Visual Representation and Classification by Learning Group Sparse Deep Stacking Network

Jun Li, *Member, IEEE*, Heyou Chang, Jian Yang, *Member, IEEE*, Wei Luo, and Yun Fu, *Senior Member, IEEE*

***Abstract*— Deep stacking networks (DSNs) have been successfully applied in classification tasks. Its architecture builds upon blocks of simplified neural network modules (SNNM). The hidden units are assumed to be independent in the SNNM module. However, this assumption prevents SNNM from learning the local dependencies between hidden units to better capture the information in the input data for the classification task. In addition, the hidden representations of input data in each class can be expectantly split into a group in real-world classification applications. Therefore, we propose two kinds of group sparse SNNM modules by mixing l_1-norm and l_2-norm. The first module learns the local dependencies among hidden units by dividing them into non-overlapping groups. The second module splits the representations of samples in different classes into separate groups to cluster the samples in each class. A group sparse DSN (GS-DSN) is constructed by stacking the group sparse SNNM modules. Experimental results further verify that our GS-DSN model outperforms the relevant classification methods. Particularly, GS-DSN achieves the state-of-the-art performance (99.1%) on 15-Scene.**

***Index Terms*— Deep learning, stacking network, sparse representation, image classification.**

I. Introduction

DEEP stacking networks (DSNs) have received an increasing attention over the past five years due to its success in speech recognition, image classification and information retrieval [5], [8], [9]. The DSN architecture is built on simplified neural network modules (SNNM) [6]. SNNM constructs a non-linear mapping from an input layer to a hidden layer by using a lower-layer weight matrix **W** and a sigmoid activation function, and the hidden layer is linearly mapped to an output layer by a upper-layer weight matrix **U**. Learning **U** could be carried out necessarily involving large-batch training since it can be formulated as a convex optimization problem with a closed-form solution [6]. This provides a reasonable solution to handle the insurmountable scalability problem when facing the fast increasing data [8]. Despite DSN's success in those tasks, its framework suffers from several drawbacks.

Firstly, the non-linear operation of the hidden layer in the traditional SNNM module is implemented by only using the sigmoid function [9]. Sigmoid has a number of disadvantages although it has been widely applied into neural network models [36], [41]. For instance, under the slow training process with random initialization, the parameter solution may stay at a bad local solution, which leads to a poor performance [15]. There are several popular types of activation functions. One is the *hyperbolic tangent*, which has been used in deep neural networks. Unfortunately, the hyperbolic tangent also suffers from the same problems as the sigmoid function. The other function is the *rectifier linear unit* (ReLU), which can be thought of as an exponential number of linear models that share parameters since N ReLU units can create 2^N regions on a surface of a hypersphere[1] [38]. Moreover, ReLU often trains faster and is quite useful for image classification and information retrieval [16], [33]. However, ReLU is non-differentiable at zero. To investigate whether ReLU hurts optimization, a smooth activation function, soft-plus $soft(a) = log(1 + e^a)$, is also considered.

Secondly, sparse representation plays a key role in the architecture of many neural networks [33]. Sparsity can typically be obtained by adding sparse regularization to form the objective function [30]. The sparse regularization allows the neural networks to be trained on data sets with limited size without severe over-fitting [12]. Moreover, Thom *et al.* [47] proved that in a single hidden layer neural network, sparse activity (and sparse connectivity) can improve classification capabilities by using a sparseness-enforcing projection operator (activation function or transfer function). Furthermore, there exists a sparse activation phenomenon in neuroscience because it has more than 90% silent neurons, which are not activated [42]. However, the conventional SNNM module does not encourage the sparse representation.

Thirdly, hidden units without any connections in the SNNM module result in independent hidden representations [21]. However, this is not the case in real-world applications. For example, the connection weights among hidden units

Manuscript received April 23, 2016; revised December 26, 2016, April 9, 2017, and September 16, 2017; accepted October 5, 2017. Date of publication October 23, 2017; date of current version November 9, 2017. This work was supported by the National Science Foundation of USA, IIS award, under Grant 1651902. The work of J. Yang was supported by the National Science Foundation of China under Grant 91420201, Grant 61472187, Grant 61233011, and Grant 61373063. The work of W. Luo was supported by the National Science Foundation of China under Grant 61702197. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Weisheng Dong. *(Corresponding author: Jun Li.)*

J. Li and Y. Fu are with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: junl.mldl@gmail.com; yunfu@ece.neu.edu).

H. Chang and J. Yang are with the School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: csjyang@njust.edu.cn).

W. Luo is with the College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510000, China (e-mail: cswluo@scau.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The supplementary PDF discusses three problems with the ReLU activation function and provides additional data in the form of graphs and tables. The total size of the file is 47.5 MB. Contact junl.mldl@gmail.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2765833

[1] Suppose the hypersphere is at least N-dimensional.

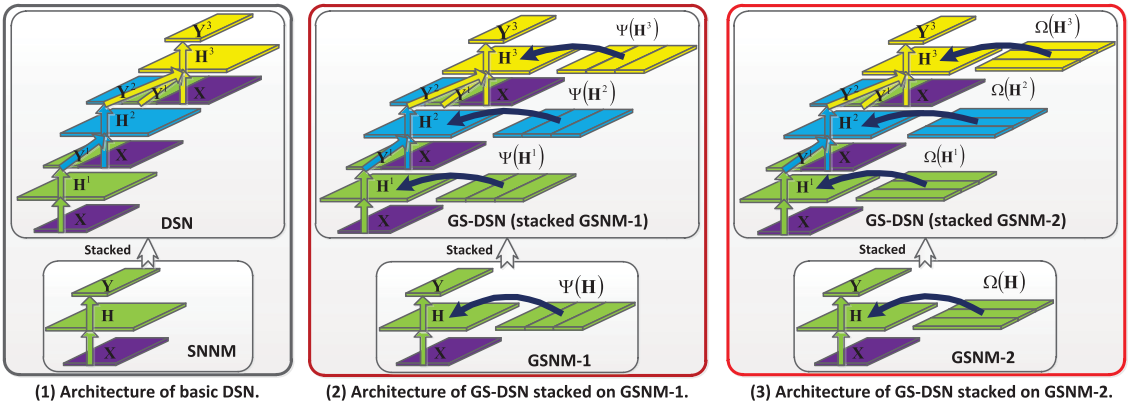


Fig. 1. Illustrations of deep sacking models. **(1)** shows the architecture of basic DSN, which forms the input for the higher module by using the original input and the outputs of all the lower modules [9]. **(2)** plots the architecture of GS-DSN stacked on GSNM-1, which divides the hidden units into non-overlapping groups. **(3)** illustrates the architecture of GS-DSN stacked on GSNM-2, which splits the hidden representations of every class into a group. In fact, **(2)** and **(3)** indicate that GSNM-1 splits the number of hidden nodes into subsets, while GSNM-2 splits the dataset into subsets. The stacking operations of GS-DSN stacked on GSNM-1 and GSNM-2 are exactly the same as that for the basic DSN.

in neural networks (e.g. deep Boltzmann machines [41] and recurrent sparse auto-encoders [40]) are used to describe the dependence among the hidden units; there is also a function between horizontal connections in the primary visual cortex (V1) in terms of a prior over natural images [14]. The resulting states of the hidden units in Boltzmann machines can exhibit statistical dependencies to better model the input data [36], [41]. If SNNM directly increases the connections among the hidden units, then it becomes a recurrent neural network. But, training the weight parameters in recurrent neural network is difficult because of the "vanishing/exploding gradients" phenomenon [3]. Fortunately, promoting local dependencies among the hidden units without the connections can be implemented by dividing them into non-overlapping groups. In the SNNM module, the local dependencies can be done by using a l_1/l_2-norm upon the outputs of hidden units for classification tasks, also known as Group Sparse SNNM Module-1 (GSNM-1).

Lastly, there is a desired characteristic that the representations are similar within-class but diverse between-class for classification tasks [2], [20], [54]. Unfortunately, it is difficult to use GSNM-1 to capture this characteristic since it cannot increase diversity between-class. To learn the representations with the characteristic in the SNNM module, we use l_2 regularization to penalize the representations of images in the same class, and utilize l_1 regularization to compete with representations of images in different classes for increasing diversity between-class. In the SNNM module, therefore, the hidden representations of different classes can be split into different groups [20], [59] by using l_1/l_2 regularization, called as Group Sparse SNNM Module-2 (GSNM-2).

This paper uses Group Sparse Deep Stacking Networks (GS-DSN) for image classification. In Fig. 1 **(2)** and **(3)**, we illustrate the modular architectures of two kinds of GS-DSN constructed by stacking three complete blocks. The two kinds of GS-DSN are obtained by respectively stacking the GSNM-1 and GSNM-2, which are performed by adding the group sparse penalty (l_1/l_2 regularization) into the SNNM modular architecture. In GSNM-1 and GSNM-2, we consider three activation functions,[2] the ReLU function, the soft-plus function, and the sigmoid function. The proposed GS-DSN has the following advantages:

1) Unlike other sparse coding techniques (e.g. LLC [53] and LC-KSVD [25]), GSNM-1 and GSNM-2 can directly learn the projective dictionary matrix, which results in a quick inference since the sparse representations are easily computed by only using a matrix multiplication and a non-linear activation function.
2) Compared with the SNNM Module, the hidden units in GSNM-1 can capture their the local dependencies by dividing them into non-overlapping groups. In GSNM-2, the hidden representations within different classes can be split into separate groups for learning group sparse representations of samples in each class. GSNM-1 and GSNM-2 are used to build GS-DSN, which still retains the scalable structure of DSN.

We perform several experiments on Extended YaleB, AR, 15-Scene, and Caltech101 datasets to demonstrate the effectiveness of GS-DSN for image classification. Experimental results further verify that our GS-DSN model achieves better recognition accuracy than other benchmark and similar methods. It is important to note that we get a 99.1% classification result on the 15-Scene dataset.

This paper is a significant extension of our conference publication [31], which only investigates hidden units divided into non-overlapping groups, and uses a simply stacking operation, which forms the input for the higher module by using the original input and the output of the lower module [7], [21]. In contrast, we consider another stacking scheme, where the input of higher module combines the raw data with outputs of all lower modules in Fig. 1. We also study another kind of group sparse modules in this extension shown in Fig. 1 **(3)**. It splits the hidden representations of every class into a group. Extensive experimental results demonstrate in more

[2]According to the previous discussions in the second paragraph, soft-plus is used to verify the advantages of ReLU, and tanh suffers from the same problems (such as quite slow convergence, and poor results) as the sigmoid function. Thus, we consider ReLU, soft-plus, and sigmoid.

comparative studies. In addition, we discuss the convergence of the ReLU activation function in supplementary material (I).

The rest of this paper is organized as follows: Section II reviews the sparse representation and its applications in image classification. In Section III we review the Deep Stacking Network. We study two kinds of group sparse modules, which are then stacked into GS-DSN in Section IV. Experiments are presented in Section V to demonstrate the effectiveness of the proposed method for image classification. Finally, we conclude this paper in Section VI.

II. Related Work

In this section, we will review deep learning and sparse representation since they have been successfully and extensively used in image classification tasks.

Deep learning methods [12] aim to learn feature hierarchies with features from higher levels composed from lower level features (e.g. deep convolutional neural network [29], deep belief networks [18], stacked denoising autoencoders [37], [49] and DSN [7]). In particular, the convolutional neural network [27] and the deep convolutional activation feature (DeCAF) [10] have achieved competition-winning numbers on large benchmark datasets, such as ImageNet. Recently, Hybrid-convolutional neural network (Hybrid-CNN) [57] was used to learn deep features for scene recognition tasks by combining the training set of a new scene-centric dataset (with over 7 million labeled pictures of scenes) and the training set of ImageNet. In this paper, we focus on another deep model–DSN, which stacks many layers of the SNNM modules [7].

The DSN architecture was inspired by a stacking scheme [51] that worked by learning a high-level classifier to combine the predictions (outputs) of multiple base-level classifiers. Sigletos *et al.* [43] saw that the stacking scheme was consistently effective in the context of information extraction, performing better than the best base-level classifiers. The success of the stacking scheme arose from its ability to harness the diversity in the predictions of base-level classifiers. DSN was previously called as a Deep Convex Network or DCN [7] to highlight its convex nature in the principal learning algorithm for learning the deep networks. Recently, DSN has also been extended to a tensor formation named Tensor-DSN (T-DSN) [21] to capture the higher-order relationships among the pixels in the image data. However, DSN and T-DSN cannot learn group representations from natural images.

Besides deep networks, many sparse coding methods are applied to image classification. Sparse coding represents an input signal image by a sparse linear combination of elementary atoms from an over-complete dictionary [11]. By employing the entire set of training samples as the dictionary, Wright *et al.* [52] exploited discriminative sparse coding for face recognition. By incorporating label information, the label consistent K-SVD (LC-KSVD) [25] and the discriminative, structured low-rank sparse representations (DSLRR) [56] were proposed to learn an image representation for natural image classification. However, in many real-world problems, dictionary elements (or representation coefficients) are related in some complex manner. For example, in order to use the same dictionary words to encode all images of the same class, Bengio *et al.* [2] proposed group sparse coding by using l_1/l_2-norm regularization allowing the representation coefficients of images in the different classes to be naturally designated between different groups [20], [59]. Learning structured dictionaries was implemented by employing a tree-structured sparse regularization to exploit possible relationships among dictionary atoms [22]. Kim *et al.* [26] proposed tree-guided group representation coefficients for estimating such structured sparsity under multi-response regression, and Szlam *et al.* [46] presented a group structured sparse coding (GSSC) that used a tree structure for inference in an object recognition system. Moreover, Chen *et al.* [4] proposed graph-guided representation coefficients for structured multi-task regression.

The conventional sparse coding models often spend expensive computation time in the inference process (e.g. calculating the sparse representation coefficients), even if they employ the labels to optimize discriminative and compact dictionaries [1], [25], [56]. In contrast, our GSNM-1 and GSNM-2 (the basic modules of GS-DSN) can learn group sparse representations and quickly calculate the hidden representation by only using a matrix multiplication and a non-linear activation function. In Section V, we will show that our model outperforms relevant classification methods.

III. Deep Stacking Network

The architecture of DSN was firstly proposed in the literature [7], and shown in Fig. 1 **(1)**. It was constructed by stacking the basic SNNM modules. We mathematically describe the DSN architecture as follows.

Let a target matrix be denoted as $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_i, \cdots, \mathbf{t}_N] \in \mathbb{R}^{C\times N}$, where $\mathbf{t}_i = [t_{1i}, \cdots, t_{ji}, \cdots, t_{Ci}]^T$, C is the number of classes (labels), and N is the number of the training samples. Let an input data matrix be denoted as $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_N] \in \mathbb{R}^{D\times N}$, where $\mathbf{x}_i = [x_{1i}, \cdots, x_{ji}, \cdots, x_{Di}]^T$, D is the dimensionality of input vector. In the basic SNNM module, the number of the units is denoted by L in the hidden layer. Let $\mathbf{W} \in \mathbb{R}^{D\times L}$ be lower-layer weights from the input layer to the hidden layer. Let $\mathbf{U} \in \mathbb{R}^{L\times C}$ be upper-layer weights from the hidden layer to the upper-layer. The output of the upper-layer is computed by $\mathbf{Y} = \mathbf{U}^T\mathbf{H}$ and $\mathbf{H} = \sigma(\mathbf{W}^T\mathbf{X}) \in \mathbb{R}^{L\times N}$, where the sigmoid function is $\sigma(a) = 1/(1 + e^{-a})$ [6], [7]. The weight matrices $\mathbf{U}$ and $\mathbf{W}$ are trained by minimizing the following square error

$$\min_{\mathbf{U},\mathbf{W}} f_{dsn} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|_\mathbf{F}^2 + \alpha\|\mathbf{U}\|_\mathbf{F}^2, \tag{1}$$

with a regularization parameter α. Clearly, by using the ridge regression, a closed-form solution of $\mathbf{U}$ is written as

$$\mathbf{U} = (\mathbf{H}\mathbf{H}^T + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{T}^T. \tag{2}$$

To minimize the square error in (1) by a gradient descent algorithm [7], [32], the gradient of $\mathbf{W}$ in the SNNM module is derived as

$$\frac{\partial f_{dsn}}{\partial \mathbf{W}} = 2\mathbf{X}\Big[\mathbf{H}^T \circ (\mathbf{1} - \mathbf{H}^T) \circ (\mathbf{U}\mathbf{U}^T\mathbf{H} - \mathbf{U}\mathbf{T})^T\Big], \tag{3}$$

where a matrix of all ones is denoted by $\mathbf{1}$, and a element-wise multiplication is denoted as $\circ$.

The closed-form solution (2) accentuates the "convex" role to train $\mathbf{U}$ in each SNNM module [6]. Many SNNM modules are often used to form a deep model by a stacked operation. In particular, the input units of a higher SNNM module can be constructed by cascading the raw input and the output units of the lowest SNNM module. DSN has recently been extended to T-DSN by using two parallel branches of hidden representations, which combine them bi-linearly to produce the predictions [21]. Moreover, T-DSN still retains the computational advantage of the DSN in parallelism and scalability during training all parameters.

IV. Group Sparse Deep Stacking Network

GS-DSN, described in Fig. 1 **(2)** and **(3)**, is a group sparse variant of DSN. The stacking operation of GS-DSN is the same as DSN [9]. The general paradigm uses the original input and the outputs of all the lower modules to construct the expanded "input" of the higher module. Our modular architecture in GS-DSN is different from that in DSN. We propose two group sparse modular architectures (GSNM-1 and GSNM-2) in GS-DSN, while DSN only considers a simple SNNM modular architecture (SNNM). The first one divides hidden units into non-overlapping groups and the second one splits hidden representations of samples of different classes into different groups. GSNM-1 and GSNM-2 are finally stacked into GS-DSN.

The output of the upper-layer is calculated by $\mathbf{Y} = \mathbf{U}^T\mathbf{H}$, and the output of the hidden layer (hidden representation) is computed by

$$\mathbf{H} = \phi(\mathbf{W}^T\mathbf{X}) \in \mathbb{R}^{L\times N}, \tag{4}$$

where $h_{j,i}$ denotes a state of j-th hidden unit of the i-th sample, $\phi(a)$ is an activation function (e.g., sigmoid, soft-plus, and ReLU), N is the number of the training samples, and L is the number of the hidden units.

A. GSNM-1: Divide Hidden Units Into Non-Overlapping Groups

Let a set of all hidden units be denoted as $\mathcal{H} = \{1, 2, \cdots, L\}$. GSNM-1 evenly and sequentially divides $\mathcal{H}$ into G groups, where G is the number of groups. Let the g-th group be denoted by Γ_g, where $\mathcal{H} = \bigcup_{g=1}^{G}\Gamma_g$ and $\bigcap_{g=1}^{G}\Gamma_g = \varnothing$. So, $\mathbf{H} = [\mathbf{H}_{\Gamma_1,:}\cdots\mathbf{H}_{\Gamma_g,:}\cdots\mathbf{H}_{\Gamma_G,:}]$, where $\mathbf{H}_{\Gamma_g,:} = [h_{j,i}]$ $(j \in \Gamma_g, 1 \le i \le N)$ are the hidden representations that belong to the g-th group Γ_g. For example, given $L = 500$ and $G = 5$, we have $\mathcal{H} = \bigcup_{g=1}^{5}\Gamma_g = \{1, 2, \cdots, 100\}\bigcup\{101, 102, \cdots, 200\}\bigcup\{201, 202, \cdots, 300\}\bigcup\{301, 302, \cdots, 400\}\bigcup\{401, 402, \cdots, 500\}$ and $\mathbf{H} = [\mathbf{H}_{\Gamma_1,:}\mathbf{H}_{\Gamma_2,:}\cdots\mathbf{H}_{\Gamma_5,:}]$.

The weight matrices $\mathbf{U}$ and $\mathbf{W}$ are trained by minimizing the regularized squares error

$$f_{sdsn1} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|_{\mathbf{F}}^2 + \alpha\|\mathbf{U}\|_{\mathbf{F}}^2 + \beta\Psi(\mathbf{H}), \tag{5}$$

where α and β are the regularization parameters of $\mathbf{U}$ and $\Psi(\mathbf{H})$, and $\Psi(\mathbf{H})$ is an imposed penalty over $\mathbf{H}$. Typically, enforcing sparsity on each representation is done by using the l_1-norm as the following form

$$\Psi(\mathbf{H}) = \sum_{i=1}^{N}\|\mathbf{H}_{:,i}\|_1, \tag{6}$$

where $\mathbf{H}_{:,i}$ is the hidden representation given i-th sample.

However, the SNNM module using l_1 sparse regularization cannot capture the local dependencies between hidden units. To incorporate the local dependencies, the set $\mathcal{H} = \{1, 2, \cdots, L\}$ is evenly divided into non-overlapping groups Γ_g for restraining the dependencies within these groups and competing with each other [36]. Fortunately, a mixed-norm regularization (l_1/l_2-norm) can be applied into the hidden units to achieve group sparse representations. Following the group sparse representation in [2] and [36], l_1/l_2-norm is considered as the following form

$$\Psi(\mathbf{H}) = \sum_{g=1}^{G}\|\mathbf{H}_{\Gamma_g,:}\|_{\mathbf{F}}, \tag{7}$$

where $\mathbf{H}_{\Gamma_g,:} = [h_{j,i}]$ $(j \in \Gamma_g, 1 \le i \le N)$ are the hidden representations that belong to the g-th group Γ_g.

1) Learning Weights- Algorithm 1: Given the fixed $\mathbf{W}$, $\mathbf{H}$ is uniquely determined. Then, $\mathbf{U}$ is easily solved by minimizing the following square error

$$f_{sdsn1}^{u} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|_{\mathbf{F}}^2 + \alpha\|\mathbf{U}\|_{\mathbf{F}}^2. \tag{8}$$

Clearly, by using the ridge regression, $\mathbf{U}$ in (8) has a closed-form solution, which is same to (2). Then, we have two gradient descent algorithms to train $\mathbf{W}$. First, given a fixed $\mathbf{U}$, $\mathbf{W}$ can be solved by using a gradient descent algorithm [7], [32] to minimize the following square error

$$f_{sdsn1}^{1} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|_{\mathbf{F}}^2 + \beta\Psi(\mathbf{H}). \tag{9}$$

By deriving the gradient, we obtain

$$\frac{\partial f_{sdsn1}^{1}}{\partial\mathbf{W}} = 2\mathbf{X}\left[d\phi(\mathbf{H}^T)\circ(\mathbf{U}\mathbf{U}^T\mathbf{H} - \mathbf{U}\mathbf{T})^T\right] + 2\beta\mathbf{X}\left[d\phi(\mathbf{H}^T)\circ\mathbf{H}^T\circ/\widetilde{\mathbf{H}}^T\right], \tag{10}$$

where the element-wise multiplication and division are respectively denoted as $\circ$ and $\circ/$, the element-wise gradient computation is denoted by $d\phi(\mathbf{H}^T)$, the gradient of $\phi(a)$ is denoted as $d\phi(a)$, $\widetilde{\mathbf{H}} = [\widetilde{\mathbf{H}}_{\Gamma_1,:}\cdots\widetilde{\mathbf{H}}_{\Gamma_g,:}\cdots\widetilde{\mathbf{H}}_{\Gamma_G,:}]$, $\widetilde{\mathbf{H}}_{\Gamma_g,:} = [\widetilde{h}_{j,i}]$ $(j \in \Gamma_g, 1 \le i \le N)$, and $\widetilde{h}_{j,i} = \sqrt{\sum_{a\in\Gamma_g}h_{a,i}^2}$. We mainly consider the sigmoid activation function and the ReLU activation function. To better analyze the non-differentiable ReLU activation function, we also consider its smooth function, soft-plus or $soft(a) = log(1 + e^a)$ (more analysis is provided in supplementary material (I)). Then, $d\phi(a)$ is described as

$$d\phi(a) = \begin{cases} \phi(a)\times(1-\phi(a)), & \text{if } \phi(a) \text{ is the sigmoid;} \\ \dfrac{e^a}{1+e^a}, & \text{if } \phi(a) \text{ is the soft-plus;} \\ \begin{cases} 1, & a > 0; \\ 0, & a \le 0. \end{cases}, & \text{if } \phi(a) \text{ is the ReLU.} \end{cases} \tag{11}$$

Note that in this paper, we select a subgradient 0 for ReLU because there exist a subgradient set $\{0, 1\}$ although ReLU is non-differentiable at zero.

Algorithm 1 Training Algorithm of GSNM-1 (or GSNM-2)

1: **Input:** Parameters $\theta = \{\epsilon, \varepsilon, \alpha, \beta, G\}$, data matrix $\mathbf{X}$, label matrix $\mathbf{T}$, and training epochs E.
2: **Initialize:** $index$ = 0, weights $\mathbf{W}$ with small random initialization.
3: **While** $index < E$ and f_{sdsn1} (orf_{sdsn2}) $> \varepsilon$ **do**
4: Update $\mathbf{H}$ by Eq. (4) given a fixed $\mathbf{W}$,;
5: Update $\mathbf{W}$ by Eq. (14) (or Eq. (21));
6: $index$ ++;
7: **End While**
8: **Return** Weight matrix $\mathbf{W}$.

Second, the deterministic relationship between $\mathbf{U}$ and $\mathbf{W}$ is used to calculate the gradient of $\mathbf{W}$ for faster moving $\mathbf{W}$ towards the optimal points. By plugging (2) into criterion (5), the square error is rewritten as

$$f^2_{sdsn1} = \|[(\mathbf{H}\mathbf{H}^T + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{T}^T]^T\mathbf{H} - \mathbf{T}\|^2_{\mathbf{F}} + \alpha\|(\mathbf{H}\mathbf{H}^T + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{T}^T\|^2_{\mathbf{F}} + \beta\Psi(\mathbf{H}). \quad (12)$$

Unfortunately, the gradient of f^2_{sdsn1} tends to be complicated when the regularization of $\mathbf{U}$ is used in the function (5) (i.e. $\alpha > 0$). To simply compute the gradient, we assume $\alpha = 0$[3] in f^2_{sdsn1}. Similar to [7], we then derive the gradient $\frac{\partial f^2_{sdsn1}}{\partial \mathbf{W}}$ to yield

$$\frac{\partial f^2_{sdsn1}}{\partial \mathbf{W}} = 2\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ [\mathbf{H}^\dagger(\mathbf{H}\mathbf{T}^T)(\mathbf{T}\mathbf{H}^\dagger) - \mathbf{T}^T(\mathbf{T}\mathbf{H}^\dagger)]\Big] + 2\beta\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ \mathbf{H}^T \circ /\widetilde{\mathbf{H}}^T\Big], \quad (13)$$

where $\mathbf{H}^\dagger = \mathbf{H}^T(\mathbf{H}\mathbf{H}^T)^{-1}$, $d\phi(\cdot)$, and $\widetilde{\mathbf{H}}$ are defined in (10).

By using the gradient defined in (10) and (13), the algorithm then updates $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{W} - \epsilon\frac{\partial f^1_{sdsn1}}{\partial \mathbf{W}} \quad or \quad \mathbf{W} = \mathbf{W} - \epsilon\frac{\partial f^2_{sdsn1}}{\partial \mathbf{W}}, \quad (14)$$

with a learning rate ϵ. The complete learning process of the weight matrix is summarized in **Algorithm 1**.

B. GSNM-2: Split Hidden Representations of Samples in Different Classes Into Different Groups

In the above subsection IV-A, we use l_1/l_2 regularization to learn the statistical dependencies between hidden units. However, it is difficult to increase diversity between-class in GSNM-1 for classification tasks. In this subsection, we use l_2 regularization to prevent over-fitting, and utilize l_1 regularization to obtain diversity between-class. Thus, the l_1/l_2 regularization can be applied to another group sparse representation to enforce the image representations in each class into a group [2], [20], [59]. The formulation of group sparse representation is as follows.

Let a set of all samples be denoted by $\mathfrak{H} = \{1, 2, \cdots, N\}$. GSNM-2 divides the set $\mathfrak{H}$ into C groups, where C is the number of categories, $\mathfrak{H} = \bigcup_{c=1}^{C}\Lambda_c$, $\bigcap_{c=1}^{C}\Lambda_c = \varnothing$, and Λ_c is the c-th group. Thus, $\mathbf{H} = [\mathbf{H}_{:,\Lambda_1}; \cdots; \mathbf{H}_{:,\Lambda_c}; \cdots; \mathbf{H}_{:,\Lambda_C}]$, where $\mathbf{H}_{:,\Lambda_c} = [h_{j,i}]$ $(1 \le j \le L, i \in \Lambda_c)$ are the hidden representations corresponding to all samples of c class. In this paper, we select the same number of the training samples in every class. For example, given a dataset with 5 classes, we select 20 training samples for each class. We then have $N = 5 \times 20 = 100$, $\mathfrak{H} = \bigcup_{c=1}^{5}\Lambda_c = \{1, 2, \cdots, 20\} \bigcup \{21, 22, \cdots, 40\} \bigcup \{41, 42, \cdots, 60\} \bigcup \{61, 62, \cdots, 80\} \bigcup \{81, 82, \cdots, 100\}$, and $\mathbf{H} = [\mathbf{H}_{:,\Lambda_1}; \mathbf{H}_{:,\Lambda_2}; \cdots; \mathbf{H}_{:,\Lambda_5}]$.

Following the group sparse representation [2], [59], we consider another l_1/l_2-norm as follows

$$\Omega(\mathbf{H}) = \sum_{c=1}^{C}\|\mathbf{H}_{:,\Lambda_c}\|_{\mathbf{F}}, \quad (15)$$

where $\mathbf{H}_{:,\Lambda_c} = [h_{j,i}]$ $(1 \le j \le L, i \in \Lambda_c)$ are the hidden representations corresponding to all samples of c class.

By replacing the $\Psi(\mathbf{H})$ with $\Omega(\mathbf{H})$, the square error (5) can be rewritten as

$$f_{sdsn2} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|^2_{\mathbf{F}} + \alpha\|\mathbf{U}\|^2_{\mathbf{F}} + \gamma\,\Omega(\mathbf{H}), \quad (16)$$

where γ is a tuning parameter denoting the weights of the group sparse regularization term $\Omega(\mathbf{H})$.

1) Learning Weights- Algorithm 2: Similar to GSNM-1, the weight matrices $\mathbf{U}$ and $\mathbf{W}$ can also be trained by using the gradient descent algorithm [7] to minimize the square error (16). Given a fixed $\mathbf{W}$, $\mathbf{H}$ is uniquely determined, and $\mathbf{U}$ has the closed-form solution in Eq. (2).

Given fixed current $\mathbf{U}$, learning $\mathbf{W}$ can be preformed by minimizing the following two square errors

$$f^1_{sdsn2} = \|\mathbf{U}^T\mathbf{H} - \mathbf{T}\|^2_{\mathbf{F}} + \gamma\,\Omega(\mathbf{H}), \quad (17)$$

and

$$f^2_{sdsn2} = \|[(\mathbf{H}\mathbf{H}^T + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{T}^T]^T\mathbf{H} - \mathbf{T}\|^2_{\mathbf{F}} + \alpha\|(\mathbf{H}\mathbf{H}^T + \alpha\mathbf{I})^{-1}\mathbf{H}\mathbf{T}^T\|^2_{\mathbf{F}} + \gamma\,\Omega(\mathbf{H}). \quad (18)$$

Similar to (10) and (13), the gradients of the square errors (17) and (18) are calculated as follows

$$\frac{\partial f^1_{sdsn2}}{\partial \mathbf{W}} = 2\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ (\mathbf{U}\mathbf{U}^T\mathbf{H} - \mathbf{U}\mathbf{T})^T\Big] + 2\gamma\,\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ \mathbf{H}^T \circ /\overline{\mathbf{H}}^T\Big], \quad (19)$$

and

$$\frac{\partial f^2_{sdsn2}}{\partial \mathbf{W}} = 2\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ [\mathbf{H}^\dagger(\mathbf{H}\mathbf{T}^T)(\mathbf{T}\mathbf{H}^\dagger) - \mathbf{T}^T(\mathbf{T}\mathbf{H}^\dagger)]\Big] + 2\gamma\,\mathbf{X}\Big[d\phi(\mathbf{H}^T) \circ \mathbf{H}^T \circ /\overline{\mathbf{H}}^T\Big], \quad (20)$$

where $\circ$, $\circ/$ and $d\phi(\mathbf{H}^T)$ are defined in (10), $\mathbf{H}^\dagger$ is defined in (13), the gradient of $\phi(a)$ is denoted as $d\phi(a)$ defined in (11), $\overline{\mathbf{H}} = [\overline{\mathbf{H}}_{:,\Lambda_1}; \cdots; \overline{\mathbf{H}}_{:,\Lambda_c}; \cdots; \overline{\mathbf{H}}_{:,\Lambda_C}]$, $\overline{\mathbf{H}}_{:,\Lambda_c} = [\overline{h}_{j,i}]$ $(1 \le j \le L, i \in \Lambda_c)$, and $\overline{h}_{j,i} = \sqrt{\sum_{a\in\Lambda_c} h^2_{j,a}}$.

By using the gradient defined in (19) (or (20)), the algorithm then updates $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{W} - \epsilon\frac{\partial f^1_{sdsn2}}{\partial \mathbf{W}} \quad or \quad \mathbf{W} = \mathbf{W} - \epsilon\frac{\partial f^2_{sdsn2}}{\partial \mathbf{W}}, \quad (21)$$

with the learning rate ϵ. The complete learning process of the weight matrix is summarized in **Algorithm 1**.

[3]Assuming $\alpha > 0$. It would become more complicated due to lack of cancellation of a number of terms [7]. Unfortunately, $\alpha = 0$ will lead to a bad result as there is no regularization issue to prevent $\mathbf{U}$ over-fitting.

Algorithm 2 Stacking Algorithm of Our GS-DSN Model

Input: Label matrix $\mathbf{T}$, data matrix $\mathbf{X}$, parameters $\theta = \{\epsilon, \varepsilon, \alpha, \beta, G\}$, the number of layers K and training epochs E (in this paper E and K are set to 5 and 3, respectively).
Initialize: $k = 1$ and $\mathbf{X}^1 = \mathbf{X}$.
for $k \leq K$
  Train the weight matrix $\mathbf{W}^k$ of k-th sparse module by using the **Algorithm 1** given $\mathbf{X}^k$, $\mathbf{T}$, θ and E;
  Given $\mathbf{X}^k$ and $\mathbf{W}^k$, calculate $\mathbf{H}^k$ by Eq. (4), $\mathbf{U}^k$ by Eq. (2) and $\mathbf{Y}^k = \left(\mathbf{U}^k\right)^T \mathbf{H}^k$;
  $\mathbf{X}^{k+1} = [\mathbf{X}^k; \mathbf{Y}^k]$;
  k ++;
end for
return Weight Matrices $\mathbf{W}^k (k = 1, \cdots, K)$.

C. The GS-DSN Architecture

Two kinds of K-layer GS-DSN are constructed by respectively stacking the GSNM-1 and GSNM-2 modules, which are described in the above two subsections. In the k-th sparse module, the input, hidden representations, output, and weight matrices are denoted as $\mathbf{X}^k$, $\mathbf{H}^k$, $\mathbf{Y}^k$, $\mathbf{W}^k$ and $\mathbf{U}^k$, respectively. Given a data matrix $\mathbf{X}$ and a label matrix $\mathbf{T}$, k and $\mathbf{X}^1$ are set to 1 and $\mathbf{X}$, respectively. Then the generally stacking procedure of GS-DSN can be described as the following three phases:

1) Learn the weight matrices $\mathbf{W}^k$ and $\mathbf{U}^k$ in k-th module by using the **Algorithm 2** (GSNM-1 or GSNM-2) to minimize the squares error between $\mathbf{Y}^k$ and $\mathbf{T}$.
2) Construct the input data $\mathbf{X}^{k+1}$ in the $k+1$-th module by cascading the output $\mathbf{Y}^k$ of k-th sparse module and the input $\mathbf{X}^k$ of the k-th module.
3) Iterate 1) and 2) to build our GS-DSN model.

The optimization of GS-DSN is summarized in **Algorithm 2**. Two kinds of GS-DSN with three stacking modules are illustrated in Fig. 1 **(2)** and **(3)**. To capture the group sparse representation information in the data, this paper proposes GS-DSN, which is performed by penalizing the hidden unit activations. GS-DSN still keeps the parallelism and scalability advantages of DSN during training all weight parameters because of its simple structure.

V. Experiments

In this section, we firstly do experiments to confirm the fast inference of our proposed models. We secondly perform group sparseness comparison among DSN, GS-DSN(sigm), GS-DSN(soft) and GS-DSN(relu). We finally demonstrate the excellent performance of GS-DSN.

A. Datasets

Four popular image datasets (i.e., Extended YaleB, AR, 15-Scene and Caltech101) are described as follows.

- Extended YaleB dataset consists of 2,414 frontal face images with 38 people. It approximately has 64 images in each person. We crop and normalize these images to 192×168 pixels.
- AR dataset contains over 4,000 color images with 126 people. There are 26 face images taken during two sessions for each person. There are more facial variations (e.g., illuminations, and sunglasses) in these images. We sample a subset from the dataset, which contains 2,600 images from 50 male people and 50 female people, and crop and normalize them to 165×120 pixels.
- 15-Scene dataset consists of 4485 scene images with 15 categories (e.g., office, kitchen, living room, bedroom, store, industrial). These images are about 250×300 resolution, with 200 to 400 images per category.
- Caltech-101 dataset consists of 9144 images from 102 classes (e.g., cars, pagodas, dollars). The samples from each class have significant shape variabilities. The size of each image is roughly 300×200 pixels, with 31 to 800 images per class.

All four datasets are processed[4] using the approach proposed in [25]. With a random matrix created by using a zero-mean normal distribution, we project every image onto a feature vector with n-dimension in the Extended YaleB and AR face datasets. Correspondingly, the dimensions of their random features, which are normalized to $[-1, 1]$, are 504 and 540, respectively. By using a four-level structure and a 200 SIFT-descriptors codebook, the spatial pyramid features are calculated from the SIFT descriptors, which are extracted from patches in the 15-Scene dataset [25]. Moreover, we reduce these features to 500 dimensions by using PCA. For the Caltech101 dataset, we use the pre-trained VGG model[5] to extract the CNN features with 4096 dimensions, and they are finally reduced to 2000 dimensions by PCA. Note that our result on Caltech101 is 89.0% accuracy, which is less than 92.3% reported in [44] as the VGG model provided in MatConvNet[6] is worse than [44].

We randomly initialize the weight parameters by using a normal distribution with zero-mean and standard deviation of 0.01. For simplicity, the learning rate ϵ and the parameter α are selected from $\{2, 1, 0.2, 0.15, 0.1, 0.05, 0.02, 0.01, 0.001\}$ and $\{5, 1, 0.5, 0.3, 0.2, 0.1\}$, respectively. In GSNM-1, the parameter β is selected from $\{0.1, 0.01, 0.001, 0.0001, 0.0005, 0.00001\}$, while in GSNM-2 the parameter γ is selected from $\{0.1, 0.01, 0.001, 0.005, 0.0001, 0.00001\}$. Both GSNM-1 and GSNM-2 choose the group number G from $\{1, 5, 10, 20, 50\}$. The selected parameters of all datasets are provided in supplementary material (II). In all experiments, the training epochs E, the number of hidden units, and the number of layers K are set to 5, 500 and 3, respectively. The left equations in (14) and (21) are selected as the update rules in Algorithm 1. For each data set, we run the experiments 10 times, with randomly selected training, validation, and testing data, and report their average recognition results. The validation data are used to tune the best hyper-parameters for classification results. Some notation is shown in Table I.

[4]The pre-processed data that is available for download http://www.umiacs.umd.edu/ zhuolin/projectlcksvd.html
[5]http://www.vlfeat.org/matconvnet/pretrained/
[6]http://www.vlfeat.org/matconvnet/

TABLE I
NOTATION

DSN: DSN with sigmoid GS-DSN(sigm): GS-DSN with sigmoid GS-DSN(soft): GS-DSN with soft-plus GS-DSN(relu): GS-DSN with ReLU		
first layer	second layer	third layer
DSN-1	DSN-2	DSN-3
GS-DSN(sigm)-1	GS-DSN(sigm)-2	GS-DSN(sigm)-3
GS-DSN(soft)-1	GS-DSN(soft)-2	GS-DSN(soft)-3
GS-DSN(relu)-1	GS-DSN(relu)-2	GS-DSN(relu)-3

TABLE II
COMPUTATION TIME FOR A TEST IMAGE ON EXTENDED YALEB DATASET

Methods	SRC [52]	LC-KSVD1 [25]	LC-KSVD2 [25]
Average (ms)	20.121	0.531	0.502
Methods	DSN [7]	S-DSN [31]	GS-DSN(sigm)
Average (ms)	0.078	0.070	0.081
Methods	GS-DSN(soft)	GS-DSN(relu)	
Average (ms)	0.089	0.069	

B. Fast Inference

GS-DSN is always faster than LC-KSVD and SRC, since it only computes a projection multiplication and a non-linear transformation. Based on the discriminative dictionaries, LC-KSVD use the orthogonal matching pursuit (OMP) algorithm [48] to compute sparse codes. Although OMP can obtain a fast inference, it still needs many iterative computations. SRC also solves the l_1-minimization problem, which has an expensive inference. We only select a repetition experiment on the Extended YaleB dataset in subsection V-D to compute the time as it is just to verify the fast inference. At testing time, we compare with SRC, LC-KSVD1, and LC-KSVD2 for inferring a single test sample. As showed in Table II, our GS-DSN(relu) is 7 times faster than LC-KSVD2, and is close to DSN and S-DSN as they have the similar inference method.

C. Group Sparseness Comparisons

Compared to DSN, we present the group sparseness analysis of GS-DSN with sigm, soft and relu in this subsection. The effects of a l_1/l_2 regularization can be interpreted on two levels: a between-group and a within-group level. On the within-group level, the l_2 norm will equally penalize the activation of all hidden units in the same group for preventing the over-fitting. On the across-group level, the proposed models use l_1 norm to compete with each other for learning group sparse representations. In this paper, GS-DSN is implemented by adding a l_1/l_2 regularization to DSN. The group effects of GS-DSN are not as obvious as the ones described by group sparse coding (GSC) [2]. The reason is that GSC uses the different dictionary words to represent the images of the different class and easily obtain group coefficients, while GS-DSN randomly initializes the weights and uses the l_1/l_2 regularization to reinforce the group effects. In addition, we also show some differences between DSN and GS-DSN in supplementary material (III).

Fig. 2. Extended YaleB dataset.

Similar to the sparseness analysis (sparse group RBMs) [36], **Hoyer's sparseness measure** (HSM) [19] is employed to measure the sparsity of the representations, which are trained by GS-DSN(sigm), GS-DSN(soft), GS-DSN(relu) and DSN. Given a vector $\mathbf{y}$ with d dimension, its HSM is calculated by:

$$HSM(\mathbf{y}) = \frac{\sqrt{d} - (\sum_{i=1}^{d} |\mathbf{y}_i|)/\sqrt{\sum_{i=1}^{d} \mathbf{y}_i^2}}{\sqrt{d} - 1}. \quad (22)$$

This measure is calculated in a normalized interval $[0, 1]$. If the vector $\mathbf{y}$ contains more zero elements, then $HSM(\mathbf{y})$ is closer to 1 and vice versa. Table III shows the sparseness comparisons on the Extended YaleB dataset. We observe that GS-DSN(sigm) and GS-DSN(relu) have higher HSM and higher recognition results. When GSNM-1 is used to do experiments, Table III shows that the average HSM of the three layers in GS-DSN(sigm) and GS-DSN(relu) are about 0.118 and 0.263, respectively. In contrast, the average sparseness of the three layers in DSN and GS-DSN(soft) is on average below 0.036. It can be seen that GS-DSN(sigm) and GS-DSN(relu) can learn much sparser representations. Although we add the sparse penalties in GS-DSN(soft), HSM is still low because the soft-plus activation function enforces the negative of output of hidden units activations to approximate to zeros from the positive part. soft-plus is not similar to ReLU, which rectifies the negative activations. If we enforce the sparse penalties, the classification result is very poor. In the same way, when GSNM-2 is used to do experiments, there is a similar observation that GS-DSN(sigm) and GS-DSN(relu) have higher HSM and recognition results.

D. Results

1) Face Recognition: Extended YaleB is a challenging dataset because of varying conditions (e.g., illuminations and expressions), which are shown in Fig. 2. For each class, we randomly choose 29 images for training, 3 images for validation and the rest for testing. The validation data are used to select the parameters. Compared to the extended YaleB dataset, the AR dataset has more facial variations (i.e., illuminations, expressions, and facial "disguises"), which are shown in Fig. 3. For each person, we randomly select 18 images for training, 2 images for validation and the other 6 for testing.

We compare GS-DSN with many state-of-the-art methods, such as, DSN [7], locality-constrained linear coding (LLC) [50], label consistent K-SVD (LC-KSVD) [25],

TABLE III

HSM ON EXTENDED YALEB. WE CHOOSE 15 TRAINING IMAGES PER CLASS, AND THE REST FOR TESTING. IN THE GSNM-1 MODULE THE HIDDEN UNITS ARE DIVIDED INTO 5 NON-OVERLAPPING SETS, WHILE THE GROUP SIZE FOR GS-DSN ARE 38 IN THE GSNM-2 MODULE SINCE THE REPRESENTATIONS OF EACH CLASS ARE ARRANGED INTO A GROUP, AND EXTENDED YALEB HAS 38 PEOPLE

	layer	DSN		GS-DSN(sigm)		GS-DSN(soft)		GS-DSN(relu)	
GSNM-1		HSM	Accuracy (%)	HSM	Accuracy (%)	HSM	Accuracy (%)	HSM	Accuracy (%)
G=5	1	0.035	86.8	0.106	86.9	0.004	86.7	0.314	89.7
	2	0.039	88.1	0.119	89.3	0.004	88.9	0.237	89.9
	3	0.035	89.1	0.128	90.6	0.004	89.5	0.237	91.2
GSNM-2	1	0.035	86.8	0.102	86.7	0.004	86.5	0.297	89.5
G=38	2	0.039	88.1	0.103	88.9	0.003	88.6	0.217	89.7
	3	0.035	89.1	0.113	90.2	0.002	89.3	0.225	91.0

TABLE IV

RECOGNITION ACCURACIES ON THE EXTENDED YALEB DATASET BY USING RANDOM FACE FEATURES

	Methods	Accuracy (%)	Methods	Accuracy (%)	Methods	Accuracy (%)
	SRC [52]	97.2 ± 0.5	LC-KSVD [25]	96.7 ± 0.2	LLC [53]	90.7 ± 0.3
	DSN-1 [7]	96.6 ± 0.5	DSN-2 [7]	96.9 ± 0.6	DSN-3 [7]	97.4 ± 0.4
	S-DSN-1 [31]	96.9 ± 0.4	S-DSN-2 [31]	97.6 ± 0.3	S-DSN-3 [31]	97.6 ± 0.5
GSNM-1	GS-DSN(sigm)-1	97.6 ± 0.5	GS-DSN(sigm)-2	97.9 ± 0.5	GS-DSN(sigm)-3	$\mathbf{98.1 \pm 0.4}$
	GS-DSN(soft)-1	96.1 ± 0.6	GS-DSN(soft)-2	96.2 ± 0.4	GS-DSN(soft)-3	96.2 ± 0.3
	GS-DSN(relu)-1	96.7 ± 0.7	GS-DSN(relu)-2	96.8 ± 0.5	GS-DSN(relu)-3	96.8 ± 0.4
GSNM-2	GS-DSN(sigm)-1	97.3 ± 0.4	GS-DSN(sigm)-2	97.7 ± 0.5	GS-DSN(sigm)-3	97.7 ± 0.5
	GS-DSN(soft)-1	95.8 ± 0.5	GS-DSN(soft)-2	96.0 ± 0.6	GS-DSN(soft)-3	96.1 ± 0.4
	GS-DSN(relu)-1	96.7 ± 0.3	GS-DSN(relu)-2	96.7 ± 0.4	GS-DSN(relu)-3	96.8 ± 0.3

TABLE V

RECOGNITION ACCURACIES ON THE AR FACE DATASET BY USING RANDOM FACE FEATURES

	Methods	Accuracy (%)	Methods	Accuracy (%)	Methods	Accuracy (%)
	SRC [52]	97.5 ± 0.4	LC-KSVD [25]	97.8 ± 0.3	LLC [53]	88.7 ± 0.5
	DSN-1 [7]	97.6 ± 0.5	DSN-2 [7]	97.9 ± 0.5	DSN-3 [7]	98.1 ± 0.6
	S-DSN-1 [31]	97.9 ± 0.3	S-DSN-2 [31]	98.1 ± 0.6	S-DSN-3 [31]	98.2 ± 0.5
GSNM-1	GS-DSN(sigm)-1	98.3 ± 0.4	GS-DSN(sigm)-2	98.7 ± 0.5	GS-DSN(sigm)-3	$\mathbf{98.8 \pm 0.5}$
	GS-DSN(soft)-1	96.9 ± 0.1	GS-DSN(soft)-2	97.3 ± 0.1	GS-DSN(soft)-3	97.5 ± 0.2
	GS-DSN(relu)-1	97.5 ± 0.3	GS-DSN(relu)-2	97.7 ± 0.3	GS-DSN(relu)-3	97.7 ± 0.2
GSNM-2	GS-DSN(sigm)-1	98.3 ± 0.3	GS-DSN(sigm)-2	98.5 ± 0.2	GS-DSN(sigm)-3	98.6 ± 0.4
	GS-DSN(soft)-1	97.2 ± 0.2	GS-DSN(soft)-2	97.5 ± 0.1	GS-DSN(soft)-3	97.6 ± 0.2
	GS-DSN(relu)-1	97.2 ± 0.4	GS-DSN(relu)-2	97.6 ± 0.3	GS-DSN(relu)-3	97.7 ± 0.4

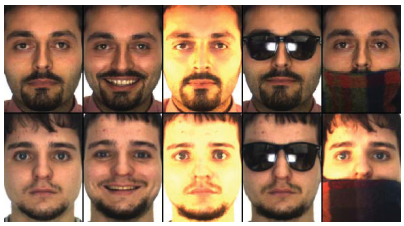


Fig. 3. AR dataset.

and sparse representation-based classifier (SRC) [52]. Table IV and Table V summarize the recognition accuracies on Extended YaleB and AR, respectively. We observe that our GS-DSN(sigm) has better recognition accuracies than DSN, LLC, LC-KSVD, and SRC. Table IV shows that GS-DSN(sigm)-1 and GS-DSN(sigm)-3 stacked by GSNM-1 have better recognition accuracies than LC-KSVD, and achieve about 0.9% and 1.4% improvement on Extended YaleB. Moreover, Table V also shows that GS-DSN(sigm)-1 and GS-DSN(sigm)-3 stacked by GSNM-1 also have better recognition accuracies than LC-KSVD and achieve about 0.5% and 1.0% improvement on AR. Compared to our conference results [31], we have 0.5% and 0.6% improvement in Extended YaleB and AR, respectively.

The learning rate ϵ is chosen to balance the magnitude of input data. When the value of input data is big, ϵ is small, and vice versa. α, which is the regularization parameter of $\mathbf{U}$, is chosen in our given set $\{5, 1, 0.5, 0.3, 0.2, 0.1\}$. So, we mainly consider the parameters G and β, which are used to determine the classification accuracy and the sparseness of GSNM-1. Classification accuracies with different G and β are shown in Fig. 4. Given $\epsilon = 0.1$ and $\alpha = 0.2$ we can get a good performance when $G = 10$ and $\beta = 0.001$.

2) 15-Scene: The 15-Scene dataset shown in Fig. 5 includes living room, bedroom, kitchen, highway, mountain etc. For

TABLE VI
RECOGNITION ACCURACIES ON THE 15-SCENE DATASET BY USING THE SPATIAL PYRAMID FEATURES

	Methods	Accuracy (%)	Methods	Accuracy (%)	Methods	Accuracy (%)
	ITDL [39]	81.1	SR-LSR [34]	85.7	DPD [45]	86.0
	RSP [23]	88.1	LLC [50]	89.2	BMDDL [58]	96.9
	ISPR+IFV [35]	91.1	SRC [52]	96.2	LC-KSVD [25]	97.0
	DBDL [1]	98.7	GSSC [46]	82.6	DeepSC [17]	83.8
	Hybrid-CNN [57]	91.9	DeCAF [10]	88.0	DSFL+DeCAF [60]	92.8
	DSN-1 [7]	96.3 ± 0.5	DSN-2 [7]	97.1 ± 0.5	DSN-3 [7]	97.4 ± 0.5
	S-DSN-1 [31]	98.6 ± 0.3	GS-DSN(relu)-2	98.7 ± 0.3	GS-DSN(relu)-3	98.8 ± 0.3
GSNM-1	GS-DSN(sigm)-1	96.7 ± 0.4	GS-DSN(soft)-2	97.4 ± 0.2	GS-DSN(soft)-3	97.7 ± 0.2
	GS-DSN(soft)-1	98.0 ± 0.2	GS-DSN(soft)-2	98.3 ± 0.2	GS-DSN(soft)-3	98.4 ± 0.2
	GS-DSN(relu)-1	98.3 ± 0.2	GS-DSN(relu)-2	98.8 ± 0.2	GS-DSN(relu)-3	98.9 ± 0.2
GSNM-2	GS-DSN(sigm)-1	97.2 ± 0.4	GS-DSN(soft)-2	97.8 ± 0.3	GS-DSN(soft)-3	98.0 ± 0.3
	GS-DSN(soft)-1	98.0 ± 0.3	GS-DSN(soft)-2	98.2 ± 0.3	GS-DSN(soft)-3	98.5 ± 0.3
	GS-DSN(relu)-1	98.5 ± 0.2	GS-DSN(relu)-2	98.9 ± 0.2	GS-DSN(relu)-3	$\mathbf{99.1 \pm 0.2}$

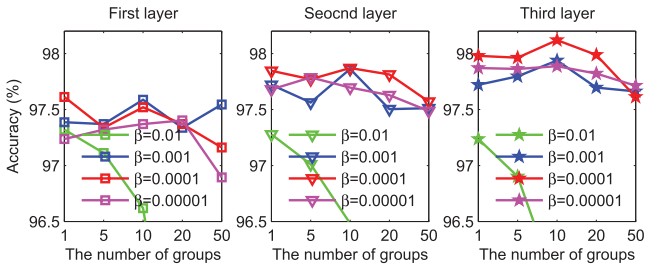


Fig. 4. Classification accuracies with different G and β on the Extended YaleB dataset. In this experiment, GSNM-1 with sigmoid activation function is used to build GS-DSN(sigm).

Fig. 5. 15 categories in the 15-Scene dataset.

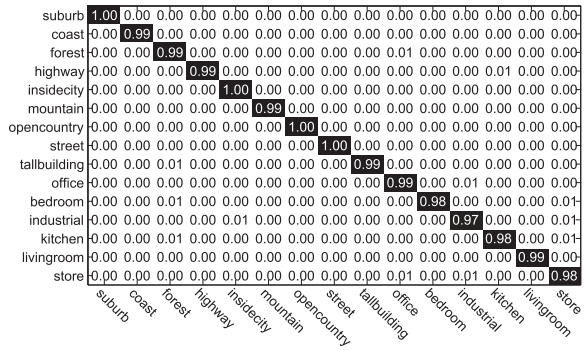


Fig. 6. Confusion matrix for 15-Scene category dataset using GSNM-1.

each category, we randomly choose 90 images for training data, 10 images for validation data, and the rest for test data. The validation data are used to select the parameters.

We compare our results with SRC [52], LC-KSVD [25], deep sparse coding (DeepSC) [17], DSN [7], and other state-of-the-art approaches: spatial pyramid matching using Laplacian sparse coding (LScSPM) [13], LLC [50], information-theoretic dictionary learning (ITDL) [39], discriminative Bayesian dictionary learning (DBDL) [1], important spatial pooling region + improved Fisher vector (ISPR+IFV) [35], spatial regularized latent semantic representation (SR-LSR) [34], deep convolutional activation feature (DeCAF) [10], group structured sparse coding (GSSC) [46], DSFL+DeCAF [60], Hybrid-CNN [57], discriminative part detector (DPD) [45], and randomized spatial partition (RSP) [23]. The detailed comparison results are shown in Table VI. We successfully learned the sparse representation by directly training the discriminative dictionaries. Table VI shows that GS-DSN(relu)-3 has a better performance than DBDL with a 0.4% improvement, and reaches about 1.7% improvement over the deep networks (e.g., DeepSC, Hybrid-CNN, DSN, DeCAF and DSFL+DeCAF). Compared to our conference results [31], we have 0.3% improvement. The confusion matrices for GS-DSN(relu) stacked by GSNM-1 and GSNM-2 are shown in Fig. 6 and Fig. 7, respectively. We can observe that the accuracies of bedroom, industrial, kitchen and store are lower than others since they are similar.

As far as we know, compared to the existing methods GS-DSN(relu) gets the best accuracy, even though Hybrid-CNN [57] combines the training set of a new scene-centric dataset and the training set of ImageNet to train the deep convolutional neural network. There are two reasons. First, spatial pyramid feature is possibly better than Hybrid-CNN in 15-Scene dataset because it is developed specifically for recognizing natural scene categories (more details are provided in [25] and [28]). Second, our model, GS-DSN(relu), has more discriminative power than other sparse models, such as SRC [52], LC-KSVD [25], DBDL [1], BMDDL [58].

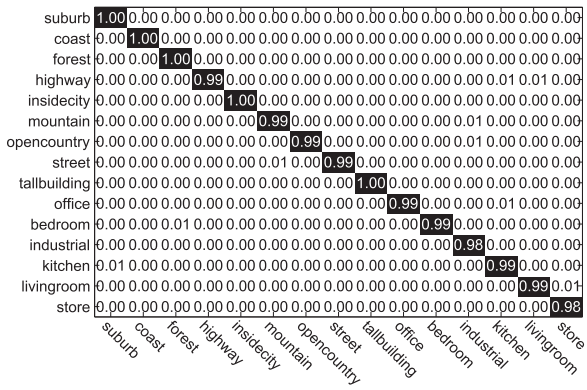


Fig. 7. Confusion matrix for 15-Scene category dataset using GSNM-2.

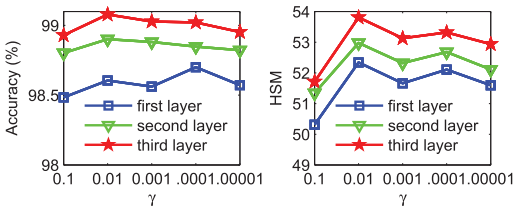


Fig. 8. Classification accuracies with different γ on the 15-Scene dataset. In this experiment, GSNM-2 with ReLU activation function is used to build the GS-DSN(relu).

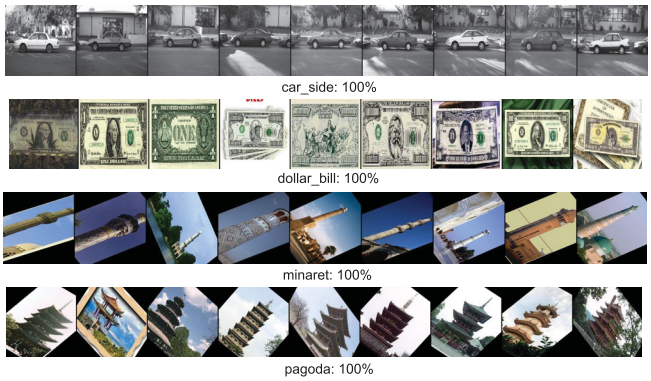


Fig. 9. Images from categories with 100% accuracy in the Caltech101 dataset. In fact, there are 23 categories with 100% accuracy.

In addition, we consider the parameter γ, which is used to determine the classification accuracy and the sparseness of GSNM-2. Classification accuracies with different γ are shown in Fig. 8. Given $\epsilon = 1$ and $\alpha = 0.1$ we can observe that there is little effect on performance with γ, and the best classification accuracy is achieved at $\gamma = 0.01$.

3) Caltech101: For each category, we randomly select 9, 18 and 27 images for training data, 1, 2 and 3 images for validation data, and the rest for test data, respectively. The validation data are used to select the parameters. By using the CNN features, We compare our approach with SRC [52], LC-KSVD [25], DeepSC [17], DSN [7] and other approaches LScSPM [13], LLC [50], low-rank sparse coding (LRSC) [55], DBDL [1], learning structured low-rank representations (LSLR) [56], and locality-constrained low-rank coding (LCLR) [24]. Table VII report the average classification

TABLE VII

RECOGNITION ACCURACIES ON THE CALTECH101 DATASET BY USING THE CNN FEATURES

Methods	10	20	30
Unsupervised Methods			
ScSPM [13]	-	-	85.1 ± 0.7
SRC [52]	73.2 ± 0.9	77.5 ± 0.9	81.8 ± 0.8
LLC [50]	73.7 ± 0.4	78.3 ± 0.5	82.5 ± 0.4
LRSC [55]	74.5 ± 0.2	78.7 ± 0.3	82.8 ± 0.2
LCLR [24]	75.1 ± 0.3	79.6 ± 0.5	84.3 ± 0.5
GSSC [46]	-	-	86.2 ± 0.2
Supervised Methods			
LSLR [56]	-	-	82.1 ± 0.5
LC-KSVD [25]	77.5 ± 0.1	83.2 ± 0.1	87.6 ± 0.1
DBDL [1]	78.8 ± 0.2	83.6 ± 0.3	88.0 ± 0.3
DSN-1 [7]	78.3 ± 0.5	80.3 ± 0.8	81.2 ± 1.1
DSN-2 [7]	79.2 ± 0.6	81.0 ± 0.5	81.9 ± 1.0
DSN-3 [7]	79.5 ± 0.7	81.3 ± 0.5	82.3 ± 1.0
S-DSN-1 [31]	78.9 ± 0.5	82.4 ± 0.5	85.3 ± 0.5
S-DSN-2 [31]	79.5 ± 0.4	83.5 ± 0.3	86.6 ± 0.6
S-DSN-3 [31]	79.9 ± 0.5	83.9 ± 0.4	87.5 ± 0.5
GSNM-1			
GS-DSN(sigm)-1	78.5 ± 1.0	80.6 ± 0.9	81.5 ± 0.9
GS-DSN(sigm)-2	79.6 ± 1.1	81.2 ± 1.0	82.2 ± 0.9
GS-DSN(sigm)-3	79.9 ± 1.1	81.8 ± 0.9	82.7 ± 0.8
GS-DSN(soft)-1	77.5 ± 0.6	82.1 ± 0.7	83.3 ± 0.5
GS-DSN(soft)-2	77.7 ± 0.4	81.2 ± 0.6	83.4 ± 0.5
GS-DSN(soft)-3	77.7 ± 0.6	81.2 ± 0.7	83.4 ± 0.7
GS-DSN(relu)-1	80.1 ± 0.7	83.2 ± 0.7	87.3 ± 0.3
GS-DSN(relu)-2	80.6 ± 0.7	83.8 ± 0.6	88.5 ± 0.4
GS-DSN(relu)-3	$\mathbf{80.9 \pm 0.6}$	$\mathbf{84.4 \pm 0.7}$	$\mathbf{89.0 \pm 0.3}$
GSNM-2			
GS-DSN(sigm)-1	78.5 ± 1.1	80.5 ± 1.2	81.3 ± 0.9
GS-DSN(sigm)-2	79.5 ± 0.6	81.2 ± 1.2	82.1 ± 0.9
GS-DSN(sigm)-3	79.8 ± 0.7	81.7 ± 1.0	82.3 ± 0.8
GS-DSN(soft)-1	76.7 ± 0.6	80.2 ± 0.6	82.5 ± 0.6
GS-DSN(soft)-2	76.8 ± 0.6	80.2 ± 0.7	82.5 ± 0.6
GS-DSN(soft)-3	76.8 ± 0.6	80.2 ± 0.7	82.5 ± 0.6
GS-DSN(relu)-1	76.3 ± 0.9	81.8 ± 0.5	87.2 ± 0.6
GS-DSN(relu)-2	77.0 ± 0.8	82.0 ± 0.5	88.3 ± 0.5
GS-DSN(relu)-3	77.3 ± 1.0	82.2 ± 0.5	88.8 ± 0.5

accuracies. We see that our GS-DSN(relu)-3 exceeds 1.0% compared to the competing dictionary learning methods (e.g., DBDL, LC-KSVD, LRSC, SRC and GSSC. GS-DSN(relu)), and also gets about 6.7% improvement over DSN. Compared to our conference results [31], we have 1.5% improvement. Moreover, Fig. 9 shows some images from four classes which achieve high classification accuracy when training 30 images per class.

We estimate the performances of our GS-DSN with different numbers of hidden units. We randomly select 27 images for training, 3 images for validation, and the rest for test data. We consider five numbers (i.e., 100, 500, 1,000, 2,000, 3,000), and report the recognition accuracies in Fig. 10. We see that our GS-DSN maintains high recognition accuracies, and is better than DSN. Clearly, we also observe that the accuracies can be improved when increasing the number of hidden units.

4) Effects From the Number of Layers: The deep model can abstract a better representation of images by utilizing multiple-layers architecture. Based on the Tables IV, V, VI and VII to check the effects, we see that the classification accuracy can be improved with increasing the number of layers. Moreover, compared with the sparse coding and dictionary learning methods, the advantage of GS-DSN is that it can quickly infer

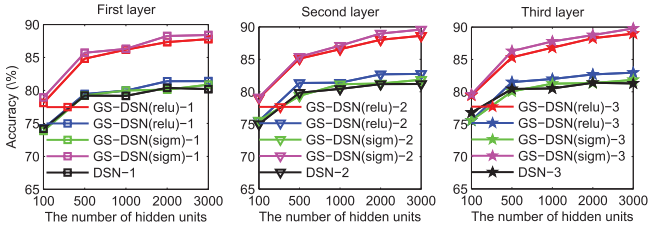


Fig. 10. Recognition accuracy with different number of hidden units used in GS-DSN(sigm), GS-DSN(soft), GS-DSN(relu), S-DSN [31] and DSN [9] on Caltech101. We choose the number of hidden units ranging from 100 to 3000. The red and green curves plot the accuracy of GS-DSN stacked GSNM-1 with 5 groups, while the cyan and black curves also plot the accuracy of S-DSN and DSN, respectively.

the sparse representation by a deep network. When the number of layers increases to four or five, the accuracy is slightly improved because there is a suitable number of layers for a given dataset.

VI. Conclusion

For image classification tasks, we proposed two kinds of group sparse deep stacking network model, GS-DSN, which was created by stacking GSNM-1 (or GSNM-2) modules. GSNM-1 exhibited local dependencies between hidden units and GSNM-2 gathered the image representations in each class into a group. In each module, the lower-layer weights were estimated using gradient descent with upper-layer weights, which had the closed-form solution. We used the GS-DSN to further extract features from the random facial features for facial recognition, and the spatial pyramid features for object recognition. Experimental results showed that GS-DSN outperformed the relevant classification methods on four public datasets.

References

[1] N. Akhtar, F. Shafait, and A. Mian, "Discriminative Bayesian dictionary learning for classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 12, pp. 2374–2388, Dec. 2016.

[2] S. Bengio, F. Pereira, Y. Singer, and D. Strelow, "Group sparse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 82–89, Vancouver, BC, Canada, 2009.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[4] X. Chen, Q. Lin, S. Kim, J. G. Carbonell, and E. Xing, "Smoothing proximal gradient method for general structured sparse regression," *Ann. Appl. Stat.*, vol. 6, no. 2, pp. 719–752, Jun. 2012.

[5] L. Deng, X. He, and J. Ga, "Deep stacking networks for information retrieval," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Vancouver, BC, Canada, May 2013, pp. 3153–3157.

[6] L. Deng and D. Yu, "Accelerated parallelizable neural network learning algorithm for speech recognition," in *Proc. Interspeech*, 2011, pp. 2281–2284.

[7] L. Deng and D. Yu, "Deep convex networks: A scalable architecture for speech pattern classification," in *Proc. Interspeech*, 2011, pp. 2285–2288.

[8] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2013.

[9] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. IEEE Conf. Acoust., Speech, Signal Process.*, Mar. 2012, pp. 2133–2136.

[10] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. 31th Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 647–655.

[11] D. Donoho, "For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution," *Commun. Pure Appl. Math.*, vol. 59, no. 6, pp. 797–829, Jun. 2006.

[12] D. Erhan, Y. Bengio, P. Manzagol, A. Courville, and P. Vincent, "Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, vol. 11, no. 2, pp. 625–660, Feb. 2010.

[13] S. Gao, I. Tsang, L.-T. Chia, and P. Zhao, "Local features are not lonely-laplacian sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 3555–3561.

[14] P. Garrigues and B. Olshausen, "Learning horizontal connections in a sparse coding model of natural images," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 505–512.

[15] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, Sardinia, Italy, 2010, pp. 249–256.

[16] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[17] Y. He, K. Kavukcuogluy, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *Proc. SIAM Int. Conf. Data Mining*, 2014, pp. 902–910.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, Dec. 2004.

[20] J. Huang, F. Nie, H. Huang, and C. Ding, "Supervised and projected sparse coding for image classification," in *Proc. AAAI Conf. Artif. Intell.*, Washington, DC, USA, 2013, pp. 438–444.

[21] B. Hutchinson, L. Deng, and D. Yu, "Tensor deep stacking networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1944–1957, Aug. 2013.

[22] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, "Proximal methods for sparse hierarchical dictionary learning," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 487–494.

[23] Y. Jiang, J. Yuan, and G. Yu, "Randomized spatial partition for scene recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 730–743.

[24] Z. Jiang, P. Guo, and L. Peng, "Locality-constrained low-rank coding for image classification," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2780–2786.

[25] Z. Jiang, Z. Lin, and L. S. Davis, "Label consistent K-SVD: Learning a discriminative dictionary for recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2651–2664, Nov. 2013.

[26] S. Kim and E. P. Xing, "Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eQTL mapping," *Ann. Appl. Stat.*, vol. 6, no. 3, pp. 1095–1117, Sep. 2012.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, , Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[28] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, Jun. 2006, pp. 2169–2178.

[29] 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, Nov. 1998.

[30] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 873–880.

[31] J. Li, H. Chang, and J. Yang, "Sparse deep stacking network for image classification," in *Proc. AAAI Conf. Artif. Intell.*, Austin, TX, USA, 2015, pp. 3804–3810.

[32] J. Li, Y. Kong, H. Zhao, J. Yang, and Y. Fu, "Learning fast low-rank projection for image classification," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4803–4814, Oct. 2016.

[33] J. Li, T. Zhang, W. Luo, J. Yang, X. Yuan, and J. Zhang, "Sparseness analysis in the pertraining of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1425–1438, Jun. 2017.

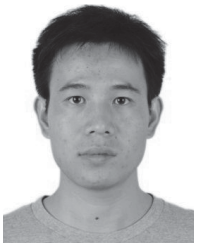
[34] X. Li and Y. Guo, "Latent semantic representation learning for scene classification," in *Proc. Int. Conf. Learn. Represent.*, Banff, Canada, 2014, pp. 532–540.

[35] D. Lin, C. Lu, R. Liao, and J. Jia, "Learning important spatial pooling regions for scene classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Columbus, OH, USA, Jun. 2014, pp. 3722–3729.

[36] H. Luo, R. Shen, C. Niu, and C. Ullrich, "Sparse group restricted Boltzmann machines," in *Proc. AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2011, pp. 429–434.
[37] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional sparse autoencoders for image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2017.2712793.
[38] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 807–814.
[39] Q. Qiu, V. M. Patel, and R. Chellappa, "Information-theoretic dictionary learning for image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2173–2184, Nov. 2014.
[40] J. Rolfe and Y. LeCun, "Discriminative recurrent sparse auto-encoders," in *Proc. Int. Conf. Learn. Represent.*, 2013, p. 1.
[41] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2009, pp. 448–455.
[42] S. Shoham, D. O'Connor, and R. Segev, "How silent is the brain: Is there a 'dark matter' problem in neuroscience? *J. Comput. Physiol. A*, vol. 192, no. 8, pp. 777–784, Aug. 2006.
[43] G. Sigletos, G. Paliouras, C. Spyropoulos, and M. Hatzopoulos, "Combining information extraction systems using voting and stacked generalization," *J. Mach. Learn. Res.*, vol. 6, no. 11, pp. 1751–1782, Nov. 2005.
[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, p. 1.
[45] J. Sun and J. Ponce, "Learning discriminative part detectors for image classification and cosegmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3400–3407.
[46] A. Szlam, K. Gregor, and Y. LeCun, "Fast approximations to structured sparse coding and applications to object classification," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 200–213.
[47] M. Thom and G. Palm, "Sparse activity and sparse connectivity in supervised learning," *J. Mach. Learn. Res.*, vol. 14, no. 4, pp. 1091–1143, Apr. 2013.
[48] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.
[49] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[50] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3360–3367.
[51] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[52] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.
[53] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 1794–1801.
[54] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 543–550.
[55] T. Zhang, B. Ghanem, S. Liu, C. Xu, and N. Ahuja, "Low-rank sparse coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sydney, NSW, Australia, Dec. 2013, pp. 281–288.
[56] Y. Zhang, Z. Jiang, and L. Davis, "Learning structured low-rank representations for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 676–683.
[57] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 487–495.
[58] P. Zhou, C. Zhang, and Z. C. Lin, "Bilevel model-based discriminative dictionary learning for recognition," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1173–1187, Mar. 2017.
[59] Y. Zhuang, Y. Wang, F. Wu, Y. Zhang, and W. Lu, "Supervised coupled dictionary learning with group structures for multi-modal retrieval," in *Proc. AAAI Conf. Artif. Intell.*, Washington, DC, USA, 2013, pp. 1070–1076.
[60] Z. Zuo, G. Wang, B. Shuai, L. Zhao, Q. Yang, and X. Jiang, "Learning discriminative and shareable features for scene classification," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 552–568.

Jun Li (M'16) received the B.A. degree in applied mathematics from Pan Zhi Hua University in 2006, the M.S. degree in computer application from China West Normal University in 2009, and the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology in 2015. From 2012 to 2013, he was a Visiting Student with the Department of Statistics, Rutgers University, Piscataway, NJ, USA.

He is currently a Post-Doctoral Research Associate with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. He has published over 30 papers in AAAI, IJCAI, ACM MM, IEEE TNNLS, IEEE TIP, and other venues. His current research interests include deep learning, sparse representations, subspace clustering, and recurrent neural networks. He has served as a PC member for AAAI 2017, 2018, IJCAI 2017, IEEE FG 2017, 2018, and IEEE ICMLA 2016, 2017, and a reviewer for over 10 international journals, such as the IEEE TNNLS and the IEEE TIP.



Heyou Chang received the B.S. degree in computer science and technology from the Henan University of Science and Technology, Luoyang, China, in 2011. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His current research interests include computer vision, machine learning, and dictionary learning.

Jian Yang (M'08) received the B.S. degree in mathematics from Xuzhou Normal University in 1995, the M.S. degree in applied mathematics from Changsha Railway University in 1998, and the Ph.D. degree with a focus on the subject of pattern recognition and intelligence systems from the Nanjing University of Science and Technology (NUST), in 2002. In 2003, he was a Post-Doctoral Researcher with the University of Zaragoza. From 2004 to 2006, he was a Post-Doctoral Fellow with the Biometrics Centre, The Hong Kong Polytechnic University. From 2006 to 2007, he was a Post-Doctoral Fellow with the Department of Computer Science, New Jersey Institute of Technology. He is currently a Professor with the School of Computer Science and Technology, NUST. He has authored over 80 scientific papers in pattern recognition and computer vision. His journal papers have been cited over 3000 times in the ISI Web of Science and 7000 times in the Web of Scholar Google. His research interests include pattern recognition, computer vision, and machine learning. He is currently an Associate Editor of the *Pattern Recognition Letters* and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Wei Luo received the B.S. degree in electronics information engineering from the Hunan Institute of Science and Technology in 2007, the M.S. degree in pattern recognition and intelligence systems from the Henan University of Technology in 2010, and the Ph.D. degree in pattern recognition and intelligence systems from the Nanjing University of Science and Technology in 2015. He is currently an Assistant Professor with the College of Mathematics and Informatics, South China Agriculture University. His research interests include computer vision and machine learning.

Yun Fu (S'07–M'08–SM'11) received the B.Eng. degree in information engineering and the M.Eng. degree in pattern recognition and intelligence systems from Xian Jiaotong University, China, and the M.S. degree in statistics and the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign. He has been an Interdisciplinary Faculty Member affiliated with the College of Engineering and the College of Computer and Information Science, Northeastern University, since 2012. He has extensive publications in leading journals, books/book chapters, and international conferences/workshops. His research interests are machine learning, computational intelligence, big data mining, computer vision, pattern recognition, and cyber-physical systems. He is a fellow of IAPR, a Lifetime Senior Member of ACM and SPIE, a Lifetime Member of AAAI, OSA, and Institute of Mathematical Statistics, a member of the ACM Future of Computing Academy, the Global Young Academy, and INNS, and the Beckman Graduate Fellow from 2007 to 2008. He received seven prestigious young investigator awards from NAE, ONR, ARO, IEEE, INNS, UIUC, and Grainger Foundation; seven best paper awards from the IEEE, IAPR, SPIE, and SIAM; and three major industrial research awards from Google, Samsung, and Adobe. He serves as an associate editor, the chair, a PC member, and a reviewer for many top journals and international conferences/workshops. He is currently an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEANING SYSTEMS.