Features Combined From Hundreds of Midlayers: Hierarchical Networks With Subnetwork Nodes

Yimin Yang, *Member, IEEE*, and Q. M. Jonathan Wu, *Senior Member, IEEE*

***Abstract*— In this paper, we believe that the mixed selectivity of neuron in the top layer encodes distributed information produced from other neurons to offer a significant computational advantage over recognition accuracy. Thus, this paper proposes a hierarchical network framework that the learning behaviors of features combined from hundreds of midlayers. First, a subnetwork neuron, which itself could be constructed by other nodes, is functional as a subspace features extractor. The top layer of a hierarchical network needs subspace features produced by the subnetwork neurons to get rid of factors that are not relevant, but at the same time, to recast the subspace features into a mapping space so that the hierarchical network can be processed to generate more reliable cognition. Second, this paper shows that with noniterative learning strategy, the proposed method has a wider and shallower structure, providing a significant role in generalization performance improvements. Hence, compared with other state-of-the-art methods, multiple channel features with the proposed method could provide a comparable or even better performance, which dramatically boosts the learning speed. Our experimental results show that our platform can provide a much better generalization performance than 55 other state-of-the-art methods.**

***Index Terms*— Feature extraction, feature representation, generalization performance, image recognition, neural networks (NNs).**

I. Introduction

THE past few years have witnessed the bloom of deep learning (DL) including autoencoders, convolutional neural networks (CNNs), and so on. [1]–[7]. In fact, DL has been around for many years dating back to the works in 1980s [1]–[3], [8]–[10]. Over many benchmark data sets, recent DL methods, including GoogLeNet [11], AlexNet [12], very deep convolutional network [13], deep residual network [14], DenseNet [15], recurrent deep network [16], fuzzy deep network [17], and so on [18], [19], have substantially advanced the state-of-the-art accuracies of objection recognition and have turned out to be very good at discovering intricate structures in high-dimensional data.

However, because the iterative method of learning has become a paradigm for training hierarchical NNs, it is clear that the learning effectiveness and learning speed of these networks are far slower than required, which has been a major bottleneck for many applications. Furthermore, the iterative methods such as back propagation used in DL suffer from slow convergence, getting trapped in a local minimum and being sensitive to the learning rate setting. Noniterative learning methods had penetrated into networks for a long time such as the random forest [20], the random vector functional link network [21]–[23], Quick-Net [24], extreme learning machine (ELM) [25]–[27], and others [28], but utilizing a noniterative learning strategy in the training process of the hierarchical network has been rarely explored.

Furthermore, recent evidence reveals that network depth is of crucial importance, and the learning results on data sets all utilized a large number of "graduation" layer models, with a depth of 16 [12], 39 [13], to 160 [14], [16]. However, unrestrainedly increasing network depth brings an incredible computational cost with a limited performance boost. For example, on large-scale data set ImageNet, only 1.6% top-1 accuracy boost occurred by switching the 264-layer DenseNet [15] from the 169-layer DenseNet, but increase nearly two times the computational time.

Third, initial CNN models such as AlexNet and VGG -16 do implement multiple fully connected layers as a classifier after convolutional layers, from our point of view, hoping to improve generalization performance further. However, recent CNN models such as DenseNet and ResNet forfeit the multiple fully connected layers in their network architecture only including one fully connected layer to obtain final predict outputs. Such movement reflects the multifaced reasons such as computation cost and parameter numbers, but the most critical idea is that adding extra traditional multiple fully connected layers embedded in the CNN models are unable to improve any generalization performance. Therefore, our question is if traditional multiple fully connected layers were unable to provide performance gains, how about hierarchical networks with hundreds of fully connected layers?

Last but not least, with network depth increasing, deep learners utilizes extensive computational power and can take a quite long time to train, making it difficult to validate, repeat, and improve their results widely. For example, even with GPUs, training a medium image categorization task such as CIFAR10 with ImageNet pretrained VGG16 takes around 15 h. It motivates us to design a hierarchical network that could provide a comparable performance with pretrained deep feature directly rather than optimized features generated by the end-to-end training.

In this paper, we address the above-mentioned motivations by introducing a deep three-general layer learning framework. In particular, this paper contributes the following.

Manuscript received March 13, 2018; revised August 26, 2018 and December 23, 2018; accepted December 24, 2018. This work was supported by the Natural Sciences and Engineering Research Council of Canada. *(Corresponding authors: Yimin Yang; Q. M. Jonathan Wu.)*

Y. Yang is with the Computer Science Department, Lakehead University, Thunder Bay, ON P7B 5E1, Canada (e-mail: yyang48@lakeheadu.ca).

Q. M. J. Wu is with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P4, Canada (e-mail: jwu@uwindsor.ca).

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/TNNLS.2018.2890787

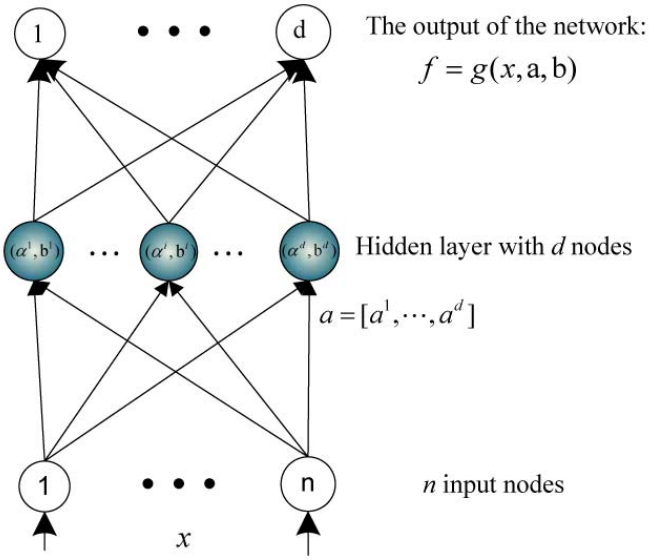


Fig. 1. Structure of single-layer feedforward network.

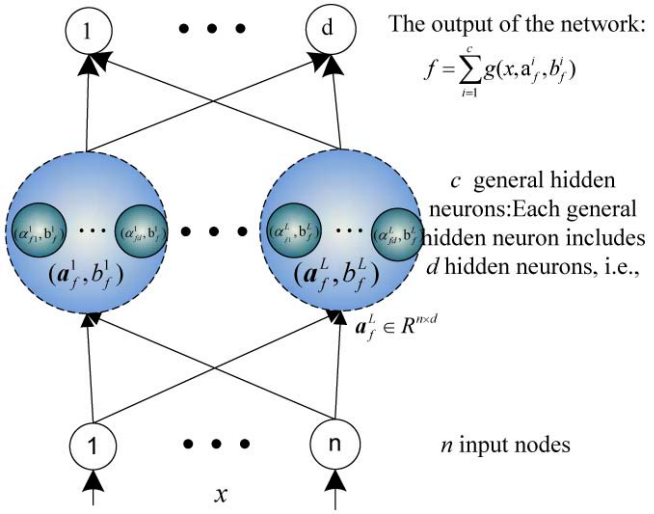


Fig. 2. Single-layer feedforward network with subnetwork neurons.

1) Hundreds of two-layer networks with subnetwork nodes are embedded into the first general layer. A subnetwork node, which itself could be constructed by other nodes, can be used like a local feature extractor to generate subspace features from the input data independently. Optimal features extracted and combined from the local extractors encourage the inputs from the same class to have resemble codes and those from different classes to have dissimilar codes. Thus, optimal features can be obtained by combining heterogeneous features which extracted from a two-general layer (a depth of two), rather than a depth of hundreds layers, which can dramatically reduce the computational workloads.
2) Superb generalization performance. Our method is evaluated with other 55 state-of-the-art methods and provides a much better generalization performance than the 55 rival methods hundred times faster speed. For instance, the learning platform achieves categorization accuracy of 98.2% on the Scene15 data set in 20 s, which is even higher than human-level performance. For medium data sets, our method without CNN training even beats CNN networks with fine-tuning regarding recognition accuracy and training speed. On large-scale data set Places365, experimental results show that with deep features, 16-VGG net provides 44.2% accuracy and requires more than 7 days, while our method provides 45.3% accuracy.

II. Proposed Method

A. Motivations of the Proposed Method

Mathematically, the output of the single-layer network (see Fig. 1) f with d neurons can be represented as

$$f = g(\mathbf{x}, \mathbf{a}, b)$$
$$\mathbf{a} = [\mathbf{a}^1, \ldots, \mathbf{a}^d] \in \mathbf{R}^{n \times d}, \quad b \in \mathbf{R} \tag{1}$$

where $\mathbf{x} \in \mathbf{R}^{M \times n}$ represent the input data which includes M samples with n dimensions, $\mathbf{a}$, b represent input weights and biases of the hidden layer, $\mathbf{a}^1 \in \mathbf{R}^n$ represents the input weight of a neuron, and $g(\cdot)$ denotes the activation function of the hidden layers. Furthermore, for multilayer network structure, the output of each hidden layer can be represented as

$$\mathbf{H}_f^i = g\left(\mathbf{H}_f^{i-1} \cdot [\mathbf{a}^1, \ldots, \mathbf{a}^d]^T + b\right) \tag{2}$$

where $\mathbf{H}_f^i$ is the output of the ith layer and $\mathbf{H}_f^{i-1}$ is the output of the $(i-1)$th layer.

Unlike the multilayer network structure mentioned above, in this paper, we propose a new multilayer network architecture with subnetwork nodes (see Fig. 2). Compared to (1) and (2), the output of the single-layer network with c subnetwork nodes can be represented as

$$f^c = \sum_{i=1}^{c} g\left(\mathbf{x}, \mathbf{a}_f^i, b_f^i\right), \quad \mathbf{a}^i \in \mathbf{R}^{d \times n} \tag{3}$$

where subnetwork node $\mathbf{a}$ represents the input weights of a subnetwork neuron. The structural differences between a single-layer feedforward NN (SLFN) and an SLFN with subnetwork neurons are depicted in Figs. 1 and 2.

The proposed architecture is indicated in Fig. 3. In general, there are two differences between the proposed structure and other networks.

1) Following the idea that a neuron itself can be constituted by neurons [18], [29], [30], in this paper, the subnetwork node is designed as a feature extractor without any activation function inside. Furthermore, different from traditional network connection rule, which indicates that all the nodes should be connected entirely, in our method, the subnetwork nodes are not fully connected with each other
2) In the proposed method, multiple data channels [see Fig. 3 (right)] are used in the first general layer to "extend heterogeneous features." By doing so, the deepest/combined features actually generated by hundreds

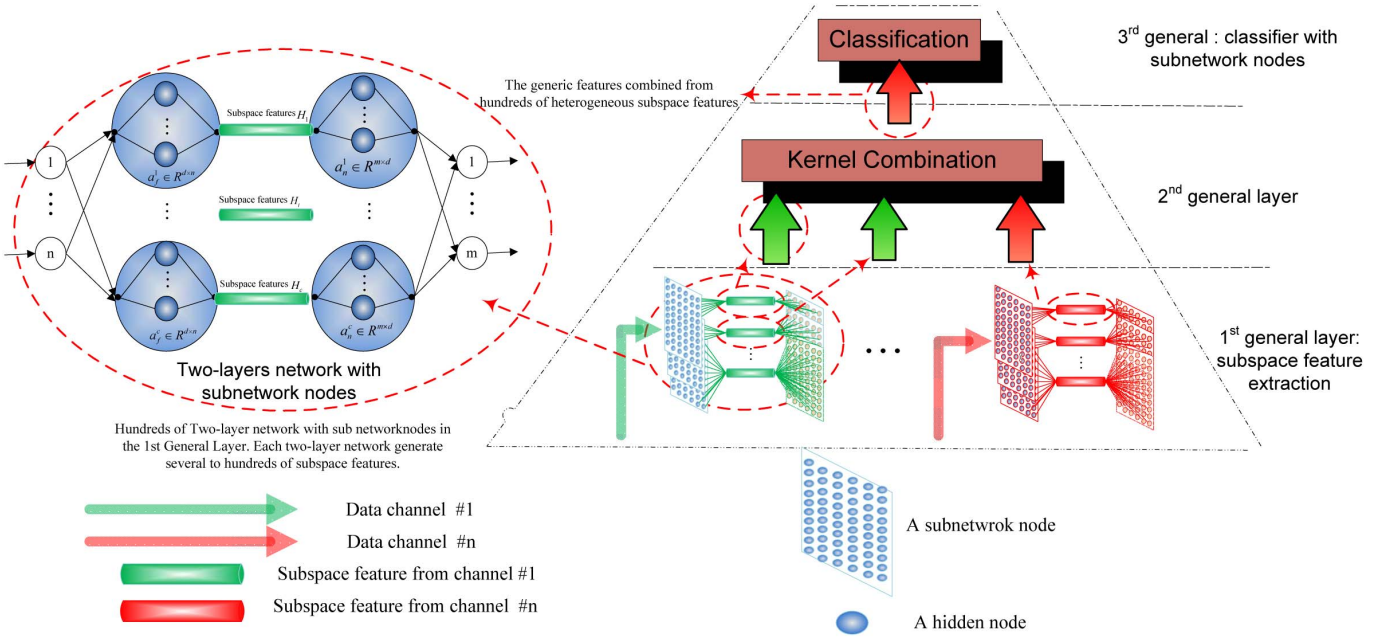


Fig. 3. Proposed method with multiple data channel structures.

of subnetwork nodes, which bring a significant "heterogeneous characters." Compared with the recent hierarchical NN models, the proposed model has a wider and more shallower structure, providing a much faster learning speed with a comparable performance.

B. First General Layer: Subspace Feature Extraction

Fig. 3 (left) shows how to obtain subspace features.

Step 1: Given M training samples $\{(\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{M}, \mathbf{x}_k \in \mathbf{R}^n$ from a continuous system, and set subnetwork node index equals to 0 ($c = 0$).

Step 2: Generate a new subnetwork node randomly as

$$\begin{aligned} c &= c + 1 \\ \mathbf{H}_f^c &= \mathbf{a}_f^c, \quad \cdot\mathbf{x} + b_f^c \end{aligned} \tag{4}$$

where $\mathbf{a}_f \in \mathbf{R}^{d\times n}, b_f \in \mathbf{R}$ is the random subnetwork neuron. $\mathbf{H}_f^c$ is the cth subspace features.

Step 3: For desired targets $\mathbf{y}$, the subnetwork nodes in the second layer $(\mathbf{a}_h^c, b_h^c)$ [see Fig. 3 (left)] are calculated by

$$\begin{aligned} \mathbf{a}_h^c &= \mathbf{y} \cdot \left({\mathbf{H}_f^c}^T \left(\frac{C}{\mathbf{I}} + \mathbf{H}_f^c {\mathbf{H}_f^c}^T \right)^{-1} \right), \quad \mathbf{a}_h \in \mathbf{R}^{d\times m} \\ b_h^c &= \sqrt{\text{mse}(\mathbf{a}_h \cdot \mathbf{H}_f^c - (\mathbf{y}))}, \quad b_n \in \mathbf{R} \end{aligned} \tag{5}$$

where C is a positive value.

Step 4: Calculate the residual error $\mathbf{e}$ as

$$\mathbf{e} = \mathbf{y} - \left(\mathbf{a}_h^c \cdot \mathbf{H}_f^c + b_h^c\right). \tag{6}$$

Step 5: Calculate the error feedback data as

$$\mathbf{P} = \mathbf{e} \cdot \left(\left(\mathbf{a}_h^c\right)^T \left(\frac{C}{\mathbf{I}} + \mathbf{a}_h^c \left(\mathbf{a}_h^c\right)^T \right)^{-1} \right). \tag{7}$$

Step 6: Update the normalized error feedback data as

$$\mathbf{P} = u\left(\mathbf{P} + \mathbf{H}_f^c\right) \tag{8}$$

where u is a normalized function $u(\cdot)$: $\rightarrow [-1, 1]$.

Step 7: Update parameter of the *subnetwork node* $\mathbf{a}_f^c, b_f^c$ in the first layer by

$$\begin{aligned} \mathbf{a}_{\text{temp}} &= \mathbf{P} \cdot \left(\mathbf{x}^T \left(\frac{C}{\mathbf{I}} + \mathbf{x}\mathbf{x}^T \right)^{-1} \right) \\ \mathbf{a}_f^c &= \mathbf{a}_f^c + \lambda \cdot \left(\mathbf{a}_{\text{temp}} - \mathbf{a}_f^c\right) \\ b_f^c &= \sqrt{\text{mse}\left(\mathbf{a}_f^c \cdot \mathbf{x} - \mathbf{P}\right)}, \quad b_f^c \in \mathbf{R} \end{aligned} \tag{9}$$

where λ represents the learning rate to overcome the overfitting problem.

Step 8: Calculate the cth subspace features

$$\mathbf{H}_f^c = \mathbf{a}_f^c \cdot \mathbf{x} + b_f^c. \tag{10}$$

Step 9: Generate $L-1$ subnetwork nodes by steps 1–8, and obtain the L subspace features $\{\mathbf{H}_f^1, \ldots, \mathbf{H}_f^c \ldots \mathbf{H}_f^L\}$.

C. Second General Layer: Subspace Features Combination

In the proposed method, combination operator is used for subspace feature combination. In particular, the combined features $\mathbf{H}$ can be

$$\mathbf{H}^{i\oplus j} = K\left(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma\right) \tag{11}$$

where $K(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma), i \neq j$ is a combination operator. Given several subspace features $\mathbf{H}_f^1, \ldots, \mathbf{H}_f^c$, the expression is

$$\begin{aligned} \mathbf{H}^{1\oplus 2} &= K\left(\mathbf{H}_f^1, \mathbf{H}_f^2, \gamma\right) \\ \mathbf{H}^{1\oplus 2\oplus 3} &= K\left(K\left(\mathbf{H}_f^1, \mathbf{H}_f^2, \gamma\right), \mathbf{H}_f^3, r\right) \\ &\vdots \\ \mathbf{H}^{1\oplus 2\oplus \ldots \oplus c} &= K\left(\ldots\left(K\left(\mathbf{H}_f^1, \mathbf{H}_f^2, \gamma\right), \mathbf{H}_f^3, r\right)\ldots\right). \end{aligned} \tag{12}$$

For example, such combination operator can be but are not limited to

1) *Plus operator*

$$K\left(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma\right) = \mathbf{H}_f^i + \mathbf{H}_f^j. \tag{13}$$

2) *Linear kernel*

$$K\left(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma\right) = \mathbf{H}_f^i \cdot \left(\mathbf{H}_f^j\right)^T. \tag{14}$$

3) *Radial basis function kernel*

$$K\left(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma\right) = \exp\left(-\gamma \left\|\mathbf{H}_f^i - \mathbf{H}_f^j\right\|^2\right). \tag{15}$$

4) *Polynomial kernel*

$$K\left(\mathbf{H}_f^i, \mathbf{H}_f^j, \gamma\right) = \left(\mathbf{H}_f^i \cdot \left(\mathbf{H}_f^j\right)^T\right)^\gamma. \tag{16}$$

5) *Concatenation operator*

$$K\left(\mathbf{H}_f^i, \mathbf{H}_f^j\right) = \left[\mathbf{H}_f^i, \mathbf{H}_f^j\right]. \tag{17}$$

D. Third General Layer: Classifier With Subnetwork Nodes

We use our previous work [31] as a final classifier. Given M distinct feature samples combined from combination operator $(\mathbf{H}, \mathbf{t})$, a normalized function $u(x)$: $\mathbf{R} \to (0, 1]$, a sigmoid or sine activation function g, and then for any continuous outputs $\mathbf{t}$, we have $\lim_{n\to+\infty} \|\mathbf{t} - ((g(\mathbf{a}_p^1, b_p^1, \mathbf{H})) \cdot \boldsymbol{\beta}_p^1 + \cdots + (g(\mathbf{a}_p^c, b_p^c, \mathbf{H})) \cdot \boldsymbol{\beta}_p^c)\| = 0$ holds with probability one if

$$\begin{aligned}
\mathbf{a}_p^c &= g^{-1}((\mathbf{e}_{c-1})) \cdot \mathbf{H}^T \left(\frac{C}{\mathbf{I}} + \mathbf{H}\mathbf{H}^T\right)^{-1} \\
b_p^c &= \text{sum}\left(\mathbf{a}_p^c \cdot \mathbf{H} - h^{-1}((\mathbf{e}_{c-1}))\right)/N, \quad b_p^c \in \mathbf{R} \\
g^{-1}(\cdot) &\begin{cases} \arcsin(\cdot) & \text{if } g(\cdot) = \sin(\cdot) \\ -\log\left(\dfrac{1}{(\cdot)} - 1\right) & \text{if } g(\cdot) = 1/(1 + e^{-(\cdot)}) \end{cases}
\end{aligned} \tag{18}$$

$$\begin{aligned}
\mathbf{e}_c &= \mathbf{t} - u_n^{-1} g\left(\mathbf{H}, \mathbf{a}_p^c, b_p^c\right) \\
\boldsymbol{\beta}_p^c &= \frac{\left\langle \mathbf{e}_{c-1}, u^{-1}\left(g\left(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c\right)\right)\right\rangle}{\left\|u^{-1}\left(g\left(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c\right)\right)\right\|^2}
\end{aligned} \tag{19}$$

where $[\cdot]^{-1}$ represents its inverse function. The detailed learning steps could be found in Algorithm 1.

Algorithm 1 Proposed Method

Given N features groups $(Q_1, \ldots, Q_N)$ generated from the same data set $Q_1 = \{(\mathbf{x}_k^1, \mathbf{y}_k^1)\}_{k=1}^M, \mathbf{x}_k^1 \in \mathbf{R}^{n_1}, \ldots, Q_N = \{(\mathbf{x}_k^N, \mathbf{y}_k^N)\}_{k=1}^M, \mathbf{x}_k^N \in \mathbf{R}^{n_N}$, positive coefficient C, and the number of subnetwork nodes L.

first general layer: Subspace feature extraction:

Set $n = 1$, and let $\{\mathbf{x}, \mathbf{y}\}$ equals Q_1.

while $n < N$ **do**

  while $c < L$ **do**

    Obtain a subspace feature $\mathbf{H}_f^{c+((n-1)\times L)}$ based on Section II-B Step 1–8.

  end while

  return Obtain L subspace features $\{\mathbf{H}_f^{1+((n-1)\times L)}, \ldots, \mathbf{H}_f^{L+((n-1)\times L)}\}$ from data group Q_n.

end while

return Obtain $L \times N$ subspace features $\{\mathbf{H}_f^1, \ldots, \mathbf{H}_f^{L\times N}\}$.

second general layer Feature combination:

Obtain combined features $\mathbf{H}$ as:

$$\mathbf{H} = \mathbf{H}^{1\oplus 2\oplus \ldots \oplus (N\times L)} \tag{20}$$

third general layer: Pattern learning: Given combined feature $\mathbf{H}$, set $c = 1, e_1 = \mathbf{t}$.

while $c < L$ **do**

  Step 1: Calculate the cth subnetwork node $(\mathbf{a}_p^c, b_p^c)$, and output weights $\boldsymbol{\beta}_p^c$ as:

$$\begin{aligned}
\mathbf{a}_p^c &= g^{-1}((\mathbf{e}_{c-1})) \cdot \mathbf{H}^T \left(\frac{C}{\mathbf{I}} + \mathbf{H}\mathbf{H}^T\right)^{-1}, \quad \mathbf{a}_p^c \in \mathbf{R}^{n\times m} \\
b_p^c &= \text{sum}\left(\mathbf{a}_p^c \cdot \mathbf{H} - h^{-1}((\mathbf{e}_{c-1}))\right)/N, \quad b_p^c \in \mathbf{R} \\
\boldsymbol{\beta}_p^c &= \frac{\left\langle \mathbf{e}_{c-1}, u^{-1}\left(g\left(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c\right)\right)\right\rangle}{\left\|u^{-1}\left(g\left(\mathbf{a}_p^c \cdot \mathbf{H} + b_p^c\right)\right)\right\|^2}
\end{aligned} \tag{21}$$

  Step 2: Calculate $\mathbf{e}_c = \mathbf{e}_{c-1} - \boldsymbol{\beta} \cdot g\left(\mathbf{a}_p^c, b_p^c, \boldsymbol{\beta}_p^c\right)$.

end while

III. Experimental Verification

A. Rival Methods and Experiment Environment Settings

In this section, we evaluate 55 the state-of-the-art methods arising from the following four families.

1) Sparse representation-based classification methods include sparse representation [49], sparse coding [36], sparse kernel singular value decomposition (KSVD) [50], sparse representation with combination features [51], laplacian sparse coding [37], locality constrained linear coding (LLC) [52], low-rank representation learning [53], and low-rank representation with spatial pyramid feature [54].
2) Recent feature coding methods include hard/soft assignment [40], [41], centrist [42], image similarity search [55], visual word ambiguity [39], feature fusion [38], [56], color fusion features [57], local pyramidal descriptors [58], hierarchical matching pursuit feature [59], natural image statistics [60], cosegmentation classifier [61], and discriminative color descriptors [62].
3) Hierarchical networks include multilayer deep network [45], max-pooling with spatial pyramid features [44], feature pooling [43], multiway local pooling [63], hybrid-CNN features [48], multilayer ELM [29], [64], CNN networks [65], hierarchical sparse coding with deep network [66], hierarchical manifold deep network [46], deconvolutional network [67], very deep CNN [68], ImageNet-pretrained CNNs [12], [13], [69], Places-pretrained AlexNet [47], Places-pretrained googleLeNet [47], recurrent convolutional network [16], Network in Networks [18], DenseNet [15], deep-supervised Nets [70], deep attention selective networks [71], and sumproduct network with deep architecture [72].
4) Single improved classifiers based on NN/support vector machine (SVM)/Kernel/Kernel NN (KNN):

TABLE I
SPECIFICATION OF IMAGE DATA SETS AND THE EXPERIMENTAL SETTINGS OF OUR METHOD

Datasets	Feature Type	#Image	# Training image per Category	# Category	# Parameters	# subnetwork neurons	neurons in #each subnetwork neuron	# channel	computation time
Scene15	Spatial pyramid	4,485	100	15	4.9 million	3	100	1	20 second
Caltech-101	HMP, SIFT, Spatial pyramid	9,144	15,30	102	13.4 million	3	100	3	6 min.
Caltech-101	HMP, SIFT, Spatial pyramid, AlexNet deep feature	9,144	15,30	102	15.3 million	5	100	1	8 min.
Caltech-256	SIFT, Spatial pyramid, AlexNet, ResNet, VGG16 deep feature	30,607	15,30	257	65.2 million	8	200	1	21 min.
CIFAR10	ImageNet pretrained deep feature	60,000	5000	10	6.6 ~ 0.7 million	4	400	1	3 ~ 6 minutes
Place365	ImageNet pretrained deep feature	365,000	800	365	-	3	200-800	1	32-36 minutes

nearest-neighbor NN [73], KNN-SVD [74], kernel codebook [32], multiple feature with SVM [75], [76], KNN with localized multiple kernel [34], kernel/linear spatial pyramid [35], discriminative-KSVD [77], object-to-class kernels [33], large margin learning [78], and linear SVM with local coordinate coding [79].

To verify the performance of the algorithms in our experiments, tested data sets are selected from a wide variety of available data sets, including ones with small, medium, large, and/or high dimensions. All databases are preprocessed in the same way (using the held-out method). Table I shows the training sample numbers and the category numbers of the corresponding data sets. Permutations of the whole data set are taken with random replacement, and training images (given in Table I) are used to create the training set. The remaining data set is used for the test set. For these image data sets, we perform all processing in grayscale even when color images are available. All experiments are repeated 10 times with different randomly selected training and test images, and the average of per-class recognition rates is recorded for each run. For all image data sets, several well-known feature descriptor methods, including CNN features, spatial pyramid features [35], [36], [40], hierarchical matching pursuit feature [59], scale-invariant feature transform, or raw image, are used for the following data sets: Caltech101/256, Oxford Flower 102, Scene15, CIFAR10, and Places365. Our experimental results for both Caltech101 and Caltech256 are generated by 15 or 30 images per category for training; for the Flower 102, Scene15, CIFAR10, and Places365 data sets, we use 15, 100, 5000, and 800 images per category for training. All the remaining images are used for testing. For our proposed method, parameter C is selected from $C \in \{2^{-15}, \ldots, 2^{15}\}$. The experimental results of the proposed method and competing methods are given in Tables II–VII. In these tables, the apparent best results are shown in boldface and the second best results are underlined.

For CNNs related features, we extract the features from penultimate layer of the particular network. CNNs have the quite similar network architectures for AlexNet, GoogleNet, DenseNet, and VGG. For AlexNet and VGG, we used the 4096-dimensional features from the fully connected layer (FC7) of the CNN. For GoogleNet, we used the 1024-dimensional features from the response of the global average pooling layer before softmax producing the class predictions. For DenseNet, we used the 448-dimensional (40-layer DenseNet) and 1168-dimensional (100-layer DenseNet) features from the global average pooling layer before the dense layer.

We compare our method with other state-of-the-art methods in three ways: 1) our method versus other classifiers; 2) our method versus deep transfer learning; and 3) our method versus CNN models with end-to-end training.

In detail, we define the following terms to separate experimental results related to DL models.

1) Deep Features, which mean that we extract features directly from a pretrained CNN model without any end-to-end training. In the experiments, we only use two pretrained CNN models including ImageNet pretrained CNN model and Places365 pretrained CNN model. Thus, the deep features only contain innate priors regarding the ImageNet or Places365 data set.
2) Deep Transfer Learning Features, which mean that we utilize a pretrained CNN model as an initial model for an end-to-end training task. In the experiments, we used the existing pretrained CNN networks (ImageNet or Places365 pretrained models) and associated the pretrained weights as initial parameters with end-to-end training on the target data set. Then, we extract the features from a penultimate layer of the particular network. In the case, the deep transfer learning features contain knowledge from both the ImageNet/Places365 data set and the target data set.
3) Deep Learning Features, which mean that we extract features from a trained CNN model after fine-tuning. In the experiments, we train a CNN network from scratch with random initialization, then extract the features from a penultimate layer of the particular network. In the case, the DL features only contain the priors from the target data set.

B. Scene Recognition

Scene15 data set contains 4486 gray-value images, of which 3860 images are from the 15-category scenes. Following the common experimental settings, we randomly select 100 images per category as training data and use the rest as test data. The Plus operator is used in the combination operator part. In each subnetwork node, the number of hidden nodes is set to 100.

In Table II, we include results from complex approaches that incorporate many cues and learning-optimal feature

TABLE II

SCENE-15 CLASSIFICATION ACCURACY FOR OUR METHOD AGAINST OTHER LEADING ALTERNATE APPROACHES

Method	Scene15
Improved classifiers based on NN/SVM/Kernel/KNN	
Kernel codebook [32]	76.6
Object-to-class kernels [33]	88.8
KNN with localized multiple kernel [34]	89.1
Label Consistent K-SVD, Spatial pyramid [35]	92.9
Sparse representation-based methods	
Linear spatial pyramid, sparse coding [36]	80.3
Laplacian sparse coding, feature combination [37]	88.9
Recent feature coding methods	
Feature fusion [38]	71.6
Visual word ambiguity [39]	76.7
Hard assignment [40]	81.4
Soft assignment [41]	82.2
Centrist, Spatial PACT [42]	83.9
Hierarchical networks	
Feature pooling [43]	80.6
Multilayer ELM, SIFT features [29]	82.4
Sparse coding, Max-pooling[44]	84.3
Six-layer deep network, Macro Feature [45]	85.4
Five-layer manifold deep network [46]	86.9
pre-trained deep features [47]	
SVM, with ImageNet pretrained AlexNet deep features	84.0
SVM, with Places365 pretrained AlexNet deep features	89.3
SVM, with ImageNet pretrained GoogleLeNet deep features	85.0
SVM, with Places365 pretrained GoogleLeNet deep features	91.3
SVM, with ImageNet pretrained VGG16 deep features	86.3
SVM, with Places365 pretrained VGG16 deep features	92.0
Human-level Performance[1] [48]	**95.0**
Our architecture without pretrained deep features	
Our method, Spatial pyramid features, single channel	**98.2**

combinations and leading alternate approaches. For example, Zhou *et al*. [48] terms a new data set (Places data set), which almost contains more than 7 million images from more than 300 place categories, making it the largest image database of scenes and places so far. After pretrained this large-scale data set, the performance of Scene15 obtained by [48], which is approaching nearly human-level performance at 95%. However, our best result is 98.2% without pretrained methods, which is even higher than human-level performance. As per our knowledge (see Fig. 4), label consistent-KSVD (LC-KSVD) [35] is the current leading results on the Scene15 database. Thus, it can be indicated that our approach can provide a better performance than other current leading methods. Note also that our method uses single features with a three-layer network, while some other state-of-the-art approaches combine multiple features, feature fusion methods, and even pretrained super-large data sets (Places data set).

C. Caltech101/256

In this section, we test our approach on two object recognition data sets.

1) The Caltech101 data set contains 9144 images in 102 object categories with the number of images per category varies from 31 to 800. Following the common experimental settings, we train on 15 and 30 samples per category and test on the rest.

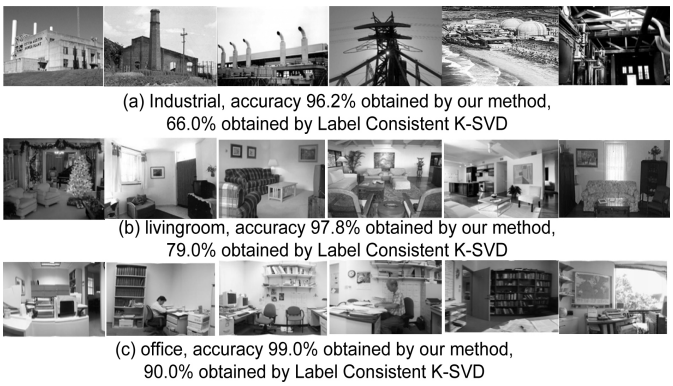


Fig. 4. Example images from classes with comparison classification accuracy from the Scene15 data set. The full name of LC-KSVD. (a) Industrial, accuracy 96.2% obtained by our method, 66.0% obtained by label consistent K-SVD. (b) Living room, accuracy 97.8% obtained by our method, 79.0% obtained by label consistent K-SVD. (c) Office, accuracy 99.0% obtained by our method, 90.0% obtained by label consistent K-SVD.

2) The Caltech256 data set contains 30 607 images of 257 categories. There are at least 80 images per category. Compared to the Caltech101 data set, it is much more difficult due to the large variations in object location, pose, and size.

For Caltech101, we evaluated our approaches on both 15 and 30 random training images per class and compare with more than 30 state-of-the-art approaches including LLC [52], sparse coding [36], hierarchical deep network [66], color fusion features [57], deconvolutional network [67], Alexnet [65], 16-layer VGG network, 19-layer VGG network, and so on. The comparisons are given in Table III. We use three subnetwork nodes (100 hidden nodes in each subnetwork node, i.e., feature dimensionality equals 100) with five channels (totally 15 channels) for 15 and 30 training samples per category. Our best result is 89.1%, whereas compared methods score 70–80%. Our approach consistently outperforms all the competing approaches. The basic reason for the good recognition performance, even with only a few training examples, is that the optimal features extracted and combined from multiple layers encourages the input signals from the same class to have similar codes and those from different classes to have dissimilar codes.

For Caltech256, we compared our results with the state-of-the-art methods. The comparisons are given Table III as well. We used eight subnetwork nodes (200 hidden nodes in each subnetwork node, i.e., feature dimensionality equals 200) for 30 training samples per category. Our approach could provide a performance comparable to that of other current leading methods. Note also that our method uses single features with a three-layer network, while some other state-of-the-art approaches combine multiple features, feature fusion methods, or much more complex network structure.

Furthermore, pretraining big image data can boost image recognition accuracy. For example, the ImageNet pretrained CNN [13], [69] can provide nearly 85%–88% and 70%–77% accuracy on Caltech101 and Caltech256, respectively.

TABLE III

CALTECH-101 CLASSIFICATION ACCURACY FOR OUR METHOD AGAINST OTHER STATE-OF-THE ART METHODS

Method	Caltech101(30 Tr.)	Caltech101(15 Tr.)	Caltech256(30 Tr.)
Improved classifiers based on NN/SVM/Kernel/KNN			
Nearest-neighbor NN([73])	70.4	65.0	26.8
KNN-SVD [74]	66.2	59.1	-
Object-to-class kernels [33]	65.5	-	-
Kernel codebook [32]	64.2	-	-
Kernel spatial pyramid [35]	73.6	67.7	28.9
Large margin learning [78]	-	68.0	-
Sparse representation-based methods			
Sparse representation [49]	70.7	64.9	-
Sparse coding [36]	73.2	67.0	27.7
LLC [52]	73.4	65.4	41.2
Sparse kernel SVD [50]	73.2	65.2	-
Multi-task spparse representation, combination features [51]	-	70.2	-
laplacian sparse coding, feature combination [37]	-	-	35.7
Low-rank representation learning [53]	73.6	66.1	-
Recent feature coding methods			
Local pyramidal descriptors [58]	80.2	71.5	44.7
Hierarchical matching pursuit feature [59]	81.4	-	48.0
image similarity search [55]	69.5	61.0	-
Hard assignment [40]	64.4	56.4	-
Visual word ambiguity [39]	64.1	-	27.2
Soft assignment [41]	76.5	-	-
Feature pooling [43]	75.1	-	-
Color fusion features [57]	76.2	-	-
Feature fusion[56]	69.7	61.2	-
Natural image statistics [60]	78.5	70.1	-
Hierarchical networks: Deep network or others			
Convolutional deep network [65]	66.3	-	-
Deconvolutional network [67]	66.9	58.6	-
multi-way local pooling [63]	77.1	-	-
Sparse coding, max-pooling[44]	75.7	-	-
Multilayer ELM, SIFT features [29]	77.6	71.7	38.2
Six-layer deep network, Macro Feature [45]	79.7	72.1	41.7
Hierarchical sparse coding, deep network [66]	74.0	-	-
*Our architecture **without** pretrained deep feature*			
Ours, HMP + SIFT feature, single channel	**88.9**	**83.9**	**54.8**
Ours, HMP + SIFT + spatial pyramid feature, 3 channels	**89.5**	-	-

TABLE IV

CALTECH101/256 CLASSIFICATION ACCURACY FOR OUR METHOD AGAINST OTHER CNN PRETRAINED MODELS

Method	Caltech101(30 Tr.)	Caltech256(30 Tr.)
AlexNet deep features		
SVM, Places205-pretrained AlexNet deep features [47]	65.3	34.3
SVM, Places365-pretrained AlexNet features [47]	66.4	46.5
SVM, ImageNet-Pretrained AlexNet features	82.7	-
SVM, ImageNet-pretrained AlexNet features [47]	87.7	67.0
Ours, ImageNet-Pretrained AlexNet Features, single channels	**88.4**	**68.0**
VGG deep features		
SVM, Places205-pretrained VGG16 deep features [47]	67.6	49.2
SVM, ImageNet-pretrained VGG16 deep features [47]	88.4	75.0
Ours, ImageNet-pretrained VGG deep features, single channels	**89.2**	**75.8**
Deep network, features fusion or model fusion		
ImageNet-pretrained CNN features, combination feature [69]	86.5	70.6
ImageNet-pretrained VGG-16, transfer learning [69]	92.1	73.2
ImageNet-pretrained ResNet-50, transfer learning, 20 training epoches	93.0	79.3
pretrained **Extended ImageNet (2000 Classes)** VGG-16 transfer learning [68]	91.8	85.0[a]
pretrained **Extended ImageNet (2000 Classes)** VGG-19 transfer learning [68]	92.3	85.1[a]
Ours, HMP + SIFT + spatial pyramid feature + ImageNet-Pretrained AlexNet + VGG-16 Deep Features, single channels	**93.9**	**-**
Ours, SIFT + spatial pyramid feature + ImageNet-Pretrained AlexNet + ResNet-50 + VGG-16 Deep Features, single channel	-	**84.1**

[a] Different training settings. For Caltech256, [68] uses 60 training images per a category, while our method uses 30 training images per a category.

Thus, it is interesting to compare our results with these pretrained CNN methods. Table IV provides the comparison results of Caltech101/256 classification accuracy. It should be noted that Extended ImageNet includes more than 30 000 000 images in 2000 categories, which is an unprecedented number. Using such pretrained features definitely can provide a very competitive performance. However, quite surprisingly, our approach, with 20–30-min training time, outperforms all the competing CNN approaches, even including 19-layer VGG network with extended ImageNet features.

D. CIFAR10

The CIFAR10 data set consists 60 000 color images in ten categories including airplane, bird, automobile, cat, dog, frog, deer, ship, horse, and truck. The number of images per class is 6000. There are 50 000 images (5000 per class) for training and the remaining parts for testing.

For the comparison, we separate our experimental results into three categories: single-layer classifiers, CNN networks, and CNN pretrained networks. This is because networks based on DL structure normally provide a better generalization performance, but with additional computational workloads. For AlexNet and 16-VGG network, we use a momentum of 0.9 and start with a learning rate of 0.001 with a single GTX 1060 GPU. As seen from Table VII, it is easy to accept that CNN networks, in general, provide a better performance when going deeper, which is consistent with the results showed in this table that 110-layer ResNet obtained the best performance among all the CNN models. In the proposed method, we used four subnetwork nodes (400 hidden nodes in each subnetwork node) with a single channel. But quite interestingly, by using such optimized features, our relatively simple model with several minutes training time could further boost the recognition performance. For example, the top-1 accuracy of AlexNet with ImageNet pretrained features is 88.5%, but our model with the same features offers 90.7% accuracy.

E. Places365 Data Set

In the recent years, large-scale databases including ImageNet and Places205 data sets are widely selected for image recognition. Well-known CNN models (AlexNet, GoogleNet, inception model, VGG model, and Resnet model) also choose these data sets to build pretrained models. In order to test the performance of the proposed method on large-scale data sets, here, we use an extended Places205 data set, called Places365 released by Massachusetts Institute of Technology in 2016. Similar to the previous Places205, the Places365-Standard database [47] has 365 categories with 1 803 460 training images with the image number per class varying from 3068 to 5000. To further test our proposed method with other-state-of-the-art methods, comparison experiments are conducted on the Places365 data set. Due to the computing resource limitation, we randomly select 1000 images per class from the original Places365 to reduce its scale. Thus, the data set used in the experiment has totally 365 000 images with 365 categories. All the experimental results are generated by 800 images per category for training, and the remaining 200 images per category are used for testing.

In order to obtain fair comparison experimental results, we use some well-known pretrained models (AlexNet, 16 layers VGG net, GoogleNet) to extract deep features or to conduct transfer learning tasks. Because ImageNet and Places365 are two different databases, the specialty of the units in the object-centric CNN (ImageNet) and scene-centric CNN (Places365) yield very different performances of generic visual features on a variety of recognition benchmarks. Here, we evaluate the scene-centric data set (Places365) performances of the deep features from a object-centric CNN model (ImageNet pretrained model).

The proposed method is conducted in MATLAB 2009 with 128-GB memory and an E5-2604 1.8-Hz CPU workstation. We only use CPU without any parallel computing algorithm. For other CNN methods, we used GTX 1060 GPU (6 GB) running under MATLAB 2017a environment. In order to compare the performance with CNN models, here, we perform transfer learning to obtain the experimental results to speed up training. Here, we set max epochs equals 100 and initial learning rate equals 0.001.

As seen from Table VI, the profits of our method are obvious. Our method with ImageNet feature performs much better than the ImageNet feature plus SVM baseline. Importantly, as expected, our method also outperformed ImageNet-VGG transfer learning model and ImageNet-AlexNet transfer learning model with 2–3% margin due to their further deeper features. Furthermore, although the category number of Place365 is quite larger than Place205 data set, our method provides more than 5% accuracy boosted than the results of SVM on Place205 data set. Note that for the training time, our method outperforms other state-of-the-art methods with tremendous advantage of training speed. For example, VGG-16 model requires more than 7 days with 44.2% accuracy, while our method provides 46.4% accuracy with only 36 min.

F. Structure and Parameters Sensitivity

In this section, we carry out several experiments to show how the proposed approach is reliable. The experimental results are shown in Table VIII. Inspired by reviewers, we conduct our experiments in the following three ways.

1) *Our Method With Fixed Structure:* We use the network structure given in Table I, including number of subnetwork neurons, number of neurons in each subnetwork neuron, and number of data channel. We also use normal distribution to randomly generate initial weights in (4).
2) *Our Method With Random Structure:* We randomly select the structure parameters to evaluate the performance sensitivity. Here, number of subnetwork neurons is randomly selected from [4, 10], number of neurons in each subnetwork neuron is randomly generated from [200, 1000]. Furthermore, we randomly pick two normal distribution random process or orthogonal random process to generate weights in (4). The orthogonal random process: the initial parameters are obtained by orthogonal random process

$$\begin{aligned} \mathbf{H}_f &= G(\mathbf{a}_f, b_f, \mathbf{x}) \\ (\mathbf{a}_f)^T \cdot \mathbf{a}_f &= \mathbf{I}, \quad (b_f)^T \cdot b_f = 1 \end{aligned} \tag{22}$$

3) Our method with rand features order.

G. Experimental Discussion

In the experiment, we evaluate our proposed method on several widely selected databases including a small data set Scene-15, medium data sets Caltech101/256 and CIFAR10, and a large-scale data set Places365. Based on the experimental results, we sight the following judgments.

TABLE V
COMPARISON PERFORMANCE OF CIFAR10 WITHOUT DATA ARGUMENTATION

Method	CIFAR 10
Improved classifiers based on NN/SVM/Kernel/KNN	
Linear SVM, Improved local coordinate coding [79]	74.5
Extreme Learning Machine, Kernel coding [64]	78.0
CNN networks training from scratch	
sumproduct network with deep architecture [72]	84.1
Deep attention selective networks [71]	90.7
Deep-supervised Nets [70]	90.2
96-layers Recurrent convolutional network [16]	89.7
160-layers Recurrent convolutional network [16]	91.3
The all convolutional Net [80]	90.9
110-layer ResNet [14]	93.6
40-layer DenseNet, 300 training epoches, batch size 24	92.6
40-layer DenseNet, 300 training epoches, batch size 64[15]	93.0
100-layer DenseNet, 300 training epoches, batch size 24	94.0
100-layer DenseNet, 300 training epoches, batch size 64[15]	94.2
Pretrained CNN mdoel with fine tuning	
AlexNet, pretrained ImageNet Alexnet, transfer learning [12]	88.0
Network in Networks, pretrained by ImageNet dataset, transfer learning [18]	89.6
Pretrained ImageNet AlexNet model, transfer learning, 10 epoches (Training time: 1.2 hours)	88.5
Pretrained ImageNet 16-layer VGG model, transfer learning, 10 epoches (Training time: 15.4 hours)	94.2
Pretrained ImageNet GoogleNet, transfer learning, 20 epoches (Training time: 3.5 hours)	95.4
Our architecture with transfer learning or deep features	
Our method, pretrained ImageNet Alexnet, transfer learning features	90.7
Our method, 40-layer DenseNet deep learning features (300 training epoches, batch size 24)	92.9
Our method, 100-layer DenseNet deep learning features (300 training epoches, batch size 24)	94.2
Our method, pretrained ImageNet VGG-16, transfer learning features	94.6
Our method, pretrained ImageNet GoogleNet, transfer learning features	**95.5**

TABLE VI
CLASSIFICATION ACCURACY ON SCENE-CENTRIC DATABASES FOR THE DEEP FEATURES OF OBJECT-CENTRIC DATABASES (IMAGENET). ALL THE ACCURACY IS THE TOP-1 ACCURACY

Method	Dataset	Top-1 Accuracy	Top-1 Accuracy (random structure)	Training Time
SVM, ImageNet-pretrained AlexNet deep features	Place365	32.7	-	24.6 hours
SVM, ImageNet-pretrained VGG16 deep features	Place365	36.8	-	24.1 hours
Ours, ImageNet-pretrained AlexNet deep features	Place365	34.8	34.7	18 min.
Ours, ImageNet-pretrained VGG16 + AlexNet deep features	Place365	40.9	40.7	45 min.
AlexNet, ImageNet-pretrained AlexNet model, Transfer Learning	Place365	42.5	-	47.6 hours
16 layers VGG, ImageNet-pretrained VGG model, Transfer Learning	Place365	44.2	-	> 7 days
Ours, ImageNet-pretrained VGG transfer learning features	Place365	45.2	45.3	**18 min.**
Ours, ImageNet-pretrained AlexNet deep features + VGG transfer learning features	Place365	**46.2**	46.2	**36 min.**

1) CNN Methods Versus Traditional Methods: CNN networks consistently outperform other non-CNN methods across all the tested databases. As seen from the Table VII, traditional methods may have advantages on relatively small databases such as Scene-15. However, CNN-based network could provide a much higher recognition accuracy for other medium or large data sets, which is consistent with recent research results.

2) Proposed Method With Deep Features: In the experiment, our method without CNN training, as seen from Tables II, IV, and VII, surprisedly beats CNN networks with fine-tuning regarding testing accuracy and training speed. In other words, only with pretrained CNN feature (deep feature), our method without any end-to-end training could obtain a competitive performance than an end-to-end trained CNN model. Due to such property, our method saves a lot of training time compared to CNN models. However, it should be noted that our method can be considered as a classifier, which needs convolutional layers to extract features from raw images. The proposed method can be useful for final stage training but is unable to extract useful features from raw images.

3) Proposed Method With Trained Deep Features: We also try to evaluate the performance of our method with trained deep features on CIFAR10 and Places365 databases. As seen from Figs. 5 and 6 and Tables V–VII, it is no doubt that our method could further provide a performance boost from an end-to-end trained CNN model. It is reasonable to mention that the proposed method should give a better performance than trained CNN models because it reuses the trained CNN deep features through a new classifier, which requires the extra computational cost. For example, on CIFAR10 (Table VII), our method reuses VGG-16 trained deep features from the last fully connected layer (FC7) of VGG-16, requiring additional

TABLE VII
COMPARISON PERFORMANCE OF FOUR DATA SETS WITHOUT DATA ARGUMENTATION

Method	No. of Param.	Testing Accuracy	Innate priors	CNN Training	Computation Time
		Scene15 dataset			
AlexNet	62.3M	88.0	ImageNet	20 epoches	181 second
VGG-16	138M	92.4	ImageNet	20 epoches	1305 second
GoogleNet	6.8M	90.9	ImageNet	20 epoches	418 second
ResNet50	**0.8M**	88.8	ImageNet	20 epoches	337 second
Non-CNN method [35]	-	92.9	None	-	-
Ours	4.9M	**98.2**	None	**No tuning**	**20 second**
		Caltech101 dataset			
AlexNet	62.3M	88.4	ImageNet	10 epoches	37 min.
VGG16	138M	92.1	ImageNet	20 epoches	1.3 hours
GoogleNet	6.8M	92.1	ImageNet	20 epoches	13.4 min.
ResNet50	**0.8M**	93.0	ImageNet	20 epoches	35 min.
DenseNet-100layer	7.2M	93.5	ImageNet	30 epoches	33 min.
Non-CNN method[59]	-	81.4	None	-	-
Ours without deep feature	13.4M	89.5	None	**No tuning**	**6 min.**
Ours	15.3M	**93.9**	ImageNet	**No tuning**	8 min.
		Caltech256 dataset			
AlexNet	62.3M	61.5	ImageNet	20 epoches	23 min.
VGG16	138M	73.2	ImageNet	20 epoches	1.8 hours
GoogleNet	6.8M	76.2	ImageNet	20 epoches	47 min.
ResNet100	**0.8M**	79.3	ImageNet	20 epoches	1.7 hours.
Non-CNN method[59]	-	48.0	None	-	-
Ours	65.2M	**84.1**	ImageNet	**No tuning**	**21 min.**
		CIFAR10 dataset			
AlexNet	62.3M	88.5	ImageNet	10 epoches	1.2 hours
VGG16	138M	94.2	ImageNet	30 epoches	15.4 hours
GoogleNet	6.8M	95.4	ImageNet	20 epoches	3.5 hours
DenseNet-40	**0.8M**	92.6	None	300 epoches	75.7 hours.
Non-CNN method: Kernel coding [64]	-	78.0	None	-	**114 second**
Ours	6.6M	**95.5**	ImageNet	20 epoches	3.5 hours + 6 min.

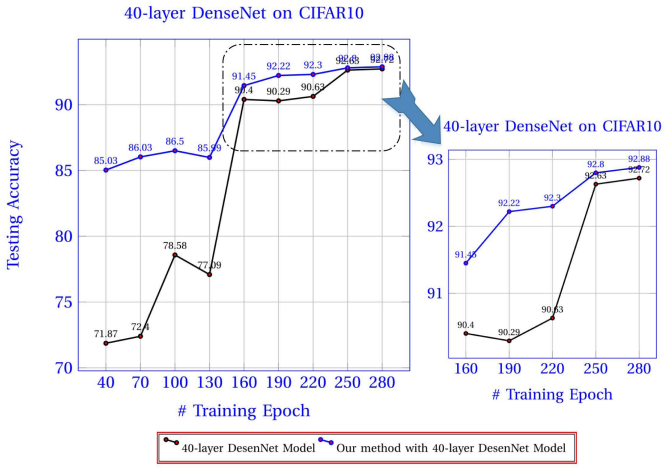


Fig. 5. Top-1 testing accuracy of CIFAR10: Our method with 40-layer DenseNet versus 40-layer DenseNet.

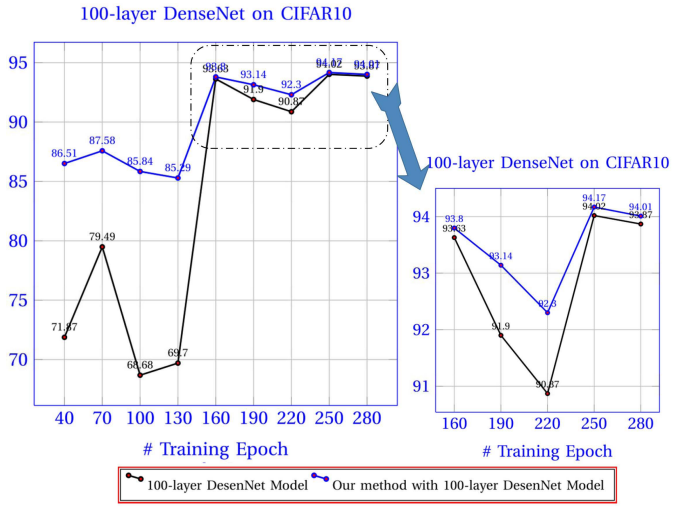


Fig. 6. Top-1 testing accuracy of CIFAR10: Our method with 100-layer DenseNet versus 100-layer DenseNet.

6-min training time. However, let us analyze the performance gap regarding computation time and recognition accuracy. As CIFAR10 has only ten categories, our method with additional 6 min only provides 0.1% accuracy boost. However, on large-scale database Places365 (see Table VI), our method with 18-min computation time can provide 1.1% accuracy boost compared to VGG-16 CNN model. While around 1.7% accuracy boost has been obtained by switching CNN models from AlexNet to VGG-16, but which costs nearly five days more.

4) Parameter Numbers Versus Computation Time: As seen from Table VII, VGG-16 with 138 million parameters requires 1.8 h to end a Caltech256 training task, while 50 layers ResNet with 0.8 million parameters also needs 1.7 h on the same database. The AlexNet has the fastest training speed

TABLE VIII
PERFORMANCE COMPARISON WITH DIFFERENT STRUCTURES (MEAN: AVERAGE TESTING ROOT MEAN SQUARE ERROR; TIME: AVERAGE TRAINING TIME)

Datasets	Ours with fixed structure		Ours with random structure		Ours with random features order	
	Mean	time(s)	Mean	time(s)	Mean	time(s)
Scene15	98.2	0.0008	98.2	0.0010	98.2	0.0008
Caltech101	93.9	0.0016	**94.0**	0.0019	93.9	0.0017
Caltech256	84.1	0.0013	83.9	0.0017	84.1	0.0012
CIFAR10	95.5	0.0003	95.5	0.0004	95.5	0.0003
Places365	46.2	0.0009	46.2	0.0010	46.2	0.0008

across all the test data sets, which has a more complicated structure (62.3 million) compared to that of ResNet (0.8 million). Thus, variety factors, including the number of parameters, number of layers, and learning method, significantly influence the training computation time. As noniterative learning strategy is used in our proposed method, the advantage of computation time is obviously. As seen from Table VII (Caltech256), our method with 65 million parameters requires 21 min. training time, while 50-layer ResNet with only 0.8 million parameters needs 1.7 h. For large-scale data set Places365 (29.2 million images), our method only required 38 min with two data channels.

IV. CONCLUSION

In this paper, we present a hierarchical network scheme with hundreds of layers for image recognition. The problem is approached from three main ways: 1) features extracted from single-subnetwork neuron rather than hidden layers; 2) low-dimensional subspace features fused by different operators; and 3) heterogeneous features produced by multiple data channels. Furthermore, this paper indicates that compared with some well-known CNN model, deep features with our method could provide a better performance. The experimental results show that our method functions as a feature extractor and a classifier, and it performs better than other relevant state-of-the-art methods.

It should be noted that our method currently is incapable of batch-by-batch learning, which is the major weakness of our method. In other words, the proposed method has to calculate all the training samples once, which needs a lot of computation memory. It is worth to investigate a batch-by-batch learning way with the proposed method.

REFERENCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[2] J. Weng, N. Ahuja, and T. S. Huang, "Cresceptron: A self-organizing neural network which grows adaptively," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Baltimore, MD, USA, vol. 1, Jun. 1992, pp. 576–581.

[3] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 153–160.

[6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[7] M. Chen, K. Q. Weinberger, Z. Xu, and F. Sha, "Marginalizing stacked autoencoders," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 3849–3875, 2015.

[8] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[9] 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.

[10] N. N. Schraudolph and T. J. Sejnowski, "Unsupervised discrimination of clustered data via optimization of binary information gain," in *Proc. Adv. Neural Inf. Process. Syst.*, San Mateo, CA, USA, 1993, pp. 499–506.

[11] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[13] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014. [Online]. Available: http://www.bmva.org/bmvc/2014/papers/paper054/index.html

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1–9.

[16] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 3367–3375.

[17] Y. Deng, Z. Ren, Y. Kong, F. Bao, and Q. Dai, "A hierarchical fused fuzzy deep neural network for data classification," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 4, pp. 1006–1012, Aug. 2017.

[18] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400v3

[19] H. Liu, J. Qin, F. Sun, and D. Guo, "Extreme kernel sparse learning for tactile object recognition," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4509–4520, Dec. 2017.

[20] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[21] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, Apr. 1994.

[22] C. L. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 1, pp. 62–72, Feb. 1999.

[23] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[24] H. White, "An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks," in *Proc. Int. Joint Conf. Neural Netw.*, Washington, DC, USA, 1989, pp. 451–455.

[25] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[26] G.-B. Huang, "What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cognit. Comput.*, vol. 7, no. 3, pp. 263–278, Jun. 2015.

[27] J. Cao, K. Zhang, H. Yong, X. Lai, B. Chen, and Z. Lin, "Extreme learning machine with affine transformation inputs in an activation function," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2018.2877468.

[28] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feedforward neural networks with random weights," in *Proc. Int. Conf. Neural. Netw.*, Hague, The Netherlands, Aug./Sep. 1992, pp. 1–4.

[29] Y. Yang and Q. M. J. Wu, "Multilayer extreme learning machine with subnetwork nodes for representation learning," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2570–2583, Nov. 2016.

[30] Y. Yang, Q. M. J. Wu, W.-L. Zheng, and B.-L. Lu, "Eeg-based emotion recognition using hierarchical network with subnetwork nodes," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 2, pp. 408–419, Jun. 2017.

[31] Y. Yang and Q. M. J. Wu, "Extreme learning machine with subnetwork hidden nodes for regression and classification," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2885–2898, Dec. 2016.

[32] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in *Proc. IEEE Eur. Conf. Comput. Vis.*, Marseille, France, 2008, pp. 696–709.

[33] L. Zhang, X. Zhen, and L. Shao, "Learning object-to-class kernels for scene classification," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3241–3253, Aug. 2014.

[34] Y. Han, K. Yang, Y. Ma, and G. Liu, "Localized multiple kernel learning via sample-wise alternating optimization," *IEEE Trans. Cybern.*, vol. 44, no. 1, pp. 137–148, Jan. 2014.

[35] 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.

[36] L. Yang, R. Jin, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 1794–1801.

[37] S. Gao, I. W.-H. 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.*, Jun. 2010, pp. 3555–3561.

[38] J. Yu, D. Tao, Y. Rui, and J. Cheng, "Pariwise constraints based multiview features fusion for scene classification," *Pattern Recognit.*, vol. 46, pp. 483–496, Feb. 2013.

[39] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1271–1283, Jul. 2010.

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

[41] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2486–2493.

[42] J. Wu and J. M. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1489–1501, Aug. 2011.

[43] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 111–118.

[44] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2559–2566.

[45] H. Goh, N. Thome, M. Cord, and J.-H. Lim, "Learning deep hierarchical visual feature coding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 12, pp. 2212–2225, Dec. 2014.

[46] Y. Yuan, L. Mou, and X. Lu, "Scene recognition by manifold regularized deep learning architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2222–2233, Oct. 2015.

[47] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2017.

[48] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. Neural Inf. Process. Syst.*, 2014, pp. 487–495.

[49] 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.

[50] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[51] X.-T. Yuan, X. Liu, and S. Yan, "Visual classification with multitask joint sparse representation," *IEEE Trans. Image Process.*, vol. 21, no. 10, pp. 4349–4360, Oct. 2012.

[52] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 3360–3367.

[53] Y. Zhang, Z. Jiang, and L. S. Davis, "Learning structured low-rank representations for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 676–683.

[54] X. Peng, R. Yan, B. Zhao, H. Tang, and Z. Yi, "Fast low rank representation based spatial pyramid matching for image classification," *Knowl.-Based Syst.*, vol. 90, pp. 14–22, Dec. 2015.

[55] P. Jain, B. Kullis, and K. Grauman, "Fast image search for learned metrics," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.

[56] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE Conf. Comput. Vis.*, Kyoto, Japan, Sep./Oct. 2009, pp. 221–228.

[57] F. S. Khan, J. van de Weijer, and M. Vanrell, "Modulating shape features by color attention for object recognition," *Int. J. Comput. Vis.*, vol. 98, no. 1, pp. 49–64, May 2012.

[58] L. Seidenari, G. Serra, A. D. Bagdanov, and A. Del Bimbo, "Local pyramidal descriptors for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 1033–1040, May 2014.

[59] L. Bo, X. Ren, and D. Fox, "Multipath sparse coding using hierarchical matching pursuit," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 660–667.

[60] C. Kanan and G. Cottrell, "Robust classification of objects, faces, and flowers using natural image statistics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2472–2479.

[61] Y. Chai, V. Lempitsky, and A. Zisserman, "BiCoS: A Bi-level co-segmentation method for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Nov. 2011, pp. 2579–2586.

[62] R. Khan, J. van de Weijer, F. S. Khan, D. Muselet, C. Ducottet, and C. Barat, "Discriminative color descriptors," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 2866–2873.

[63] Y.-L. Boureau, N. L. Roux, F. Bach, J. Ponce, and Y. LeCun, "Ask the locals: Multi-way local pooling for image recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Barcelona, Spain, Nov. 2011, pp. 2651–2658.

[64] D. Lam and D. Wunsch, "Unsupervised feature learning classification with radial basis function extreme learning machine using graphic processors," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 224–231, Jan. 2016.

[65] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, "Learning invariant features through topographic filter maps," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1605–1612.

[66] K. Yu, Y. Lin, and J. Lafferty, "Learning image representations from the pixel level via hierarchical sparse coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1713–1720.

[67] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergu, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2528–2535.

[68] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556v6

[69] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. IEEE Eur. Conf. Comput. Vis.*, Zürich, Switzerland, 2014, pp. 818–833.

[70] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. (2015). "Deeply-supervised nets." [Online]. Available: https://arxiv.org/abs/1409.5185v2

[71] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Proc. 27th Adv. Neural Inf. Process. Syst.*, 2014, pp. 3545–3553.

[72] R. Gens and P. Domingos, "Discriminative learning of sum-product networks," in *Proc. 25th Adv. Neural Inf. Process. Syst.* New York, NY, USA: Curran Associates, 2012, pp. 3239–3247.

[73] O. Boiman, M. Shechtman, and E. Irani, "In defense of nearest-neighbor based image classification," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Zürich, Switzerland, Jun. 2008, pp. 1–8.

[74] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, New York, NY, USA, Jun. 2006, pp. 2126–2136.

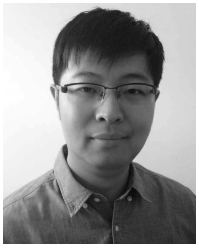
[75] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Bhubaneswar, India, Dec. 2008, pp. 722–729.

[76] S. Ito and S. Kubota, "Object classification using heterogeneous Co-occurrence features," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 701–714.

[77] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2691–2698.

[78] A. Frome, Y. Singer, F. Sha, and J. Malik, "Learning globally-consistent local distance functions for shape-based image retrieval and classification," in *Proc. IEEE Conf. Comput. Vis.*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[79] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 1215–1222.
[80] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806v3



Yimin Yang (S'10–M'13) received the Ph.D. degree in pattern recognition and intelligent system from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2013.

From 2014 to 2018, he was a Post-Doctoral Fellow with the University of Windsor, Windsor, ON, Canada. He is currently an Assistant Professor with the Computer Science Department, Lakehead University, Thunder Bay, ON, Canada. His current research interests include artificial neural networks, signal processing, and robotics.

Dr. Yang is a Program Committee Member of some international conferences. He was a recipient of the Outstanding Ph.D. Thesis Award of Hunan Province and the Outstanding Ph.D. Thesis Award Nominations of Chinese Association of Automation, China, in 2014 and 2015, respectively. He has been serving as a reviewer for many international journals of his research field and a Guest Editor of multiple journals.

Q. M. Jonathan Wu (M'92–SM'09) received the Ph.D. degree in electrical engineering from the University of Wales, Swansea, U.K., in 1990.

In 1995, he joined the National Research Council of Canada, Vancouver, Canada, where he became a Senior Research Officer and a Group Leader. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada. He has authored or co-authored more than 300 peer-reviewed papers in computer vision, image processing, intelligent systems, robotics, and integrated microsystems. His current research interests include 3-D computer vision, active video object tracking and extraction, interactive multimedia, sensor analysis and fusion, and visual sensor networks.

Dr. Wu holds the Tier 1 Canada Research Chair in automotive sensors and information systems. He was an Associated Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. He is an Associate Editor of the IEEE TRANSACTION ON CYBERNETICS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and *Cognitive Computation*. He has served on technical program committees and international advisory committees for many prestigious conferences.