Supervised Deep Learning with Auxiliary Networks

Junbo Zhang[†,‡], Guangjian Tian[‡], Yadong Mu[‡], Wei Fan[‡]
[†]School of Information Science and Technology,
Southwest Jiaotong University, Chengdu 610031, China
[‡]Huawei Noah's Ark Lab, Hong Kong
jbzhang@my.swjtu.edu.cn, {tian.guangjian,mu.yadong,david.fanwei}@huawei.com

ABSTRACT

Deep learning well demonstrates its potential in learning latent feature representations. Recent years have witnessed an increasing enthusiasm for regularizing deep neural networks by incorporating various side information, such as user-provided labels or pairwise constraints. However, the effectiveness and parameter sensitivity of such algorithms have been major obstacles for putting them into practice. The major contribution of our work is the exposition of a novel supervised deep learning algorithm, which distinguishes from two unique traits. First, it regularizes the network construction by utilizing similarity or dissimilarity constraints between data pairs, rather than sample-specific annotations. Such kind of side information is more flexible and greatly mitigates the workload of annotators. Secondly, unlike prior works, our proposed algorithm decouples the supervision information and intrinsic data structure. We design two heterogeneous networks, each of which encodes either supervision or unsupervised data structure respectively. Specifically, we term the supervision-oriented network as "auxiliary network" since it is principally used for facilitating the parameter learning of the other one and will be removed when handling out-of-sample data. The two networks are complementary to each other and bridged by enforcing the correlation of their parameters. We name the proposed algorithm SUpervision-Guided AutoencodeR (SUGAR). Comparing prior works on unsupervised deep networks and supervised learning, SUGAR better balances numerical tractability and the flexible utilization of supervision information. The classification performance on MNIST digits and eight benchmark datasets demonstrates that SUGAR can effectively improve the performance by using the auxiliary networks, on both shallow and deep architectures. Particularly, when multiple SUGARs are stacked, the performance is significantly boosted. On the selected benchmarks, ours achieve up to 11.35% relative accuracy improvement compared to the state-of-the-art models.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Deep Neural Networks; Supervision; Autoencoder

1. INTRODUCTION

In recent years, learning feature representations from deep neural networks has emerged as a prominent methodology in the machine learning and data mining communities. It has gained numerous success in the fields such as computer vision [14], speech recognition [8] and natural language processing [25]. Some comprehensive surveys of recent progress are provided by Bengio in [2, 3].

Many classic deep neural networks, such as Autoencoders and Restricted Boltzmann Machines (RBMs), operate in the unsupervised manner. For example, an autoencoder pursues the optimal network parameters piloted by the intuition of data self-reconstruction. Nevertheless the deep neural networks can be also trained in a supervised manner, such as Convolutional Neural Networks (CNN) developed by LeCun et al. [16]. However, most attempts before 2006 at training supervised deep architectures failed. It turns out that the deep supervised feedforward neural network tends to yield inferior performances in terms of prediction error than shallow ones (with 1 or 2 hidden layers). Hinton's revolutionary work on Deep Belief Networks (DBN) [12] in 2006 shed a light on effective supervised deep learning. Other multi-layered deep neural networks like Stacked Denoising Autoencoders (SDAE) [27] can be efficiently pre-trained in layer-wise manner, followed by supervised back-propagation in order to fine-tune the parameters [4].

However, existing schemes for incorporating side information into deep neural networks are far from being satisfactory. For example, though achieving striking empirical results on several real-world applications, the two-step procedure adopted by DBN (*i.e.*, unsupervised pre-training followed by supervised fine-tuning) does not effectively handle sparse side information (*e.g.*, sparse similarity or dissimilarity constraints on data pairs). Moreover, the methodology of separately performing unsupervised or supervised parameter optimization also tends to converge to non-optimal solutions. Our algorithm is proposed to address the above-mentioned issues, and motivated by the recent surge in weakly-supervised extensions of the autoencoder algorithm [4, 20, 24, 25, 32]. These extensions advance the original autoencoder algorithm by adding label-specific output besides the data reconstruction [4], using recursive structure [25] or non-parametric Gaussian process [24].

In this paper, we propose a novel deep learning model, whose architecture is illustrated in Figure 1. Each layer of the deep model includes a SUpervision-Guided AutoencodeR, which is referred to as "SUGAR". SUGAR is comprised of a main network, an auxiliary network, and a bridge that connects the two networks. The main network adopts the data self-reconstruction criteria as in the autoencoder algorithm, and enforces the solutions to be of moder-

KDD'14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623618 .

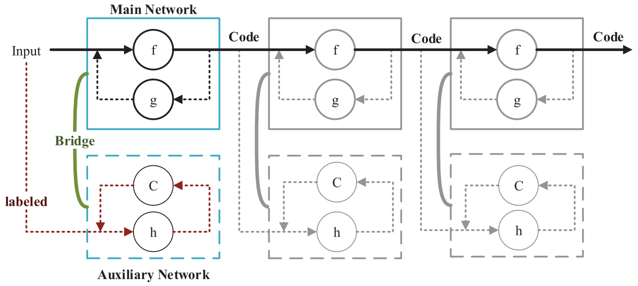


Figure 1: Deep Architecture. Each layer consists of three components: *Main Network* (solid box), *Auxiliary Network* (dotted box), and *Bridge*. f, h are two encoders, g is a decoder, C is a discernibility function. After training, the feedback decoding modules g and the encoder modules h with the corresponding classifier modules (all dashed lines) are discarded and the system is used to produce very compact representations by a feed-forward pass through the chain of encoders f.

ate sparsity in order to reduce the over-fitting risk. The auxiliary network encapsulates the supervision information. It is designed to ensure that both the similarity or dissimilarity pairwise constraints are satisfied. In the stage of parameter learning, these two networks are simultaneously optimized. The name "auxiliary networks" follows from the fact that the network will be removed when handling out-of-sample data. In other words, the auxiliary network mainly plays the role of regularizing the main network. Using the idea of auxiliary network, SUGAR is able to accomplish a seamless hybrid of unsupervised data structure discovery and sparsely-supervised learning.

Comparing to competing models, our proposed model has the following advantages:

- SUGAR employs a novel architecture to build deep networks, *i.e.*, "auxiliary networks" + "main networks" + "bridge". The auxiliary networks can be easily embedded into the conventional networks, and then provide a sparsely-supervised guidance. These two networks are complementary to each other, since the auxiliary network can learn discriminant features and the autoencoder-based main network can learn generative features. At the same time, the sparsity penalty is employed to make SUGAR more robust and efficient.

- SUGAR is very flexible and easily extendible. For example, one can replace the autoencoder (AE) term in SUGAR by many other autoencoder variants. In this paper, we tentatively explore two extensions, *i.e.*, the denoising autoencoder (DAE) [27] and the contractive autoencoder (CAE) [22].

- SUGAR can be also easily stacked as a deep learning model. After pre-training multiple hidden layers effectively, the deep stacked SUGARs can learn highly abstracted features and more meaningful representations.

2. METHODOLOGY

In this section we will elaborate on the details of the proposed algorithm.

Table 1: Mathematical notation

Symbol	Definition
N	number of data samples
N_l	number of labeled data samples
D	dimensionality of data sample
K	number of hidden units
$\mathbf{X} \in \mathbb{R}^{D \times N}$	data samples
$\mathbf{X}_l \in \mathbb{R}^{D \times N_l}$	labeled data samples
$\mathbf{x} \in \mathbb{R}^{D}$	data sample
$\widehat{\mathbf{x}} \in \mathbb{R}^{D}$	vector of reconstruction
$\mathbf{z} \in \mathbb{R}^{K}$	hidden representation
$\mathbf{h} \in \mathbb{R}^{K}$	hashing representation
$\mathbf{W} \in \mathbb{R}^{K \times D}$	weight matrix for autoencoder
$\mathbf{b} \in \mathbb{R}^{K}$	bias for encoder
$\mathbf{b}' \in \mathbb{R}^{D}$	bias for decoder
$\mathbf{P} \in \mathbb{R}^{K \times D}$	weight matrix for hashing
$\mathbf{t} \in \mathbb{R}^{D}$	bias for hashing
$\mathbf{\Omega} \in \mathbb{R}^{N_l \times N_l}$	indicator matrix
$\mathbf{I} \in \mathbb{R}^{K \times K}$	identity matrix
$\mathbb{H}^K = \{\pm 1\}^K$	K-dimensional Hamming space
$\mathcal{M}$	neighbor-pairs
$\mathcal{C}$	nonneighbor-pairs

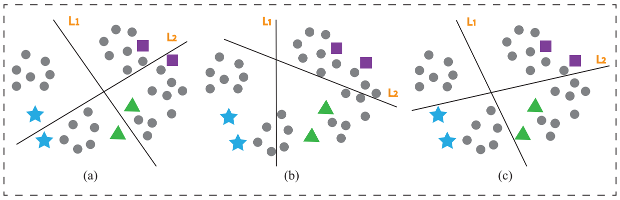


Figure 2: Illustration for the semi-supervised learning. (a) unsupervised; (b) supervised; (c) semi-supervised. Supervised learning in (b) generates reasonable solution yet does not ensure it is consistent to the underlying data distribution. The result by semi-supervised learning on (c) is more reasonable.

2.1 Problem Formulation

Suppose that there are N training samples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N} \in \mathbb{R}^{D \times N}$ where each sample $\mathbf{x}_i \in \mathbb{R}^{D}$. Among them, N_l samples are labeled, noted as $\mathbf{X}_l \in \mathbb{R}^{D \times N_l}$ and its corresponding label $\mathbf{Y}$. Table 1 lists other mathematical notation used in the paper. The traditional deep learning model such as RBM and autoencoders are learnt in the unsupervised manner which has been demonstrated to be effective for learning the latent representations. The unsupervised pre-training methods are based on the hypothesis that the marginal probability distribution of the input $P(\mathbf{X})$ contains some relevant information about the conditional probability $P(\mathbf{Y}|\mathbf{X})$. Supervised learning can remarkably improve its performance when it comes to the specific tasks. Intuitively, the labeled data comes from the joint distribution of training data $\mathbf{X}$ and its corresponding label $\mathbf{Y}$, $P(\mathbf{X}, \mathbf{Y})$, while the unlabeled data comes the marginal distribution of $\mathbf{X}$, $P(\mathbf{X})$. The joint utilization of $P(\mathbf{X}, \mathbf{Y})$ and $P(\mathbf{X})$ is supposed to be able to extract more useful information from the unlabeled data. Figure 2 illustrates a simple case in the linear feature space, where latent representations are learnt from merely unlabeled data, supervision information, or both. In order to make unsupervised learning useful, there must be connection between the joint distribution $P(\mathbf{X}, \mathbf{Y})$ and marginal distribution

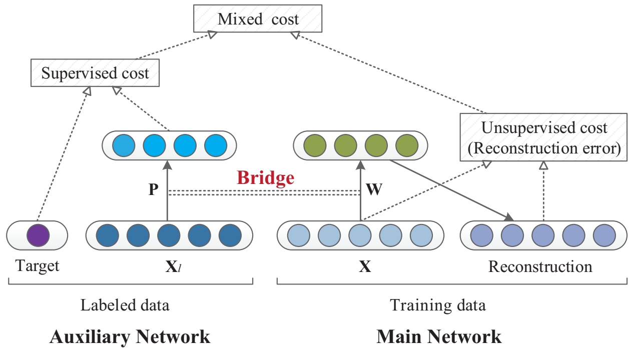


Figure 3: Architecture of SUGAR. It is trained on labeled data with the supervised learning on *Auxiliary Network* and unlabeled data with the unsupervised autoencoder on *Main Network*. They are bridged by enforcing the correlation of their parameters.

$P(\mathbf{X})$. The pure unsupervised autoencoder may not constrain this connection, so we applied an explicit sparsely-supervised model to the autoencoder to construct this connection. The flow chart of the proposed method is shown in Figure 3. The mixed model includes the following three components:

Main Network is used to reconstruct the input, *i.e.*, the unsupervised autoencoder;

Auxiliary Network is used to regularize the learnt network by pairwise similarity or dissimilarity constraints, *i.e.*, the supervised learning;

Bridge is used to connect *Main Network* and *Auxiliary Network* by enforcing the correlation of their parameters.

In the following part of this section, we first introduce the main network that is designed to capture the intrinsic data structure, then move to the auxiliary network that can regularize the main network by pairwise similarity or dissimilarity constraints among data. After that, the mixed model is introduced which involves a bridge. Finally, we describe how to learn higher level feature with deep networks.

2.2 Main Network

We propose a sparsity-encouraging variant of the classic autoencoder [23] to construct our main network. Autoencoder was introduced to address the problem of "backpropagation without a teacher", by using the input data as the teacher. It is a feed forward neural network conventionally used for reducing feature dimensionality and pre-training deep networks. For the consideration of being self-contained, we will first briefly review the key idea of autoencoder and afterwards highlight the significance of our proposed variant.

The standard autoencoder consists of two parts: an encoder and a decoder. It uses unlabeled training samples as both input and output of the neural network. The encoder is a function f that maps an input $\mathbf{x} \in \mathbb{R}^D$ to a hidden representation $\mathbf{z} \in \mathbb{R}^K$. It has the form as

$$\mathbf{z} = f(\mathbf{x}) = S_f(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{K\times D}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^K$ is a hidden bias vector, and S_f is an activation function, typically a logistic $\text{sigmoid}(\tau) = \frac{1}{1+e^{-\tau}}$ or $\tanh(\tau) = \frac{e^\tau - e^{-\tau}}{e^\tau + e^{-\tau}}$. The decoder function g maps the hidden representation $\mathbf{z}$ back to a reconstruction $\widehat{\mathbf{x}}$:

$$\widehat{\mathbf{x}} = g(\mathbf{z}) = S_g(\mathbf{W}'\mathbf{z} + \mathbf{b}'), \tag{2}$$

where $\mathbf{W}' \in \mathbb{R}^{D\times K}$ is a weight matrix, $\mathbf{b}' \in \mathbb{R}^D$ is a bias vector, and S_g is a decoder's activation function, typically either the identity (yielding linear reconstruction) or a sigmoid.

The objective of a classic autoencoder is to minimize the reconstruction error on a training set $\mathbf{X}$, *i.e.*, $\arg\min_\phi \sum_{\mathbf{x}\in\mathbf{X}} \mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}})$, with respect to the parameters $\phi = \{\mathbf{W}, \mathbf{W}', \mathbf{b}, \mathbf{b}'\}$. $\mathcal{L}$ is the loss function for the reconstruction residual. Typically it is set to be the squared error $\mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}}) = \|\mathbf{x} - \widehat{\mathbf{x}}\|^2$ when S_g is the identity function and the cross-entropy loss $\mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}}) = -(\sum_{i=1}^D x_i \log \widehat{x}_i + (1-x_i)\log(1-\widehat{x}_i))$ when S_g is the sigmoid function.

A critical downside of classic autoencoder is the over-fitting issue caused by the extremely many parameters. To attack this issue, we set $\mathbf{W}' = \mathbf{W}^T$ and enforce $\mathbf{W}$ to be sparse. Specifically, we add $\|\mathbf{W}\|_{\ell_1}$ to the objective function. In this way, we obtain

$$\arg\min_\phi \sum_{\mathbf{x}\in\mathbf{X}} \mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}}) + \lambda\|\mathbf{W}\|_{\ell_1}, \tag{3}$$

where λ is a free parameter to control the sparsity of $\mathbf{W}$, and $\|\cdot\|_{\ell_1}$ denotes the sum of the absolute values of matrix entries, *i.e.*, $\|\mathbf{W}\|_{\ell_1} \triangleq \sum_{ij}|W_{ij}|$. In the experiments, we investigate the relationship between the solution sparsity and prediction accuracies, as depicted in Figure 5(c).

2.3 Auxiliary Network

We propose to use a data similarity-preserving criterion to construct our auxiliary network. Specifically, we generate binary code for each datum and optimize the model parameters, which is known as "supervised hashing" in the literature. The inner produce of two hash codes approximately reflects the corresponding data similarity in the original feature space. Hashing is known to be more suitable for approximate similarity search for high-dimensional data. Some related methods have been proposed for learning a compact representations for the discriminative tasks, such as locality sensitive hashing [1], spectral hashing [30], and semi-supervised hashing [19, 29].

Hashing aims to map the input data into a K-dimensional Hamming space to obtain its compact representation $\mathbb{H}^K = \{\pm 1\}^K$. Suppose that there are N_l labeled training samples, so the matrix is given by $\mathbf{X}_l = \{\mathbf{x}_i\}_{i=1}^{N_l} \in \mathbb{R}^{D\times N_l}$ where each sample $\mathbf{x}_i \in \mathbb{R}^D$. Given an input $\mathbf{x}$, one can learn K hash functions $\mathbf{H} = [h_1, \cdots, h_K]$ to produce a hashing representation $\mathbf{h} \in \mathbb{R}^K$ by the form

$$\mathbf{h} = \mathbf{H}(\mathbf{x}) = \text{sgn}(\mathbf{P}\mathbf{x} + \mathbf{t}), \tag{4}$$

where $\text{sgn}(\cdot)$ is the signum function, and $\mathbf{P} \in \mathbb{R}^{K\times D}$ is a projection matrix, each column $\mathbf{p}_k \in \mathbb{R}^D$ is a projection vector, $\mathbf{t} \in \mathbb{R}^K$ is a bias vector, each element $t_k = -\frac{1}{n}\sum_{i=1}^n \mathbf{p}_k^T \mathbf{x}_i$ is equal to the mean of the projected data and zero for centered data.

In the supervised hashing, the weight matrix $\mathbf{P}$ is determined by enforcing the output of the corresponding hash functions $\mathbf{H} = [h_1, \cdots, h_K]$ to be consistent to the pre-specified side information. Suppose the side information is provided in the form of either similarity or disimilarity constraints between data pairs, we propose to maximize the following objective function:

$$\mathcal{J}(\mathbf{P}) = \sum_{k=1}^{K} \left\{ \frac{1}{|\mathcal{M}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) - \frac{1}{|\mathcal{C}|} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}} h_k(\mathbf{x}_i)h_k(\mathbf{x}_j) \right\}, \tag{5}$$

where any pair $(\mathbf{x}_i, \mathbf{x}_j)$ from set $\mathcal{M}$ denotes a neighbor-pair. $\mathbf{x}_i$ and $\mathbf{x}_j$ are either neighbors in a metric space or shared common class labels. Similarly, a pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}$ reflects the fact that $\mathbf{x}_i$ and $\mathbf{x}_j$ are far away in a metric space or have different class labels. Such an idea has been well explored in the context of discriminative subspace learning [28]. Its effectiveness in supervised deep learning was proved in prior works on neural networks such as [6]. However, our adopted formulation supports sparse supervision and is significantly more tractable.

For notation simplicity, the above objective function can be expressed in a compact matrix. For this purpose, an indicator matrix $\mathbf{\Omega}$ incorporating the pairwise labeled information from $\mathbf{X}_l \in \mathbb{R}^{D\times N_l}$ as:

$$\Omega_{ij} = \begin{cases} 1 \times \frac{1}{|\mathcal{M}|}, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}, \\ -1 \times \frac{1}{|\mathcal{C}|}, & (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

With indicator matrix $\mathbf{\Omega}$, one can rewrite Eq. (5) as

$$\mathcal{J}(\mathbf{P}) = \frac{1}{2}\,\mathbf{tr}\{\mathbf{H}(\mathbf{X}_l)\mathbf{\Omega}\mathbf{H}(\mathbf{X}_l)^T\}. \tag{7}$$

Without loss of generality, data are assumed to be zero-centered. For numerical tractability, we relax the sign of the projection values to be the signed magnitude, *i.e.*, $\mathbf{H}(\mathbf{X}_l) = \text{sgn}(\mathbf{P}\mathbf{X}_l)$ is replaced by $\mathbf{P}\mathbf{X}_l$. As in spectral hashing [30] and semi-supervised hashing [29], one can use the balancing and pairwise decorrelation constraints that can help generate good hash codes in which bits are independent and each bit maximizes the information by generating a balanced partition of the data. These constraints are replaced by the orthogonality constraints $\mathbf{P}\mathbf{P}^T = \mathbf{I}$. In summary, we intend to learn the supervised hashing by maximizing the relaxed objective function with constraints as:

$$\begin{aligned} \arg\max_{\mathbf{P}} \quad & \frac{1}{2}\,\mathbf{tr}\{\mathbf{P}\mathbf{X}_l\mathbf{\Omega}\mathbf{X}_l^T\mathbf{P}^T\}, \\ \text{subject to} \quad & \mathbf{P}\mathbf{P}^T = \mathbf{I}. \end{aligned} \tag{8}$$

2.4 Bridge

The bridge is the most important component in our model. It connects the auxiliary network to the main network by enforcing the correlation of their parameters. According to the above introduction, the main network consists in finding the parameters that minimize the reconstruction error and the auxiliary network consists in finding the parameters that maximize the empirical accuracy. We require the mixed model to inherit these two models; in other words, the learnt features are expected to be generative, such that they can produce good reconstructions, and to be discriminative, such that they can obtain the high empirical accuracy on labeled data. At the same time, we also desire the features to be sparse, which can improve robustness and efficiency of the model. Under these requirements, we connect the unsupervised term and supervised term, see Figure 3. The features $\mathbf{W}$ learnt from the unsupervised autoencoder is majorly useful for data reconstruction rather than classifier. Therefore, a good connection can make the features $\mathbf{W}$ be as consistent to the features $\mathbf{P}$ as possible where $\mathbf{P}$ is discriminative. This process can be viewed as Guidance; therefore, our model is named SUpervision-Guided AutoencodeR (SUGAR). All of these yield the following mixed objective:

$$\begin{aligned} \arg\min_{\phi,\mathbf{P}} \quad & \alpha\mathcal{J}_{AE}(\phi) + (1-\alpha)\mathcal{J}_{SH}(\mathbf{P}) + \\ & \frac{\epsilon}{2}\|\mathbf{P}-\mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_{\ell_1}, \\ \text{subject to} \quad & \mathbf{P}\mathbf{P}^T = \mathbf{I}. \end{aligned} \tag{9}$$

where ϵ is a correlation coefficient between $\mathbf{P}$ and $\mathbf{W}$, λ is sparsity penalty ratio, $\alpha \in [0, 1]$ is a guiding coefficient, and linearly blends the following two objectives:

$$\mathcal{J}_{AE}(\phi) = \sum_{\mathbf{x}\in\mathbf{X}} \mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}}) = \frac{1}{2}\sum_{\mathbf{x}\in\mathbf{X}} \|\mathbf{x} - \widehat{\mathbf{x}}\|^2, \tag{10}$$

$$\mathcal{J}_{SH}(\mathbf{P}) = -\frac{1}{2}\,\mathbf{tr}\{\mathbf{P}\mathbf{X}_l\mathbf{\Omega}\mathbf{X}_l^T\mathbf{P}^T\}. \tag{11}$$

In the autoencoder term $\mathcal{J}_{AE}$, the encoder is a linear transformation followed by a fixed element-wise nonlinearity. And the squared reconstruction error is employed.

2.4.1 Optimization Algorithm

$\|\mathbf{W}\|_{\ell_1}$ in the objective function is not differentiable at 0, which poses a problem for gradient-based methods. Though other alternative optimization schemes exist, we resort to the following simple approximation for function smoothing,

$$\|\mathbf{W}\|_{\ell_1} \approx \sum_{ij}\sqrt{W_{ij}^2 + \varpi}, \tag{12}$$

where ϖ is a pre-specified positive scalar that is negligibly small. In this way, the gradient calculation is enabled.

Another complication is from the orthogonality constraints on $\mathbf{P}$, *i.e.*, $\mathbf{P}\mathbf{P}^T = \mathbf{I}$. The constraint is difficult since it is non-convex and of high computational complexity to preserve in each iteration. Though sophisticated update schemes (*e.g.*, Crank-Nicolson-like update scheme in [31]) have been proposed to preserve such constraint, we adopt the following operation in practice, which simply sets the singular values of $\mathbf{P}$ to be all ones [13],

$$\mathbf{P} \leftarrow (\mathbf{P}\mathbf{P}^T)^{-\frac{1}{2}}\mathbf{P}. \tag{13}$$

The above operation is conducted after every gradient descent step to keep the orthogonality. We here propose to take mini-batches to update each iteration, where the true gradient is approximated by a sum over a small number of randomly training samples. During each iteration, we alternate the optimization of $\mathbf{P}$ and $\phi = \{\mathbf{W}, \mathbf{b}, \mathbf{b}'\}$ while fixing the other one. The overall algorithm is described in Algorithm 1. We name the algorithm as MSGD-SUGAR, where MSGD stands for Mini-batch Stochastic Gradient Descent.

2.5 Extensions

Nowadays, autoencoders have been widely researched and have many variants, such as [7, 8, 21, 22, 26]. The autoencoder term in our proposed model can also be replaced by many other autoencoder variants, *e.g.*, the denoising autoencoder (DAE) [27] or the contractive autoencoder (CAE) [22]. DAE encourages robustness of reconstruction $g(f(\mathbf{x}))$, whereas CAE explicitly encourages robustness of latent representation $f(\mathbf{x})$. Both learn the robust features in the unsupervised scheme with different aspects, and that, our proposed model can also be learnt in supervised scheme. Therefore, we here extend our proposed model to guide DAE and CAE.

```
Algorithm 1: MSGD-SUGAR
Input: Learning rate $\eta$, mini-batch size $m$ and $m'$, guiding coefficient
       $\alpha$, penalty coefficient $\epsilon$, sparse coefficient $\lambda$. A set of training
       data $\mathbf{X}$ where labeled data is $\mathbf{X}_l$.
begin
    Initialize the parameters $\phi = \{\mathbf{W}, \mathbf{b}, \mathbf{b}'\}, \mathbf{P}$.
    repeat
        // Fix $\phi$, Update $\mathbf{P}$
        Pick $m$ samples $\mathbf{X}'_l$ from $\mathbf{X}_l$
        Let $\mathcal{J}_1 = (1-\alpha)\mathcal{J}_{SH}(\mathbf{P}) + \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2$
                $= \frac{\alpha-1}{2m^2}\,\mathbf{tr}\{\mathbf{P}\mathbf{X}'_l\mathbf{\Omega}\mathbf{X}'^T_l\mathbf{P}^T\} + \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2$
        Update parameters $\mathbf{P}$ by
                $\mathbf{P} \leftarrow \mathbf{P} - \eta\frac{\partial\mathcal{J}_1}{\partial\mathbf{P}}$
        Orthogonal projection by
                $\mathbf{P} \leftarrow (\mathbf{P}\mathbf{P}^T)^{-\frac{1}{2}}\mathbf{P}$
        // Fix $\mathbf{P}$, Update $\phi$
        Pick $m'$ samples $\mathbf{X}'$ from $\mathbf{X}$
        Let $\mathcal{J}_2 = \alpha\mathcal{J}_{AE}(\phi) + \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_{\ell_1}$
                $= \frac{\alpha}{2m'}\sum_{\mathbf{x}\in\mathbf{X}'}\|\mathbf{x} - \widehat{\mathbf{x}}\|^2 + \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_{\ell_1}$
        Update parameters $\phi$ by
                $\phi \leftarrow \phi - \eta\frac{\partial\mathcal{J}_2}{\partial\phi}$
    until stopping criteria is met;
end
```

2.5.1 *SUGAR with DAE*

DAE can be used to learn robust representations through the additive binary making noise (some input components are randomly set as 0 in accordance with the corruption ratio ρ) or Gaussian noise with the form $\widetilde{x} = x + \zeta, \zeta \sim \mathcal{N}(0, \sigma^2 I)$, where ρ or σ are called the corruption ratio, and used to control the degree of regularization. Simply, we here use DAE to replace AE in our proposed model, then it generates a new model, *i.e.*, "SUGAR with DAE". This yields the following objective function:

$$\begin{aligned} \arg\min_{\phi,\mathbf{P}} \quad & \alpha\mathcal{J}_{DAE}(\phi) + (1-\alpha)\mathcal{J}_{SH}(\mathbf{P}) + \\ & \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_{\ell_1}, \qquad (14) \\ \text{subject to} \quad & \mathbf{P}\mathbf{P}^T = \mathbf{I}. \end{aligned}$$

where $\mathcal{J}_{DAE}(\phi) = \sum_{\mathbf{x}\in\mathbf{X}} \mathbb{E}_{\widetilde{\mathbf{x}}\sim q(\widetilde{\mathbf{x}}|\mathbf{x})}\left[\mathcal{L}(\mathbf{x}, \widehat{\widetilde{\mathbf{x}}})\right]$. Here, the expectation is over corrupted versions $\widetilde{\mathbf{x}}$ of the input $\mathbf{x}$ obtained from a corruption process $q(\widetilde{\mathbf{x}}|\mathbf{x})$ where $\widehat{\widetilde{\mathbf{x}}}$ is the reconstruction of $\widetilde{\mathbf{x}}$.

2.5.2 *SUGAR with CAE*

From the motivation of robustness to small perturbations around the training points, CAE is more strongly contracting at the training samples. Similarly, we use CAE to replace AE, and propose "SUGAR with CAE", which yields the following objective function:

$$\begin{aligned} \arg\min_{\phi,\mathbf{P}} \quad & \alpha\mathcal{J}_{CAE}(\phi) + (1-\alpha)\mathcal{J}_{SH}(\mathbf{P}) + \\ & \frac{\epsilon}{2}\|\mathbf{P} - \mathbf{W}\|_F^2 + \lambda\|\mathbf{W}\|_{\ell_1}, \qquad (15) \\ \text{subject to} \quad & \mathbf{P}\mathbf{P}^T = \mathbf{I}. \end{aligned}$$

where $\mathcal{J}_{CAE}(\phi) = \sum_{\mathbf{x}\in\mathbf{X}}\left(\mathcal{L}(\mathbf{x}, \widehat{\mathbf{x}}) + \mu\|J_f(\mathbf{x})\|_F^2\right)$. Here, μ is the contraction ratio and $\|J_f(\mathbf{x})\|_F^2 = \sum_{ij}\left(\frac{\partial z_j(x)}{\partial x_i}\right)^2$ is the Frobenius norm of the Jacobian. In the case of a sigmoid nonlinearity, this norm has a simple expression $\|J_f(\mathbf{x})\|_F^2 = \sum_{i=1}^{K}(z_i(1 - z_i))^2\sum_{i=1}^{D}W_{ij}^2$ [22] where z denotes the hidden representation of the input.

2.6 Building Deep Neural Networks

Stacking SUGARs to initialize a deep neural network works in similar way as stacking RBMs in deep belief networks (DBN) [11, 12] or standard autoencoders [4, 27]. Stacked SUGARs always consists two stages: pre-training and fine-tuning. A greedy layer-wise strategy, by Hinton [11], has been proven to be very useful in pre-training [9]. It was originally used to train a DBN one layer at a time, and it also can train each layer as an autoencoder. Unlike unsupervised pre-training, in our model, we use the supervised method with the label information into the antoencoder, *i.e.*, SUGAR. In other words, we use SUGAR to replace the standard autoencoder. The difference between SUGAR and the standard autoencoder is that the standard autoencoder only transmit the unlabeled data to next layer but SUGAR also takes the labeled data. The labeled hidden representation is used to guide the next layer as well as the current layer. The complete procedure for pre-training is shown in Figure 1.

Once a stack of encoders are built, the top level hidden representation can be used as input to a stand-alone supervised learning algorithm, *e.g.*, a logistic regression or a support vector machine classifier. In our model, as same as the strategy of [27], we also add a logistic regression layer on top of the encoders to yield a deep neural network to supervised learning. Then, all parameters of the whole system can be fine-tuned using a backpropagation technique. This stage is called as fine-tuning.

3. EXPERIMENTS

We first show the ability of SUGAR from multiple perspectives. Then we evaluate SUGARs as a pre-training strategy deep networks, using the stacking procedure that we introduced in Section 2.6. We will mainly compare the classification performance of Stacked SUGARs versus some state-of-the-art models such as Stacked Autoencoders on a benchmark of classification problems.

3.1 Datasets

We used the well-known digit classification problem and eight deep learning benchmark datasets [15].

3.1.1 *MNIST digits*

The **MNIST digits**[1] is a well-known classification problem. The entire dataset is partitioned into three parts: a training set with 50000 samples, a validation set with 10000 samples, and a test set with 10000 samples.

3.1.2 *Benchmark classification tasks*

The benchmark classification tasks[2] include 8 different datasets. Each of which contains tens of thousands of gray-level images of size 28 × 28 pixels. Among them, the datasets **Rectangles**, **Rect**$_{Img}$ and **Convex** are two-class problems. The train set (validation set) sizes of these three datasets are 1000 (200), 10000 (2000) and 7000 (1000), respectively. The following five datasets are **MNIST digits** variants with ten-class problems. The train set and validation set sizes of these five variants are 10000 and 2000,

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

[2] `http://www.iro.umontreal.ca/~lisa/icml2007.`

respectively. Each dataset has 50000 test examples. Details on the datasets are listed in Table 2 and a few example images are shown in Figure 4.

Table 2: Description of Datasets

Data Set	Train	Valid.	Test	Class
Rectangles	1000	200	50000	2
Rect$_{Img}$	10000	2000	50000	2
Convex	7000	1000	50000	2
MNIST$_{Basic}$	10000	2000	50000	10
MNIST$_{Rot}$	10000	2000	50000	10
MNIST$_{Rand}$	10000	2000	50000	10
MNIST$_{Img}$	10000	2000	50000	10
MNIST$_{RotImg}$	10000	2000	50000	10

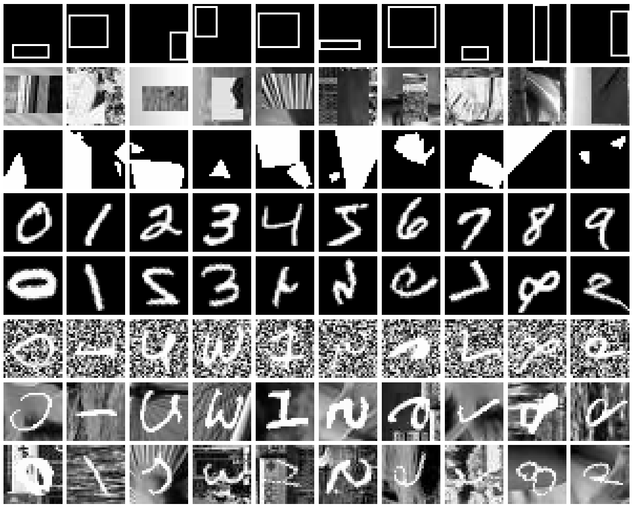


Figure 4: Samples from the various image benchmark datasets. From top row to bottom row: Rectangles, Rect$_{Img}$, Convex, MNIST$_{Basic}$, MNIST$_{Rot}$, MNIST$_{Rand}$, MNIST$_{Img}$, MNIST$_{RotImg}$.

3.2 Hyper-parameters Selection

It is non-trivial to find an optimal combination of the hyper-parameters (the number of hidden layers as well as that of hidden units in each layer, the pre-training learning rate and the fine-tuning learning rate, etc.). Fortunately, many researchers have devised various rules of thumb for choosing hyper-parameters in a neural network [17]. In our experiments, we mainly refer the strategies that used in [15]. Network parameters were trained from a random start[3], using mini-batch stochastic gradient descent[4] to perform all weight updates with a fixed learning rate (*e.g.*, 0.01 or 0.001).The guiding coefficient α shown in Eq. (9) allows us to adjust the relative contribution of the supervised guidance. We perform a grid search over the range of settings for α at intervals of 0.1. In all the datasets, we tried two values for the correlation coefficient between $\mathbf{W}$ and $\mathbf{P}$ (*i.e.*, $\epsilon = 0.000001$ or $\epsilon = 0.0000001$). The sparsity penalty ratio λ controls the sparsity of $\mathbf{W}$. We can observe the sparsity on each epoch using a small newtork architecture firstly, then empirically set a rough value and add the user-specified sparsity threshold to "stopping criteria", such as the sparsity $\geq 30\%$. We selected the value of hyper-parameters that yielded, after fine-tuning, the best classification performance on the validation set. Final classification error rate was then computed on the test set. The Python Theano Library[5] [5] is used for most of our experiments.

3.3 Performance Evaluation

3.3.1 Shallow Architecture

SUGAR with the shallow architecture (single hidden layer) is used to construct our experiments as well as the standard autoencoder on **MNIST**. We desired to know the effect by using the auxiliary network. Typically, the weight decay penalization is also used in the standard autoencoder.

1) *Effect of the guiding coefficient* α

We start to compare SUGAR and the standard autoencoder with the adjustment of the hyper-parameter α, which controls the relative importance of the supervised guidance. Both models have the same setup, *i.e.*, 500 hidden units, the $\mathrm{sigmoid}$ activation function. Figure 5(a) shows the classification error rates on three shallow neural network models, *i.e.* SUGAR, Autoencoder, and NNet (a standard single hidden layer feed-forward neural network). Obviously, SUGAR performs best in all cases. The best classification error rate of SUGAR is 1.50%. It improves 0.35% as compared with that of the standard autoencoder, whose classification error rate is 1.85%.

2) *Effect of labeled data*

In real applications, there are always fewer labeled data than unlabeled data. To assess the benefit of SUGAR on different sizes of the labeled data, we randomly sample $1000, 2000, \cdots, 10000$ labeled samples incrementally; at the same time, use all training data as unlabeled data. Concretely, taking 1000 for example, we use 1000 labeled samples and all 50000 training samples for pre-training and use 1000 labeled samples for fine-turning. We set hidden units as 100 and $\alpha = 0.2$. The same experiments were carried out in the unsupervised pre-training manner with the pure autoencoder. Figure 5(b) shows the results. We find that some supervised guidance is beneficial. And, it always performs better when increasing the labeled data.

3) *Effect of the sparsity penalty*

Here, we show the effect of the sparsity penalty. We use a subset of **MNIST** and select 1000 training samples. We here also set hidden units as 100, $\alpha = 0.2$, and fixed ϵ, but varying λ which controls the sparseness of $\mathbf{W}$. Figure 5(c) shows that the classification error firstly decreases and then increases as the sparsity increases. We also lists some related filters learnt by different sparsity, shown in Figure 6. It demonstrates our proposed model is more robust and efficient.

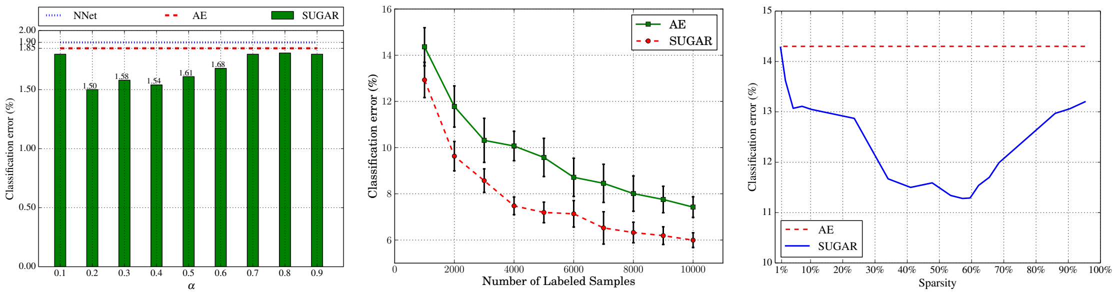
4) *Guidance to DAE and CAE*

We tried to construct the experiments to verify the guiding ability on different variants of autoencoder, *i.e.*, the denoising autoencoder and the contractive autoencoder. The autoencoder term in SUGAR can be replaced by DAE and CAE, see the details in Section 2.5. We use binary making noise in DAE and set the corruption ratio as 10%. In CAE, we set the contraction ratio as 0.1. A group of subsets of **MNIST** are chosen with varying training samples. We

[3] Weights are sampled independently form a uniform in range $[-4 \times \frac{6}{\sqrt{nin+nout}}, 4 \times \frac{6}{\sqrt{nin+nout}}]$ for the sigmoid activation that is used in our experiments where nin, $nout$ in this case are the input and output dimensions, respectively [10].

[4] In most cases, the mini-batch size is equal to 10.

[5] `http://deeplearning.net/software/theano/.`

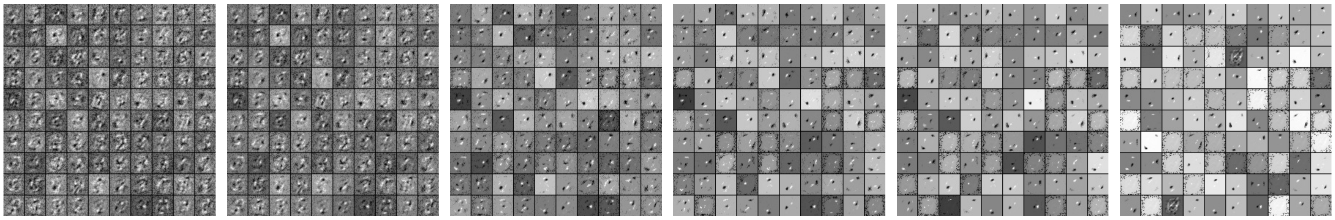


(a)

(b)

(c)

Figure 5: Experimental results on MNIST. (a) Classification error rates on the shallow neural network. NNet is a single layer neural network without pre-training and its result is taken from [4]. (b) A comparison of SUGAR ($\alpha = 0.2$) and the standard autoencoder versus the number of labeled samples. Error Bars show 95% confidence interval. (c) Classification error rates on the varying sparsity in SUGAR.



(a) 10%

(b) 25%

(c) 40%

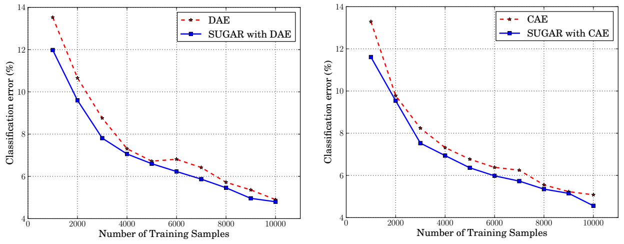
(d) 50%

(e) 60%

(f) 70%

Figure 6: Filters learnt by SUGAR from MNIST with various sparsity of $\mathbf{W}$. Filters at the same position in all six images are related here, but only by the fact that SUGAR is started from the same random initialization point in the parameter space.

also set hidden units as 100 and $\alpha = 0.2$. As shown in Figure 7, it demonstrates that the auxiliary network can improve DAE and CAE effectively, and our proposed model is flexible and can be used with other autoencoder types.



(a) DAE vs. SUGAR with DAE

(b) CAE vs. SUGAR with CAE

Figure 7: Guiding ability on autoencoder variants

3.3.2 Deep Architecture

We stacked multiple SUGARs as deep neural networks and tested classification performances on the eight benchmark datasets. In detail, 1, 2, 3 hidden layers are noted as SUGAR-1, SUGAR-2 and SUGAR-3, respectively. We used a relatively small architecture (500 or 1000 units in the first and second hidden layer and 1000 hidden units in the third layer) in most of datasets. The guiding coefficient α is an important hyper-parameter and it controls the relative importance of supervised guidance. In Section 3.3.1, we have shown that different α almost can improve the classification accuracy of an autoencoder through the grid search on $[0, 1]$. However, a full grid search is always time-consuming and even infeasible. Here, we use bisection search to replace grid search to determine α. It needs only several times to find a near optimal α. For the sparsity, we start a smaller sparsity penalty ratio (such as $\lambda = 0.0001$), and observe the sparsity of $\mathbf{W}$ and adjust λ to the proper value. Then, we add the sparsity requirement to "stopping criteria" (the sparsity $\geq 50\%$).

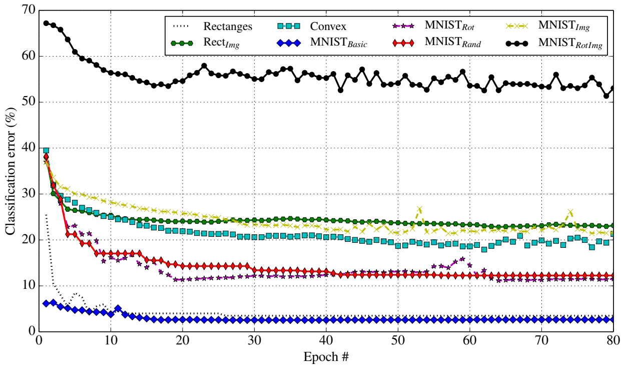


Figure 8: Validation error rates of SUGAR-3 on eight benchmark datasets. It needs only about 20 epochs to obtain almost the best validation error rate in most cases.

Figure 8 shows the error rates on validation sets in the stage of fine-tuning of SUGAR-3. It is easy to see that it needs only about 20 epochs to obtain almost the best validation error rate, which demonstrates our proposed model can provide an effective pre-training. Final classification error rate was we computed on the test set by selected parameters that yielded the best validation error rate. SUGARs with 1, 2 and 3 layers (SUGAR-1, SUGAR-2 and SUGAR-3) and SAA-3 (the deep autoencoders model) are compared in Figure 9. As the depth of the neural network increases, the classification accuracy also increases. It demonstrates our pro-

Table 3: Classification error rates on the benchmark datasets (error rates are in %). Models: SVM-RBF: SVM with RBF kernels. SVM-Poly: SVM with polynomial kernels. NNet: (MLP) Feed-forward neural net. GSM: Gated softmax classifier. NonGSM: Non-factored gated softmax classifier. SAA-3: Three-layer stacked auto-associator. RBM: Restricted boltzmann machine. SUGAR-3: Three-layer stacked SUGAR. The results of SVM-RBF, SVM-Poly, NNet, SAA-3 and RBM are taken from [15], GSM and NonGSM results from [18]. The best results obtained by all these models are marked in bold.

Dataset/Model:	SVM-RBF	SVM-Poly	NNet	GSM	NonGSM	SAA-3	RBM	SUGAR-3
Rectangles	02.15	02.15	07.16	0.83	**0.56**	02.41	04.71	03.49
Rect$_{Img}$	24.04	24.05	33.20	**22.51**	23.17	24.05	23.69	**22.55**
Convex	19.13	19.82	32.25	17.08	21.03	18.41	19.92	**17.00**
MNIST$_{Basic}$	**03.03**	03.69	04.69	03.70	03.98	03.46	03.94	03.47
MNIST$_{Rot}$	11.11	15.42	18.11	11.75	16.15	10.30	14.69	**9.53**
MNIST$_{Rand}$	14.58	16.62	20.04	10.48	11.89	11.28	**09.80**	11.40
MNIST$_{Img}$	22.61	24.01	27.41	23.65	22.07	23.00	**16.15**	20.65
MNIST$_{RotImg}$	55.18	56.41	62.16	55.82	55.16	51.93	52.21	**49.40**
Average	18.98	20.27	25.63	18.23	19.25	18.11	18.14	**17.19**

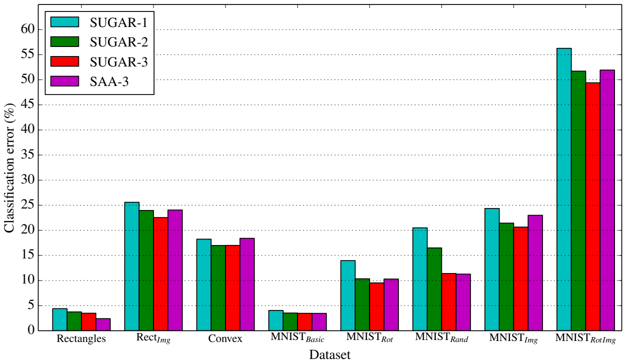


Figure 9: Classification error rates on eight benchmark classification tasks. SUGAR-3 appears to achieve performance superior or equivalent to SAA-3 on all problem except *Rectangles*. Notably, SUGAR-2 outperforms SAA-3 in three of the eight datasets.

posed model can learn more meaningful representations with deep neural networks. In most cases, stacking 3 layers of SUGAR seems to be better than stacking 3 layers of autoencoders in SAA-3.

Table 3 reports the classification perfomance obtained on eight deep learning benchmark datasets using SUGAR-3 as well as some state-of-the-art models. It indicates that SUGAR-3 performs well on all datasets. We see that SUGAR-3 outperforms the baseline SVM-RBF, as well as SVM-Poly in six out of 8 cases (except for Rectangles and MNIST$_{basic}$). The similar results can be found on the baseline GSM and NonGSM. It is among the best (within 0.04 tolerance), or the best performer, in four out of 8 cases. Intuitively, we also gave the average classification error rates of there models on all datasets. It is relatively from 1.12% up to 11.35% higher classification accuracy than these models, which demonstrates that the proposed model has a best classification ability in average.

4. CONCLUSIONS

To the best of our knowledge, most of the existing deep learning methods operate in the purely unsupervised manner or supervised manner. In this paper, we proposed a novel supervised deep learning model. Each layer includes a SUGAR, which is consisting of a main network, an auxiliary network, and a bridge. SUGAR regularizes the learnt networks by pairwise similarity or dissimilarity constraints among data. At the same time, the sparse regularization is employed, all of which make SUGAR produce good, sparse and robust representations, and therefore effectively improve deep networks with higher accuracy. The superiority of SUGAR is confirmed on MNIST digits and eight benchmark classification tasks. Further, SUGAR can be easily extended and used with the denoising autoencoder or the contractive autoencoder. It is demonstrated to provide a good guidance. Finally, the comprehensive experiments on the deep model show that it achieves performance superior to the existing state-of-the-art models on four out of eight datasets. All of these demonstrate that SUGAR can learn good feature representations for classification tasks.

Acknowledgments

The work is partly supported by a grant from China 973 Fundamental R&D Program (No.2014CB340304).

5. REFERENCES

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, Jan. 2008.

[2] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] Y. Bengio. Deep learning of representations: Looking forward. In A.-H. Dediu, C. Martín-Vidie, R. Mitkov, and B. Truthe, editors, *Statistical Language and Speech Processing*, volume 7978 of *Lecture Notes in Computer Science*, pages 1–37. Springer Berlin Heidelberg, 2013.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[5] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546 vol. 1, June 2005.

[7] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[8] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. E. Hinton. Binary coding of speech spectrograms using a deep auto-encoder. In *INTERSPEECH*, pages 1692–1695. Citeseer, 2010.

[9] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, Mar. 2010.

[10] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[12] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[13] A. Hyvärinen, P. O. Hoyer, and M. Inki. Topographic independent component analysis. *Neural computation*, 13(7):1527–1558, 2001.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[15] 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 *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 473–480, New York, NY, USA, 2007. ACM.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 1998.

[18] R. Memisevic, C. Zach, M. Pollefeys, and G. E. Hinton. Gated softmax classification. In *Advances in Neural Information Processing Systems*, pages 1603–1611, 2010.

[19] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3344–3351, June 2010.

[20] M. A. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 792–799, New York, NY, USA, 2008. ACM.

[21] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot. Higher order contractive auto-encoder. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Volume Part II*, ECML PKDD'11, pages 645–660, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986.

[24] J. Snoek, R. P. Adams, and H. Larochelle. Nonparametric guidance of autoencoder representations using label information. *J. Mach. Learn. Res.*, 13(1):2567–2588, Sept. 2012.

[25] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[26] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1096–1103, New York, NY, USA, 2008. ACM.

[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.*, 11:3371–3408, Dec. 2010.

[28] F. Wang and C. Zhang. Feature extraction by maximizing the average neighborhood margin. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[29] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431, 2010.

[30] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2008.

[31] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[32] Y. Yang, G. Shu, and M. Shah. Semi-supervised learning of feature hierarchies for object detection in a video. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1650–1657. IEEE, 2013.