of all \( S_Q(x_b) \). The generalization error for unseen samples located within the \( Q \)-union is defined as \( R_{\text{SM}} \). With probability \( 1 - \eta \), we have [22]:

\[
R_{\text{SM}}(Q) \leq (\sqrt{R_{\text{emp}}} + \sqrt{E_{S_Q}((\Delta y)^2)} + A)^2 + \epsilon = R_{\text{SM}}^*(Q)
\]

where \( \epsilon = B \ln(1/\eta)/(2N)^{1/2}, E_{S_Q}((\Delta y)^2) \), \( 1 - \eta, A \) and \( B \) denote the stochastic sensitivity measure (STSM) of output difference \( (\Delta y) \), the confidence of the upper bound, the difference between the maximum and minimum values of the target output, and the maximum possible value of the MSE, respectively. \( A \) and \( B \) can be fixed when the training dataset is given. The \( R_{\text{SM}}^* \) consists of three major components: the training error \( (R_{\text{emp}}) \), the STSM \( (E_{S_Q}((\Delta y)^2)) \), and constants defined by the training data set. The generalization error upper bound of L-GEM shows that minimizing the training error and the model sensitivity yield a classifier with small
generalization error on the training samples and their noisy (perturbed) neighborhoods.

**B. PROPOSED SML**

The L-GEM is a successful method for RBFNN architecture and feature selection. However, the computation of STSM for other neural networks, e.g. MLPNN, is time consuming and the input perturbation of feature is restricted to a small value and follow a zero mean uniform distribution. In this section, we generalize the L-GEM to a more general learning model. From Equation 3, a classifier with small generalization error on the training samples and its noisy training set can be obtained if both the training error and the sensitivity terms are minimized. So, the proposed objective function of SML is given as follows:

$$\arg \min_{\theta} L(\theta; D) = \arg \min_{\theta} J(\theta, D) + \lambda R_T(\theta, D)$$

where $J(\theta, D)$ is the training error (e.g. cross-entropy loss or square error) for the labeled training dataset and $R_T(\theta, D)$ is the STSM of the learned model, respectively. These two terms are traded-off by a regularization parameter $\lambda$ which is a larger than zero real-value. The STSM is defined as the average of the expectation of differenciers between outputs of each training sample and its noisy version generated following a rule $T$. The $D(g, g')$ is a non-negative function that measures the difference between outputs of two model, e.g. cross entropy. The $S_T(x_b)$ is the set of noisy training data points of the training sample $x_b$ generated by the rule $T$. The $S_T(x_b)$ is defined as follows:

$$S_T(x_b) = \{\tilde{x}_b | x_b + \Delta x_i, i = 1, \ldots, n\}$$

where $\Delta x_i$ is generated by the feature perturbation rule $T$ that our model targets to defense. For instance, $T$ is defined as a zero mean uniform distribution in the L-GEM.

The training error term $J(\theta, D)$ can be computed efficiently using training samples. However, the computation of the STSM term $R_T(\theta, D)$ is difficult, and $T$ is task specific. We propose an approximation method and adopt stochastic gradient descent (SGD) to minimize Equation 4 in Section IV. The application of SGD is independent to the feature perturbation type. In Section III-C, two types of random feature perturbation rules that improve generalization capability and model robustness significantly are defined.

**C. FEATURE PERTURBATION TYPES ADOPTED IN THIS PAPER**

In this section, we define two types of feature perturbations that used in this article: the random feature corruption perturbation (RFC) and the random normal distribution noise perturbation (RND). These two types of feature perturbation yield significant improvement on both generalization capability and robustness in our experiments. Testing samples suffering from these two types of feature corruptions are commonly happened in testing phase in pattern recognition tasks. These two feature perturbation types can be used alone or together in the training phase. We should note that the feature perturbation type is not limited to the two being introduced in this section. Zero mean uniform distribution noise corruption can also improve generalization capability and robustness but the improvement is not as significant as RFC’s and RND’s.

1) **RANDOM FEATURE CORRUPTION FEATURE PERTURBATION (RFC)**

We define a random feature corruption feature perturbation rule to train our model to defense input feature missing in testing samples. Due to the unknown real world corruption distribution of features, without loss of generality, each feature has an equal and independent probability $p$ to be deleted randomly for each sample $x_b \in \mathbb{R}^n$ to simulate the feature missing situation during the testing phase. Following the common pre-processing strategy for missing values, missing values are set to zero. For each training sample $x_b$ with $n$ features, there are totally $2^n$ possible corrupted versions.

Then, we define $\Delta x = x_b \odot e$ and $e \in \{-1, 0\}^n$ as the collection of all possible perturbed versions of $x_b$ with each feature being corrupted with a probability $p$, i.e. we have probability $P(e_i = -1) = p$ for feature $i$. “$\odot$” denotes the element-wise inner product of two vectors. The feature corruption probability $p$ is a hyper-parameter selected by cross validation in our experiments.

2) **RANDOM NORMAL DISTRIBUTION NOISE PERTURBATION (RND)**

The random normal noise perturbation (RND) rule is defined to train our model to defense Gaussian noises in testing samples. The RND perturbation is defined as $\Delta x = N(0, \sigma^2)$, i.e. normal distribution with standard deviation of $\sigma$ and zero mean. The perturbation magnitude $\sigma$ is a hyper-parameter selected by cross validation in our experiments.

**D. RELATIONSHIP TO OTHER REGULARIZATION METHODS**

Similar to regularization techniques, the SML consists of a training error term and a STSM penalty term. The SML needs to determine both the feature perturbation parameter and the $\lambda$. The WD assumes network parameters following a Gaussian distribution, which make important features to have large weights and most of less important features to have very small weight values to satisfy the distribution. Output smoothing assumes the input noise is small. In contrast, the STSM term does not make assumption to network parameters and noise magnitudes.

The noise injection (NI) method adds noises to training samples and then minimizes the training error on the noisy data. The NI assumes noise magnitude is small. Moreover, NI
adds unexpected factors to the objective function, which may slow down the optimization procedure [3]. Different from the NI, the SML minimizes output sensitivities between training samples and their noisy versions and minimizes the training error on the clean training data. Again, the STSM term does not make assumption on the magnitude of noise level.

For each training sample, the random dropout method randomly and independently omits the hidden activation. At the testing phase, the random dropout needs half the weights of hidden layers to approximate the geometric mean of exponential number of neural networks that shared weights. Random dropout is an approximation to the geometric mean of outputs when the number of hidden layer is larger than one. This approximation did not have upper and lower bound. Different from random dropout, SML did not need to half the weights of hidden layer at testing time. The generalization error of SML on both the training samples and its noise versions that with feature perturbation are bounded.

**IV. SML TRAINING AND ITS TIME COMPLEXITY**

The optimization of the SML using the stochastic gradient descent (SGD) [47] is introduced in Section IV-A. Section IV-B discusses the time complexity of the SML.

**A. SML OPTIMIZATION USING SGD**

The objective function \( L(\theta; T, D) \) is unconstrained and smooth, which can be optimized by gradient based optimization methods. However, the exact gradient of \( L(\theta; T, D) \) is difficult to compute because of the computation of \( R_T(\theta, D) \) usually involves infinite number of samples. To address this problem, \( L(\theta; T, D) \) is optimized using the SGD. In each weight update of the SGD, the exact gradient of \( L(\theta; T, D) \) is approximated by one training sample. The SGD can almost surely converge to a local minimum [47] when the learning rate of SGD is decreasing during the training process and satisfy a wild assumption.

Algorithm 1 shows the procedure of the optimization of \( L(\theta; T, D) \) using the SGD. In each weight update, the SGD randomly selects a training sample \( x_b \) from the training set and obtains a corrupted version of \( x_b \) (\( \tilde{x}_b \)) according to the selected feature perturbation type \( T \) (Step 4 in Algorithm 1). Then, we compute the gradient of \( L(\theta; T, D) \) using \( x_b \) and \( \tilde{x}_b \) to approximate the exact gradient (Step 5 in Algorithm 1). These two steps are repeated until the algorithm converges to a local minimum. In our experiments, we generalize Algorithm 1 to mini-batch SGD, i.e. we sample 32 training samples in each weight update to approximate the exact gradient. The parameter of \( T \), e.g. probability \( p \) (\( 0 \leq p \leq 1 \)) when use RFC, needs to be optimized by a validation set.

**B. TIME COMPLEXITY OF SML TRAINING**

Each weight update of the SGD involves both forward and backward stages computations. In the forward stage, the computation complexity is double compared to methods that only compute the output of \( x_b \) (e.g. the WD, the NI, and the random dropout) because the SML needs to compute outputs of both \( x_b \) and a noise version \( \tilde{x}_b \). In the backward stage, the SML needs to compute the gradient information of training error term as well as the STSM term with respect to network parameters \( \theta \). So, the time complexity is also double compared to methods that compute the training error term only (e.g. the NI and the random dropout). In summary, the time complexity of each weight update in the SML is double compared to methods minimizing the training error only.

**V. EXPERIMENTAL RESULTS**

Experiment results of the SML with random feature corruption feature perturbation (SML\(_{RFC}\)) and random normal distribution noise feature perturbation(SML\(_{RND}\)) are compared with the WD, the NI (with Gaussian noise and zeromask noise), and the random dropout in three classification tasks: 1) gray image classification (8 grayscale image datasets); 2) natural image classification based on Bag of Visual Word (BoW) features, and 3) face recognition on 2 face databases. Experimental setup and results of these three tasks are discussed in Section V-A, V-B and V-C, respectively.

**WD:** The WD minimizes the training error (cross entropy) and the \( L_2 \) norm of network parameters. These two terms are traded-off by a regularization parameter \( \lambda_w \). In experiments, \( \lambda_w \) is optimized by a cross validation. The range of \( \lambda_w \) is listed in Table 2.

**NI Gaussian (NI\(_G\))**: The NI Gaussian adds Gaussian noises to training samples and then minimizes the training error of the noisy training set. The noise magnitude of the Gaussian noise is listed in Table 2.

**NI Zeromask (NI\(_Z\))**: In NI Zeromask, each feature of training samples has a probability \( p \) to be deleted randomly and independently in training phase. Then, the NI Zeromask minimizes the training error on the noisy training set. The range of \( p \) is listed in Table 2.

**Random Dropout (Dropout)**: In random dropout, each hidden neuron has a probability \( p_r \) to be omitted. The range of \( p_r \) is listed in Table 2.
TABLE 1. Description of the 8 Grayscale Image Datasets.

| Datasets       | Description                                      | Features | #Classes | #Train/#Valid/#Test |
|---------------|--------------------------------------------------|----------|----------|---------------------|
| MNIST         | The standard MNIST' digit classification dataset | 784      | 10       | 50,000/10,000/10,000 |
| mnist-basic   | A selected smaller subset of the MNIST           | 784      | 10       | 10,000/20,000/50,000 |
| mnist-rot     | mnist-basic with random rotation                 | 784      | 10       | 10,000/20,000/50,000 |
| mnist-bg-img  | MNIST digits with random image background.       | 784      | 10       | 10,000/20,000/50,000 |
| mnist-bg-rand | MNIST digits with random white noise background. | 784      | 10       | 10,000/20,000/50,000 |
| mnist-rot-bg-img | MNIST digits with added random rotation | 784  | 10       | 10,000/20,000/50,000 |
| rect          | Each image contain a rectangle but with different width and height | 784 | 2        | 1,000/200/50,000   |
| rect-img      | Rect image with added random image background     | 784      | 2        | 10,000/200/50,000   |

TABLE 2. Hyper-Parameters and Their Corresponding Search Range of Different Training Methods That Need to Be Optimized by Cross-Validation.

| Hyper-parameter | Search Range               |
|-----------------|----------------------------|
| Number of hidden layers | (1, 2, 3, 4)             |
| Number of hidden neurons per layer | (100, 500, 1000, 1500) |
| Learning rate for supervised learning | (0.001, 0.005, 0.01, 0.05, 0.1, 0.2) |
| Regularization parameter $\alpha$ of WD | $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ |
| Noise level $\sigma$ of NI and $\sigma_{\text{LGB}}$ | (0.01, 0.05, 0.1, 0.15, 0.2) |
| Noise level $p$ of NI and $\text{LGB}$ | (0.1, 0.3, 0.5, 0.7, 0.9) |
| Dropout rate $p_r$ of Random Dropout | (0.1, 0.3, 0.5, 0.7, 0.9) |
| Regularization parameter $\lambda$ of SML | (10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}) |
| $D_{\text{SML}}(\epsilon, \delta)$ | $\{\text{cross entropy, square error}\}$ |

Connection weights of the SML, the WD, the NI Gaussian ($NI_G$), the NI Zeromask ($NI_z$), and the random dropout are initialized randomly and independently from a uniform distribution. Connection weights of all bias are initialized to be zero. In grayscale image classification tasks (Section V-A), we adopt the Gaussian kernel SVM ($\text{SVM}_{rbf}$) as the baseline method. The kernel parameter and the regularization parameter of the $\text{SVM}_{rbf}$ are optimized by cross validations [48]. Furthermore, to show the superiority of the SML, we also compare it with deep neural networks that pre-trained by the Restricted Boltzmann Machine (RBM) [37].

In all three classification tasks, to validate the robustness of classifiers in the situation that features of testing samples are contaminated with noise in the testing phase. We generate noisy testing sets with random feature corruption noise and random normal distribution noise respectively to validate the robustness of different methods.

**Generating noising testing sets with RFC:**

The RFC is used to simulate Gaussian white noise in testing samples. A noise level $p$ denotes each feature has a probability $p$ to be deleted randomly and independently. For each noise level $p$, we randomly generate 10 noisy testing images for each testing image in the testing set to form a noisy testing set $\tilde{D}_p$. The noise level $p$ ranges from 0.1 to 0.9 with step 0.1, then we obtain 9 noisy test sets $\{\tilde{D}_{0.1}, \tilde{D}_{0.2}, \ldots, \tilde{D}_{0.9}\}$ with different noise levels.

**Generating noising testing sets with RND:**

The RND is used to simulate Gaussian white noise in images. This noise level is controlled by the standard deviation $\sigma$. For each noise level, we randomly generate 10 noisy testing images for each original testing image in the testing set to form a noisy testing set $\tilde{D}_\sigma$. The noise level $\sigma$ ranges from 0.1 to 0.9 with step 0.1. Then, we obtain 9 noisy testing sets $\{\tilde{D}_{0.1}, \tilde{D}_{0.2}, \ldots, \tilde{D}_{0.9}\}$ with different noise levels.

**A. GRAYSCALE IMAGE CLASSIFICATION**

We validate the classification performance of SML on 8 grayscale image databases in this section. These databases include MNIST [49] and 5 variants of MNIST database with smaller training set or by different types of noise, i.e., random rotation, random background, random image background, and both random rotation and image background [48]. In addition, there are two grayscale image databases of rectangles with different widths and heights [48]. All images in 8 databases have the same size of 28 x 28 pixels and have been split into standard and fixed training/validation/testing sets. Descriptions of these 8 databases are listed in Table 1 in details. Hyper-parameters of different training methods that need to be optimized by cross-validations are listed in Table 2.

1) **CLASSIFICATION PERFORMANCE ON 8 GRAYSCALE IMAGE DATABASES**

Table 3 shows the 95% classification error confident interval of 8 grayscale image databases in $\mu \pm \delta$ form [37], [48], where $\delta = Z_{1-\alpha/2} \sqrt{\mu(1-\mu)/N_{\text{test}}}$ and $\alpha = 0.05$. In addition, $\mu$, $N_{\text{test}}$, and $Z_{1-\alpha/2}$ denote the mean testing error, the number of testing images, and the value of inverse cumulative normal distribution function at $1-\alpha/2$. In this experiment, we adopt the SVM with Gaussian kernel as the baseline method (the Gaussian kernel parameter and the regularization parameter $C$ are selected by cross-validations) [48]. Moreover, to show the superiority of the SML, experiment results of SML are also compared to deep neural networks with 3 hidden layers that pre-trained by Restricted Boltzmann Machine (RBM-3 [37], [48]). The best result of each dataset among all methods is bolded as well as the results with confident interval overlap with the best result.

As shown in Table 3, comparing with training methods using random initialization, i.e., the WD, the NI, the Random Dropout, and the SML, the $\text{SML}_{\text{RFC}}$ (SML with RFC) yields the smallest testing classification errors in 7 out of 8 databases (except bg-rand dataset). Furthermore, the classification error confident interval of the $\text{SML}_{\text{RFC}}$ on bg-rand dataset is overlapped with the best method: NI Gaussian. We should note that, without time consuming unsupervised pre-training, the $\text{SML}_{\text{RFC}}$ yields better performances than deep networks pre-trained by the RBM. The $\text{SML}_{\text{RFC}}$ yields smaller testing errors.
classification errors in 6 out of 8 databases (except bg-rand and bg-img) and significantly outperforms the RBM-3 in 5 out of 8 databases. The performance of the SMLRF is better than the SMLRND’s in this set of experiments. The SMLRND still yields smaller testing classification error in 4 out of 8 databases compared to other methods.

In summary, the NI and the random dropout yield better results than WD’s. However, the improvement of SML is more significant compared with both the NI and the random dropout. Usually, NI methods obtain the best result when the noise level is small and the performance is degrading when noise level gets larger. In contrast, the STSM term of the SML usually needs a large noise level when databases are contaminated by larger noises, e.g. in the two most difficult datasets bg-img and rot-bg-img. The best model selected by the SMLRFC with feature deletion probability $p$ is 90%. This phenomenon also shows that the SML is very different with NI methods because the SML does not limited to small noise levels.

2) CLASSIFICATION PERFORMANCE ON MNIST DATABASE WITH NOISY TESTING SAMPLES
We validate the performance of SML on the MNIST dataset when testing set is contaminated by random noises. As Section V-A1, we train model on training set, but evaluate its performance by noisy testing sets with different noise levels. We validate the robustness of SML on testing sets with random feature corruption noise (each feature has a probability $p$ to be deleted) and random normal distribution noise, respectively. For each noise level $p$ ($p = [0.1 : 0.1 : 0.9]$) for RFC and each $\sigma$ ($\sigma = [0.1 : 0.1 : 0.9]$) for RND, we generate 10 noisy testing sets for each testing set.

The model with the smallest classification error on each noisy testing set is bolded. On testing sets with random feature corruption noise, the SMLRFC is more robust to missing feature and yields smallest classification error in all noisy testing sets with different noise levels. As shown in Table 4, the SMLRFC yields 19.83% error at $p = 0.9$, while classification error of other methods are all larger than 56% (more than double to the SMLRFC). The WD, the random dropout, and the NI Gaussian do not consider missing feature at the testing phase, so their performances degrade rapidly as the added noise level increasing. Testing classification errors obtained by the WD, the random dropout, and the NI Gaussian are larger than 20% when $p = 0.7$, while the classification error obtained by the SMLRFC is 4.04% only. Due to the NI zeromark explicitly

### TABLE 3. 95% Confident Intervals $\mu \pm \delta$ of Testing Classification Error on 8 Grayscale Image Datasets Using Models Obtained by Different Training Methods (Noise Level).

| Methods | MNIST | basic | rot | bg-rand | bg-img | rot-bg-img | rect | rect-img |
|---------|-------|-------|-----|---------|--------|------------|------|----------|
| SVM$_{RF}$ [48] | 1.40 ± 0.23 | 3.03 ± 0.15 | 11.11 ± 0.28 | 14.58 ± 0.31 | 22.61 ± 0.38 | 55.18 ± 0.44 | 2.15 ± 0.13 | 24.04 ± 0.37 |
| WD | 1.43 ± 0.23 | 4.17 ± 0.18 | 12.67 ± 0.29 | 13.95 ± 0.3 | 19.48 ± 0.35 | 47.97 ± 0.44 | 5.69 ± 0.20 | 22.57 ± 0.37 |
| $N_{IG}$ | 1.18 ± 0.21 | 3.37 ± 0.17 | 11.42 ± 0.28 | 11.26 ± 0.28 | 20.05 ± 0.35 | 59.89 ± 0.43 | 2.24 ± 0.13 | 21.83 ± 0.36 |
| $p$ | 0.3 | 0.3 | 0.3 | 0.2 | 0.1 | 0.1 | 0.5 | 0.2 |
| $N_{IG}$ | 1.15 ± 0.21 | 4.13 ± 0.17 | 10.90 ± 0.27 | 9.99 ± 0.26 | 19.70 ± 0.35 | 54.57 ± 0.44 | 2.03 ± 0.12 | 22.26 ± 0.36 |
| $\sigma$ | 0.1 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Random Dropout | 1.29 ± 0.22 | 3.07 ± 0.15 | 10.00 ± 0.26 | 13.85 ± 0.30 | 19.60 ± 0.35 | 48.28 ± 0.44 | 3.83 ± 0.17 | 23.28 ± 0.37 |
| RBM-3 [48] | 1.28 ± 0.22 | 3.11 ± 0.15 | 10.30 ± 0.27 | 6.73 ± 0.22 | 16.31 ± 0.32 | 47.39 ± 0.44 | 2.60 ± 0.14 | 25.50 ± 0.37 |
| $SML_{RND}$ | 1.31 ± 0.22 | 2.92 ± 0.15 | 9.86 ± 0.26 | 11.21 ± 0.28 | 19.76 ± 0.34 | 47.31 ± 0.44 | 1.68 ± 0.11 | 22.42 ± 0.37 |
| $\sigma$ | 0.1 | 0.1 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| $SML_{RFC}$ | 1.15 ± 0.21 | 2.55 ± 0.14 | 8.44 ± 0.24 | 10.23 ± 0.27 | 17.65 ± 0.33 | 46.30 ± 0.44 | 1.55 ± 0.11 | 21.63 ± 0.36 |
| $p$ | 0.4 | 0.5 | 0.7 | 0.3 | 0.9 | 0.9 | 0.7 | 0.6 |

### TABLE 4. Classification Error (%) of Different Methods on Testing Sets of MNIST Dataset With Different RFC Noise Level ($p$).

| $D_p$ | WD | $N_{IG}$ | $N_{IG}$ | Dropout | $SML_{RND}$ | $SML_{RFC}$ |
|-------|----|---------|---------|---------|------------|-------------|
| $D_{0.1}$ | 1.74 | 1.37 | 1.37 | 1.73 | 1.54 | 1.28 |
| $D_{0.3}$ | 2.21 | 1.79 | 1.60 | 2.15 | 1.85 | 1.42 |
| $D_{0.5}$ | 3.03 | 2.38 | 1.99 | 2.94 | 2.41 | 1.59 |
| $D_{0.6}$ | 4.53 | 3.41 | 2.45 | 4.30 | 3.30 | 1.80 |
| $D_{0.7}$ | 7.42 | 5.27 | 3.44 | 7.18 | 5.02 | 2.15 |
| $D_{0.8}$ | 12.52 | 9.12 | 5.60 | 12.44 | 8.19 | 2.82 |
| $D_{0.9}$ | 21.82 | 17.03 | 10.45 | 22.64 | 14.54 | 4.04 |
| $D_{1.0}$ | 37.99 | 33.89 | 23.63 | 40.62 | 28.28 | 8.20 |
| $D_{1.5}$ | 63.56 | 66.24 | 56.04 | 67.58 | 55.94 | 19.83 |

### TABLE 5. Classification Errors (%) of Different Methods on Testing Sets of MNIST Dataset With Different RND Noise Levels ($\sigma$).

| $D_{\sigma}$ | WD | $N_{IG}$ | $N_{IG}$ | Dropout | $SML_{RND}$ | $SML_{RFC}$ |
|-------------|----|---------|---------|---------|------------|-------------|
| $D_{0.1}$ | 2.33 | 1.53 | 1.37 | 2.39 | 1.47 | 2.06 |
| $D_{0.3}$ | 3.70 | 1.81 | 3.34 | 3.82 | 1.73 | 3.32 |
| $D_{0.5}$ | 6.89 | 2.56 | 6.15 | 7.31 | 2.45 | 5.94 |
| $D_{0.6}$ | 12.39 | 3.87 | 10.87 | 13.31 | 3.79 | 9.79 |
| $D_{0.8}$ | 19.82 | 6.45 | 17.19 | 20.22 | 6.17 | 14.38 |
| $D_{0.9}$ | 27.37 | 10.72 | 24.37 | 27.81 | 9.99 | 19.89 |
| $D_{1.0}$ | 34.52 | 16.44 | 31.36 | 34.72 | 15.07 | 25.76 |
| $D_{1.5}$ | 41.17 | 23.19 | 37.88 | 41.43 | 21.39 | 31.12 |
| $D_{2.0}$ | 47.24 | 30.02 | 44.24 | 47.36 | 27.72 | 36.54 |

The model with the smallest classification error on each noisy testing set is bolded.
adds feature deletion noise to the inputs, so its performance is better than that of the WD’s, the random dropout’s, and the NI Gaussian’s. However, the classification error yielded by the NI zeromask is 6% more than that of the SML_{RFC}’s. The NI gaussian yields a smaller testing error than the SML_{RND} when noise is small (p ∼< 0.3). The SML_{RND} yields smaller testing error when p > 0.3.

On testing sets with random normal distribution noise (Table 5), the SML_{RND} yields the smallest classification error in all 9 noisy testing sets in comparison to other methods. The Ni_{G} trained using training samples with additional Gaussian noises yields comparable performance to the SML_{RND}’s when the noise level is small. The SML_{RND} significantly outperforms the Ni_{G} when noise level σ is larger than 0.5. Both the WD and the random dropout do not consider feature noise and yield much higher classification error in comparison to the NI and the SML. The Ni_{Z} and the SML_{RFC} trained on training samples with random corruption noises and the STSM term with RFC noise also yield good robustness to random normal noise and perform better than both the WD and the random dropout.

Results in Table 4 and Table 5 show that the SML is more robust to feature perturbation at the testing phase, and yields small classification error even when a large portion of features are missing or contaminated with Gaussian noises.

### B. NATURAL IMAGE CLASSIFICATION
#### BASED ON BoW FEATURES

In this section, we validate the performance of the SML on the CIFAR-10 dataset [50] (a color image dataset). The CIFAR-10 dataset consists of 60,000 color images with size 32 × 32 which is a subset of the 80 million tiny image dataset [50]. Each image is labeled by one of the ten categories. The CIFAR-10 dataset has been split into a training set with 50,000 images and a testing set with 10,000 images, respectively. In the experiment, we keep this splitting of training and testing sets while the last 10,000 images in the training set is used as validation set. So, the CIFAR-10 dataset has fixed training/validation/testing sets with 40,000/10,000/10,000 images. All training methods select optimal hyper-parameters from table 2 using the validation set.

We follow the experimental setup described in [19] to extract BoW (Bag of Word) features. Firstly, we randomly extract 400,000 6 × 6 image patches from 50,000 training images. Secondly, we apply the k-means clustering (with k = 800 centers) on these 400,000 image patches to construct a code book with 800 visual words. Thirdly, for every image in the training set and the testing set, a 6 × 6 window runs in horizontal and vertical with a step size of 1 pixel. So, we obtain 27 × 27 patches for each 32 × 32 image. Each extracted image patch is represented by a 800-dimension BoW vector where I is assigned to the entry representing the center with minimum distance from this patch and zeros are assigned for all other 799 entries of the vector. Then, each image is represented by a 27 × 27 × 800 matrix. At last, we partition this 27 × 27 × 800 matrix into four parts with equal size according to the first and second dimensions and with zero-padding. Finally, each image is represented by a n = 4 × 800 dimension vector by counting numbers of occurrences of each visual word.

1) CLASSIFICATION PERFORMANCE ON CIFAR-10 DATASET

Table 6 shows 95% confident intervals of classification errors for different methods on the CIFAR-10 dataset. We adopt the same confident interval notation as in Section V-A1. The best testing results among different methods are bolded in the table. The method proposed in [19] yields 31.4% classification error with the same experimental setup (using a SVM classifier). We adopt this method as the baseline method in this experiment. As shown in Table 6, the SML_{RFC} yields the smallest testing error (24.86%) and significantly outperforms the other five methods including the baseline method. The SML_{RFC} outperforms the baseline, the WD, the random dropout, the NI Gaussian, and the NI zeromask by 20.83%, 14.36%, 13.02%, 12.40%, and 8.37%, respectively. The WD yields the largest testing error among the four NN training methods in comparison. Compared to the WD, both the NI Gaussian and the NI zeromask improve the generalization performance of NN. Similar to experiments in Section V-A1, both NI methods improve generalization capabilities when input noise levels are small and the improvements are not as significant as the SML’s.

2) CLASSIFICATION PERFORMANCE ON CIFAR-10 DATASET WITH NOISY TESTING SAMPLES

In this section, the SML is tested on testing sets with feature perturbation at the testing phase. Similar to V-A2, we generate noisy testing sets with different noise levels for each noise type. We train NNs with the training set and test them with testing sets contaminated by feature perturbation with different noise levels to evaluate their robustness. For each method, the model yielding the best performance in Section V-B1 is adopted as the final classifier. The classification error of different methods on testing sets with different added random feature corruption noise levels and random normal distribution noises are shown in Tables 7 and 8, respectively. The best classification result for each noise level is bolded.

As shown in Table 7, the SML_{RFC} yields the smallest classification error on 9 noisy testing sets with different

| Methods       | Performance | Best Noise Level |
|---------------|-------------|------------------|
| WD            | 29.03 ± 0.89 | \( \lambda_u = 1 \) |
| NI Zeromask   | 27.13 ± 0.87 | \( p = 0.3 \) |
| NI Gaussian   | 28.38 ± 0.88 | \( \sigma = 0.05 \) |
| Random Dropout| 28.58 ± 0.89 | \( p_u = 0.5 \) |
| SML_{RND}     | 26.57 ± 0.87 | \( \sigma = 0.1 \) |
| SML_{RFC}     | 24.86 ± 0.85 | \( p = 0.8 \) |
noise levels. As the noise level increasing, performance gaps between the SML$_{RFC}$ and the other four methods increases because classification errors of SML$_{RFC}$ increase slower. For the testing set with noise level $p = 0.8$, the SML$_{RFC}$ yields 38.91% classification error while the WD, the random dropout, the NI Gaussian, and the NI zeromask yield 79.53%, 82.31%, 88.03%, and 63.81% classification error, respectively. The NI zeromask trains NN using training samples with random feature deletion, so it outperforms methods without considering feature missing noises, i.e., the WD, the random dropout, the NI Gaussian, and the SML$_{RND}$. However, the NI zeromask is limited to small noise levels, so the performance of the SML$_{RND}$ is better than that of the NI zeromask’s, especially when the testing samples contaminated by large noises, i.e. $p > 0.4$. Compared to the random dropout, the WD relieves the influence of feature deletion to some extend as shown in Table 7). However, minimizing the $L_2$ norm is too general to handle the feature corruption problem.

Similarly, as shown in Table 8, the SML$_{RND}$ yields the smallest classification error on 9 noisy testing sets with noise levels $\sigma$ from 0.1 to 0.9 with step size of 0.1 for testing sets with random normal distribution noises. The NI Gaussian trained on testing sets contaminated by random normal distribution noises outperforms the WD, the random dropout, the $N_{LZ}$, and the SML$_{RFC}$ that do not consider random normal noises. The SML$_{RFC}$ yields the smallest classification error compared to the WD, the random dropout, and the $N_{LZ}$. This shows that the SML trained by the STSM with RFC noise can also resist to RND noises.

| $p$ | WD | NI Gaussian | $N_{LZ}$ | Dropout | SML$_{RND}$ | SML$_{RFC}$ |
|-----|-----|-------------|---------|---------|-------------|-------------|
| $p_1$ | 30.49 | 29.96 | 27.87 | 30.13 | 29.61 | 25.57 |
| $p_2$ | 32.47 | 32.39 | 28.67 | 32.53 | 30.66 | 26.23 |
| $p_3$ | 35.61 | 36.08 | 29.98 | 36.02 | 31.99 | 27.25 |
| $p_4$ | 39.92 | 42.27 | 32.23 | 41.76 | 34.44 | 28.23 |
| $p_5$ | 45.90 | 52.22 | 35.72 | 49.80 | 37.84 | 29.75 |
| $p_6$ | 55.37 | 66.12 | 41.76 | 60.52 | 43.27 | 31.83 |
| $p_7$ | 67.23 | 80.39 | 50.97 | 73.01 | 52.83 | 34.50 |
| $p_8$ | 79.53 | 88.03 | 63.81 | 82.31 | 66.13 | 38.91 |
| $p_9$ | 87.91 | 99.90 | 77.48 | 87.96 | 80.14 | 53.52 |

### C. FACE RECOGNITION

In this section, we validate the performance of SML on the YaleB (Extended Yale Face Database B) [51], [52] and the UMIST (The Sheffield Face Database) [53] datasets. The YaleB dataset consists of 9 poses of 38 people and 64 illumination conditions, i.e. totally 16128 face images. In this experiment, we only select the front pose of each individual with all illumination conditions (64 images). Each face image is resized to have 32 $\times$ 32 pixels. Due to this dataset does not have a standard training/testing partition, we randomly partition the 64 faces of each individual into training/validation/testing sets with 40/10/14 images. So, we obtain training/validation/testing sets with 1520/380/532 images in each random partition. This random partition is repeated 10 times to obtain 10 different training/validation/testing partitions. The UMIST consists of 564 face images for 20 different individuals. Each individual has different number of face images and each image has different sizes. Images are cropped to preserve face information only and resized to have 32 $\times$ 32 pixels. Similar to the YaleB dataset, we randomly partition the UMNIST dataset into training/validation/testing sets based on proportions of 50%/20%/30%. This random partition is repeated 10 times to obtain 10 different training/validation/testing partitions. In the experiment, pixels are used as features for these two face datasets. Each face image is represented by a 1024 dimensional vector and the 255-level gray scale value of each pixel is divided by 255 to re-scaled to $[0, 1]$.

#### 1) CLASSIFICATION PERFORMANCE ON TWO FACE DATASETS

Each training method runs on 10 different training/testing partitions independently and selects the set of optimal hyper-parameters using the validation set. The average value and the standard deviation of classification errors and the $t$-test significance $P$ value (in brackets) of 10 independent run for each method are listed in Table 9. As shown in Table 9, both the SML$_{RFC}$ and the SML$_{RND}$ significantly improve the classification performance. The SML$_{RFC}$ and the SML$_{RND}$ yield the smallest and the second-smallest average classification errors on both the YaleB and the UMIST datasets, respectively. The SML$_{RFC}$ outperforms all other four methods significantly on the YaleB dataset with 95% confidence. On the UMIST dataset, the SML$_{RFC}$ outperforms the NI zeromask and the random dropout with 95% confidence and outperforms the WD method with 90% confidence.

| Methods | YaleB | UMIST |
|---------|-------|-------|
| WD      | 3.93 ± 0.95 (0.00)* | 1.83 ± 1.13 (0.10)* |
| NI Zeromask | 2.00 ± 0.53 (0.02)* | 2.45 ± 1.55 (0.02)* |
| NI Gaussian | 3.09 ± 0.74 (0.00)* | 1.78 ± 1.31 (0.10)* |
| Random Dropout | 3.97 ± 0.69 (0.00)* | 2.31 ± 1.20 (0.01)* |
| SML$_{RND}$ | 1.65 ± 0.26 (0.28) | 1.45 ± 1.19 (0.38) |
| SML$_{RFC}$ | 1.52 ± 0.72 | 1.07 ± 0.83 |

### TABLE 7. Classification Error of Different Methods on Testing Set of CIFAR-10 Dataset With Different RFC Noise Level ($p$).

### TABLE 8. Classification Error of Different Methods on Testing Set of CIFAR-10 Dataset With Different RND Noise Level ($\sigma$).

### TABLE 9. Average Classification Error ± Standard Deviation ($t$-test significant $P$ value) of 10 Independent Runs on YaleB and UMIST Datasets.
TABLE 10. Classification Error of Different Methods on Testing Set of YaleB Dataset With Different RFC Noise Level (\(p\)).

| \(D_p\) | WD | \(N_G\) | \(N_F\) | Dropout | \(SML_{RND}\) | \(SML_{RFC}\) |
|-------|-----|--------|--------|---------|----------------|----------------|
| \(D_{0.1}\) | 6.30 | 5.06 | 2.53 | 6.60 | 4.59 | 2.35 |
| \(D_{0.2}\) | 7.96 | 7.24 | 3.02 | 8.89 | 7.59 | 2.43 |
| \(D_{0.3}\) | 12.63 | 11.56 | 3.85 | 14.77 | 9.07 | 3.02 |
| \(D_{0.4}\) | 20.51 | 20.33 | 5.95 | 24.24 | 11.85 | 3.95 |
| \(D_{0.5}\) | 30.47 | 31.85 | 10.04 | 33.87 | 18.05 | 5.23 |
| \(D_{0.6}\) | 45.56 | 45.48 | 19.32 | 47.20 | 28.91 | 8.02 |
| \(D_{0.7}\) | 60.95 | 65.91 | 37.22 | 61.65 | 45.97 | 14.51 |
| \(D_{0.8}\) | 76.48 | 78.89 | 60.58 | 75.39 | 71.58 | 31.71 |
| \(D_{0.9}\) | 86.67 | 88.00 | 81.61 | 86.95 | 81.30 | 64.94 |

TABLE 11. Classification Error of Different Methods on Testing Set of YaleB Dataset With Different RND Noise Level (\(\sigma\)).

| \(D_\sigma\) | WD | \(N_G\) | \(N_F\) | Dropout | \(SML_{RND}\) | \(SML_{RFC}\) |
|-------|-----|--------|--------|---------|----------------|----------------|
| \(D_{0.1}\) | 10.47 | 7.33 | 5.43 | 10.06 | 6.91 | 5.64 |
| \(D_{0.2}\) | 22.28 | 12.20 | 10.19 | 22.06 | 9.09 | 11.54 |
| \(D_{0.3}\) | 38.15 | 22.10 | 17.45 | 37.78 | 12.51 | 18.89 |
| \(D_{0.4}\) | 53.89 | 33.39 | 26.07 | 51.09 | 18.99 | 26.38 |
| \(D_{0.5}\) | 64.94 | 44.81 | 35.19 | 64.20 | 28.64 | 35.70 |
| \(D_{0.6}\) | 74.16 | 55.39 | 45.00 | 71.36 | 39.77 | 45.64 |
| \(D_{0.7}\) | 78.70 | 62.16 | 53.91 | 77.96 | 48.83 | 52.61 |
| \(D_{0.8}\) | 82.20 | 70.47 | 59.03 | 82.14 | 58.02 | 59.32 |
| \(D_{0.9}\) | 85.39 | 74.67 | 66.01 | 84.57 | 64.51 | 65.10 |

2) CLASSIFICATION PERFORMANCE ON YaleB DATASET WITH NOISY TESTING SAMPLES

Due to the YaleB dataset does not have a fixed training/testing set partition, we use the dataset partition method described in Section V-C1 to obtain a training/validation/testing partition for the YaleB dataset. Each training method uses the training set to train the model and the validation set to select the set of optimal hyper-parameters. Then, we use the noisy testing set generation methods described in Section V-A2 to generate noisy testing sets by adding either random feature corruption noises or random normal distribution noises.

For each training method, we select the model with the smallest validation error as the final classifier. We range the noise level adding to the testing set from small to large, so that we obtain the performance of each method on different noise testing sets with different noise level. Classification errors of different methods for noisy testing sets with different noise levels are provided in Tables 10 and table 11.

On testing sets with random feature corruption noises, classification errors of all methods increase as the added noise level increases. However, classification errors yielded by the SMLRFC increase much slower than other methods and yields the smallest classification error on all noise levels. As shown in Table 10, the SMLRFC yields 14.81% classification error for the testing set with noise level \(p = 0.7\) while the WD, the random dropout, the NI Gaussian, and the NI zeromask yield 60.95%, 61.65%, 65.91%, and 37.21% classification errors, respectively. The WD, the random dropout, and the NI Gaussian do not consider feature missing at the testing phase, so their performances degrade faster than the NI zeromask, the SMLRND and the SMLRFC. The performance of NI zeromask degrades rapidly for testing sets with large noise levels (\(p \geq 0.5\)). This may be due to the NI zeromask method is limited to small noise levels. In contrast, the SMLRFC retains a low testing error even for testing sets with a large noises.

On testing sets with random normal distribution noises (Table 11), the SMLRND yields the best performance on 8 out of 9 noise levels. Performances of the WD and the random dropout degrade significantly even for small noise levels, i.e. \(\sigma = 0.1\). Both NI methods and both SML methods outperform the WD and the random dropout. Models trained by the NI Gaussian do not resist testing samples with large noises because the NI Gaussian is limited by small noise levels. In contrast, the SMLRND does not limit to small noises and yields the best performance on almost all noisy testing sets.

VI. CONCLUSION

We propose a general regularization method based on output stochastic sensitivity measure (STSM) in this article. The STSM measures the output deviation between each training sample and its noisy versions generated via a user defined feature perturbation strategy. The feature perturbation strategy is user defined to simulate the noise that the trained model targets to defense. We define two types of feature perturbation strategies, i.e. random feature corruption noise and random normal distribution noise, to improve the robustness of trained models. The objective function of the proposed method consists of a training error term and an STSM term. Models trained by our method with these feature perturbation rules not only have improved generalization capability but also robust to feature perturbations. Experimental results on eight grayscale image datasets, one color image dataset, and two face recognition datasets show that our method significantly outperforms several state of the art regularization techniques and achieves comparable results to deep neural networks with unsupervised pre-trained, with less training time and hyper-parameter tuning. Furthermore, the SML yields much lower classification errors compared to the WD, the NI, and the random dropout when testing sets contaminated with large noises. Our further works are to validate the performance when different feature perturbation types are combined in the training phase and to extend the SML to semi-supervise learning.

REFERENCES

[1] D. J. C. MacKay, “A practical Bayesian framework for back-propagation networks,” Neural Comput., vol. 4, no. 3, pp. 448–472, May 1992, doi: 10.1162/neco.1992.4.3.448.
[2] A. Krogh and J. A. Hertz, “A simple weight decay can improve generalization,” in Proc. Adv. Neural Inf. Process. Syst., vol. 4, Denver, CO, USA, Dec. 1991, pp. 950–957. [Online]. Available: http://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization
[3] C. M. Bishop, “Training with noise is equivalent to Tikhonov regularization,” Neural Comput., vol. 7, no. 1, pp. 108–116, Jan. 1995.
[4] R. Reed, R. J. Marks, and S. Oh, “Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter,” IEEE Trans. Neural Netw., vol. 6, no. 3, pp. 529–538, May 1995, doi: 10.1109/72.377960.
[5] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[6] G. An, “The effects of adding noise during backpropagation training on a generalization performance,” *Neural Comput.*, vol. 8, no. 3, pp. 643–674, Apr. 1996, doi: 10.1162/neco.1996.8.3.643.

[7] K. Matsuo, “Noise injection into inputs in back-propagation learning,” *IEEE Trans. Syst., Man., Cybern.*, vol. 22, no. 3, pp. 436–440, May 1992, doi: 10.1109/21.155944.

[8] X.-X. Li, D.-Q. Dai, X.-F. Zhang, and C.-X. Ren, “Structured sparse error coding for face recognition with occlusion,” *IEEE Trans. Image Process.*, vol. 22, no. 5, pp. 1889–1900, May 2013, doi: 10.1109/TIP.2013.2237920.

[9] S.-M. Huang and J.-F. Yang, “Subface hidden Markov models coupled with a universal occlusion model for partially occluded face recognition,” *IET Biometrics*, vol. 1, no. 3, pp. 149–159, Sep. 2012.

[10] Y. Deng, Q. Dai, and Z. Zhang, “Graph laplace for occluded face completion and recognition,” *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2329–2338, Aug. 2011.

[11] J. H. Shin, D. Smith, W. Swierzcz, K. Staley, J. T. Rickard, J. Montero, L. A. Kurgan, and K. J. Cios, “Recognition of partially occluded and rotated images with a network of spiking neurons,” *IEEE Trans. Neural Netw.*, vol. 21, no. 11, pp. 1697–1709, Nov. 2010.

[12] J. Lu, V. E. Ljiong, X. Zhou, and J. Zhou, “Learning compact binary face descriptor for face recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2041–2056, Oct. 2015.

[13] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian detection: An evaluation of the state of the art,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012, doi: 10.1109/tpami.2011.155.

[14] X. Zhao, Z. He, S. Zhang, and D. Liang, “Robust pedestrian detection in thermal infrared imagery using a shape distribution histogram feature and modified sparse representation classification,” *Pattern Recognit.*, vol. 48, no. 6, pp. 1947–1960, Jun. 2015.

[15] D. O. Pop, A. Rogozan, C. Chatelain, F. Nashashibi, and A. Bensrhair, “Multi-task deep learning for pedestrian detection, action recognition and time to cross prediction,” *IEEE Access*, vol. 7, pp. 149318–149327, 2019.

[16] Z. He, S. Yi, Y.-M. Cheung, X. You, and Y. Yan Tang, “Robust object tracking via key pool sparse representation,” *IEEE Trans. Cybern.*, vol. 47, no. 2, pp. 354–366, Feb. 2017.

[17] A. Globerson and S. Roweis, “Nightmare at test time: Robust learning by feature deletion,” in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, Pittsburgh, PA, USA, 2006, pp. 353–360, doi: 10.1145/1143884.1143889.

[18] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: A deep convolutional networks for visual recognition and description,” *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5017–5032, Dec. 2015.

[19] A. Coates, A. Y. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, Apr. 2011, pp. 215–223. [Online]. Available: http://www.jmlr.org/proceedings/papers/v15/coates11a/coates11a.pdf

[20] J. Yang, Y.-G. Jiang, A. G. Hauptmann, and C.-W. Ngo, “Noisy activation functions,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 3059–3068.

[21] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: Closing the generalization gap in large batch training of neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 1731–1741.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[24] S. I. Wang and C. D. Manning, “Fast dropout training,” in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 2013, pp. 118–126. [Online]. Available: http://jmlr.org/proceedings/papers/v28/wang13a.html

[25] G. E. Hinton, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&retmode=med&cmd=showdetailview&indexed=google

[26] S. Rifai, P. Vincent, X. Müller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, Bellevue, WA, USA, Jun./Jul. 2011, pp. 833–840.

[27] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1953039

[28] L. Wen, L. Gao, and X. Li, “A new deep transfer learning method based on sparse auto-encoder for fault diagnosis,” *IEEE Trans. Syst., Man, Cybern.*, vol. 49, no. 1, pp. 136–144, Jan. 2017.

[29] T. Wang, W. W. Y. Ng, M. Pelillo, and S. Kwong, “LiSSA: Localized stochastic sensitive autoencoders,” *IEEE Trans. Cybern.*, early access, Jul. 16, 2019, doi: 10.1109/TCYB.2019.2923756.

[30] R. Salakhutdinov, J. B. Tenenbaum, and A. Torralba, “Learning with hierarchical-deep models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1958–1971, Aug. 2013, doi: 10.1109/tpami.2012.269.

[31] G. Litjens, T. Kooi, B. Bejnordi, S. van der Laak, B. van Ginneken, and C. I. Sánchez, “A survey on deep learning in medical image analysis,” *Med. Image Anal.*, vol. 42, pp. 60–88, Dec. 2017.

[32] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, “Recent advances in convolutional neural networks,” *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.

[33] D. Amodei et al., “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 173–182.

[34] A. B. Nassif, I. Shahin, I. Attiti, M. Azzez, and K. Shaalan, “Speech recognition using deep neural networks: A systematic review,” *IEEE Access*, vol. 7, pp. 19143–19165, 2019.

[35] S. Jaf and C. Calder, “Deep learning for natural language parsing,” *IEEE Access*, vol. 7, pp. 131363–131373, 2019.
[46] W. W. Y. Ng, D. S. Yeung, M. Firth, E. C. C. Tsang, and X.-Z. Wang, “Feature selection using localized generalization error for supervised classification problems using RBFNN,” Pattern Recognit., vol. 41, no. 12, pp. 3706–3719, Dec. 2008.

[47] L. Bottou, “On-line learning and stochastic approximations,” in On-Line Learning in Neural Networks, D. Saad, Ed. New York, NY, USA: Cambridge Univ. Press, 1998, pp. 9–42. [Online]. Available: http://dl.acm.org/citation.cfm?id=304710.304720

[48] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, “An empirical evaluation of deep architectures on problems with many factors of variation,” in Proc. 24th Int. Conf. Mach. Learn. (ICML), Corvallis, OR, USA, 2007, pp. 473–480, doi: 10.1145/1273496.1273556.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[50] A. Torralba, R. Fergus, and W. T. Freeman, “80 million tiny images: A large data set for nonparametric object and scene recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 11, pp. 1958–1970, Nov. 2008, doi: 10.1109/tpami.2008.128.

[51] K.-C. Lee, J. Ho, and D. J. Kriegman, “Acquiring linear subspaces for face recognition under variable lighting,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 5, pp. 684–698, May 2005, doi: 10.1109/TPAMI.2005.92.

[52] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, “From few to many: Illumination cone models for face recognition under variable lighting and pose,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 23, no. 6, pp. 643–660, Jun. 2001, doi: 10.1109/34.927464.

[53] D. B. Graham and N. M. Allinson, “Characterizing virtual eigen signatures for general purpose face recognition,” in Face Recognition (NATO ASI Series), vol. 163, H. Wechsler, P. Phillips, V. Bruce, F. Soulié, and T. Huang, Eds. Berlin, Germany: Springer, 1998, pp. 446–456, doi: 10.1007/978-3-642-72201-1_25.

JINCHENG LI received the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2013. He is currently a Lecturer with the College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou. His current research interests include machine learning and pattern recognition, in particular, deep learning, image recognition, and medical diagnosis.

WING W. Y. NG (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from The Hong Kong Polytechnic University, Hong Kong, in 2001 and 2006, respectively. He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, where he is also the Deputy Director of the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information. His current research interests include neural networks, deep learning, smart grid, smart healthcare, and nonstationary information retrieval.

Dr. Ng is currently an Associate Editor of the International Journal of Machine Learning and Cybernetics. He is also the Principle Investigator of four China National Nature Science Foundation projects and a Program for New Century Excellent Talents in University from China Ministry of Education. He served as the Board of Governor for the IEEE Systems, Man and Cybernetics Society, in 2011 and 2013.

***