Review of AutoEncoder and Its Variants: A Comparative Perspective from Target Recognition in SAR Images

Ganggang Dong, Guisheng Liao, Hongwei Liu, and Gangyao Kuang

***Abstract*—The past decade has witnessed a resurgence of unsupervised learning with neural network, among which autoencoder and its variants have been studied widely. These models have been applied to the field of remote sensing recently. The representative include terrain classification, land-use scene recognition with hyperspectral image, PolSAR image classification, remote sensing imagery retrieval, semantic annotation, target recognition and detection in SAR images. Though many works have been done previously, a comprehensive review of the literature is absent, especially from a comparative perspective. This paper attempts to fill this gap. We summarize the contributions of the existing works first, followed by a systematic summary of autoencoder and its variants. More attention have been paid to the recent progresses in the remote sensing community. To verify the performance, multiple comparative studies have been pursued from the perspective of target recognition in SAR images. The experimental results prove the advantage of autoencoder neural network in comparison to the delicate-handcrafted features. In addition, some propositions concerning the configuration of deep neural network can be concluded accordingly. We go on to provide several hints for readers about the logical choice of a neural network architecture based on the task at hand. All of the experiments are implemented by the Keras and Theano library on Python 3.6. The source code of this paper was publicly released on https://ganggangdong.github.io/homepage.**

Keywords*—Autoencoder, target recognition, SAR, deep learning, neural network, convolutional, unsupervised learning.

I. Introduction

A. Background

IN the recent years, unsupervised feature learning with neural network architecture has become a new research hotspot [1]–[4]. The revival of interest in such deep network can be attributed to the development of the efficient optimization skills, by which the model parameters can be optimally estimated [5]. The milestone work has been done by G. Hinton and R. Salakhutdinov [6]. They propose to initialize the weights that allows deep autoencoder networks to learn the low-dimensional codes. The proposed encoding trick works much better than principal components analysis (PCA) in terms of dimension reduction.

Dr. Ganggang Dong, Prof. Guisheng Liao, and Prof. Hongwei Liu are with National Lab of Radar Signal Processing, Xidian University, Xi'an, China, 710000. E-mail: dongganggang@nudt.edu.cn

Prof. Gangyao Kuang is with Lab of Remote Sensing Information Processing, National University of Defense Technology.

This work was supported by the National Science Fund for Distinguished Young Scholars of China under Grant 61525105.

The basic neural network model include autoencoder [7], restricted Boltzmann machine (RBM) [8], convolutional neural network (CNN) [9]. The deep neural network architecture, such as stacked autoencoder, deep belief network, deep CNN, are formed by concatenating the basic models hierarchically. Due to the simple implementation and attractive computation cost, autoencoder has been utilized in many fields. Examples include but are not limited to natural language processing [10], image processing [11], object detection [12], biometric recognition [13], data analysis [14]. This paper mainly focuses on the applications to remote sensing. The representative include terrain classification, land-use recognition, image retrieval, semantic annotation, object detection, and target recognition in radar images and hyperspectral images [15]–[26]. Though a great many works concerning the family of topics have been done previously, a systematic summary is still absent. This paper attempts to fill the gap. We first review most of the related works. Their contributions have been categorized as fourfold.

(a) Information Fusion. In the early works, the predefined features or raw pixel values are combined by an autoencoder model, from which a refined representation can be learned. It is then connected with a softmax layer to achieve different classification tasks. Since each set of features represent certain kind of information, the autoencoder model can be actually viewed as a black-box to achieve information aggregation.

Z. Lin *et al.* design an autoencoder model for hyperspectral image classification [27], [28]. The spatial patches and the spectral bands of hyperspectral image are jointly considered for classification. The basic model and the stacked autoencoder are used to generate the shallow and deep representation. The spatial information and spectral information can be then combined by the autoencoder network. C. Tao *et al.* propose to learn a good representation adaptively from the unlabeled data [24]. They establish two learning schemes, sparse spectral feature learning and multiscale spatial feature learning. The learned spectral-spatial feature is then embedded into a linear support vector machine for classification. J. Geng *et al.* present a deep supervised and contractive neural network for SAR image classification [21], [29]. Three kinds of handcrafted features, gray level-gradient co-occurrence matrix, Gabor filter banks, and histogram of oriented gradient descriptors are jointly fed into a stacked autoencoder network. The learned representation is then input to a softmax classifier for land-cover classification. The similar thought has been employed in [30], where the geometric feature and the local texture feature are combined to train a stacked autoencoder for target recognition in SAR images. Inspired by the change detection algorithm, P. Planini and D. Gleich propose to assess the damage caused by fires [31]. They extract the higher order log cumulants of fractional Fourier transform from the tunable Q discrete wavelet transform. The obtained features are then used to train a stacked autoencoder model to distinguish the changed and unchanged areas. L. Zhang *et al.* design a framework to learn robust features from PolSAR data [32]. The local spatial information has been exploited to train a stacked sparse autoencoder model. The influence of neighbor pixels is controlled according to the spatial distance to the central pixel. To handle the presence of clouds, S. Malek *et al.* utilize an autoencoder neural network to recover the missing data in multispectral images [33]. They aim to modeling the relationship between a given cloud-free image (source image) and a cloud-contaminated one (target image). Two schemes, pixel-level and patch-level mapping are developed to exploit the spatial contextual information. The size of hidden layer is determined by a new solution that combines the minimum descriptive length criterion and a Pareto-like selection procedure.

(b) New Cost Function. Training an autoencoder model refers to estimating the model parameters, weight and bias. It is achieved by solving an optimizing problem whose objective function is composed of a loss function and certain regularization terms. The performance can be therefore improved by designing a task-specific objective function, into which certain information can be incorporated. The loss function and the regularization term can be tuned according to the task at hand. In addition, the unsupervised learning can be also converted to supervised fashion by designing a label-related cost function.

X. Ma *et al.* present a spatial updated deep autoencoder for hyperspectral image classification [17]. They present a new regularization term composing of the similarity of the hidden neurons weighted by the cosine distance of visible nodes. The contextual information can be then exploited. The similar thought has been utilized in [34], where the objective function is tuned according to the task of target recognition. The authors design a regularization term that measures the total difference of the hidden neurons. Since the class membership is incorporated into the objective function, it is actually a supervised learning fashion. Similarly, Wen Xie *et al.* present an improved objective for autoencoder model according to the task of PolSAR image classification [20]. The deviation of the reconstructed data to the initial value is measured by Wishart distance, rather than mean square error or cross entropy. K. Liang *et al.* learn a robust and discriminative feature representation by an improved model, smooth autoencoder [35]. The encoding of each sample is used to reconstruct its local neighbors. The learned representations are therefore consistent among the local neighbors and robust to small variations of input. M. Gong *et al.* propose a multi-objective sparse feature learning model based on autoencoder [36]. The model parameters are learnt by optimizing two objectives, reconstruction error and the sparsity of hidden units simultaneously.

(c) Combination with other learning scheme. To promote the performance further, some researchers propose to introduce the third party learning scheme, such as active learning, extreme learning machine (ELM). The key issue is how to utilize the additional-learning skill reasonably during the training of neural network.

Y. Sun *et al.* propose a hybrid plan, where autoencoder and active learning are simultaneously considered for hyperspectral image classification [37]. The most informative samples selected by active sampling algorithm are used to pre-train the autoencoder. Three active sampling strategies, mutual information, breaking ties, and random sampling are verified. J. Tang *et al.* present an extreme learning machine based hierarchical framework [38]. The compact and meaningful representations are obtained by unsupervised multilayer encoding and ELM-based sparse autoencoder. B. Hou *et al.* consider autoencoder model and super-pixel trick jointly for PolSAR images classification [23]. To integrate the contextual information, the RGB image formed by Pauli decomposition is used to produce superpixels. Multilayer autoencoder network is then employed to learning the features, by which the multiple categories for each pixel can be distinguished. Y. Zhou *et al.* combine the spectral feature learning and the spectral feature learning in a hierarchical fashion [39]. A shallow neural network, kernel extreme learning machine, is then embedded into the developed hierarchical network. F. Lv *et al.* achieve hyperspectral image classification with the ensemble of extreme learning machine and stacked autoencoder [19]. The latent representations learned from stacked autoencoder are fed into several ELM base classifiers, from which the inference can be drawn. Y. Chen *et al.* employed multilayer projective dictionary pair learning (MPDL) and sparse autoencoder jointly for PolSAR image classification [18]. MPDL is first used to extract the discriminative feature, followed by sparse autoencoder to get the nonlinear relationship between the elements of feature vectors in an adaptive way. E. Li *et al.* learn a mid-level feature representation by sparse autoencoder and pooling skill [22]. Sparse autoencoder is first employed to produce the relatively small number of convolutional features from the input dataset. The extended features are then extracted from the learned features by a multiple normalized difference method to compose a derivative feature set. The global statistics (histogram moments, mean, variance, standard deviation) are finally utilized to build the mid-level representation. H. Wu *et al.* propose a hybrid architecture, deep filter banks, for land-use scene classification [40]. They combine multicolumn stacked denoising sparse autoencoder and Fisher vector to automatically learn the representative and discriminative features in a hierarchical manner. H. Kim and A. Hirose consider PolSAR image classification by a quaternion autoencoder and a quaternion self-organizing map simultaneously [41]. A quaternion autoencoder is employed to extract feature information based on the natural distribution of PolSAR features. The extracted features are classified by the quaternion SOM in an unsupervised manner, by which new and more detailed land categories can be discovered.

(d) Achieve Transfer Learning. Though supervised deep learning methods are currently state-of-the-art, they require a great many labeled data. The labeled image benchmarks available are too small to train a deep supervised network effectively. To handle the problem, some researchers recommend

transfer learning by the large quantities of unrelated data, such as self-taught learning [42]. A popularly used idea is to rely on autoencoder. The small sample size and the limited training resource can be then dealt with.

R. Kemker and C. Kanan achieve hyperspectral image classification by self-taught learning [43]. They extract a bag of generalizable features (or filters) by training an unsupervised model on sufficiently large amount of unlabeled data that are distinct from the target dataset. The trained model is used to produce the discriminative features from small labeled target datasets. Two kinds of schemes are developed. The first designs a shallow architecture by means of independent component analysis, while the second presents a deep network via stacked autoencoder. Esam Othman *et al.* develop a deep model for land-use scene classification [26]. They generate the initial feature representation of scene image by a deep pretrained convolutional neural networks. The generated features are then fed into an autoencoder to refine the representation. The CNN features can be therefore transfered to the terminal learning architecture. X. Yao *et al.* consider autoencoder for automatic semantic annotation of high resolution optical satellite images [44]. They learn a high-level feature from an auxiliary satellite image dataset by an stacked discriminative sparse autoencoder. The learned high-level features are transferred to semantic annotation. The transferred representations are further fine-tuned in a weakly supervised fashion by the tile-level annotated training data. F. Zhang *et al.* design an unsupervised feature learning framework for land-use scene classification [3]. They extract a set of representative patches from the salient regions in the image data set. These unlabeled patches are then employed to generate a set of high-level features via sparse autoencoder. The scene is finally characterized by the statistics computed from the learned features. G. Chen *et al.* achieve land-use classification by an effective mid-level visual elements-oriented representation [15]. A library of pretrained part detectors, namely "partlet" is generated for mid-level visual elements discovery. Their responses to a large number of part detectors are then employed to represent the scene image. This is achieved by building a single-hidden-layer autoencoder and a single-hidden-layer neural network with an ℓ_0-norm sparsity constraint respectively.

Additionally, autoencoder has been also utilized to other related fields. Z. Shao *et al.* develop a deep learning based workflow for mapping forest above-ground biomass by integrating Landsat 8 and Sentinel-1A images with airborne light detection and ranging data [45]. They demonstrate the advantage of stacked sparse autoencoder network in comparison to five different prediction techniques including multiple stepwise linear regressions, k-nearest neighbor, support vector machine, back propagation neural networks, and random forest. H. Li and S. Misra train a variational autoencoder to generate the NMR-T2 distributions along a 300-ft depth interval in a shale petroleum system at 11000-ft depth below sea level [46]. The trained model successfully predicts the T2 distributions for 100 discrete depths at an R^2 of 0.75 and normalized root-mean-square deviation of 15%. A. Elshamli *et al.* generate the domain-invariant representations by denoising autoencoder and domain-adversarial neural networks [47]. The former plan builds an unsupervised feature learning followed by a supervised classification, while the latter plan jointly considers the invariant representation learning and classification during training. S. De *et al.* develop a new technique for the classification of urban areas in PolSAR image [48]. They leverage a synthetic target database for data augmentation. A new reference database is formed by the uniformly rotated and collated dataset. They are used to train a stacked autoencoder. The information in the augmented dataset is transformed into a compact representation. The classification is performed by a multilayer perception network. L. Windrim *et al.* extend the Spectral Angle-Stacked Autoencoder with the incorporation of a physics-based model for illumination [49]. The proposed algorithm learns a shadow invariant mapping without the need for any labeled training data, additional sensors, a priori knowledge of the scene or the assumption of Planckian illumination.

On account of the advances on deep learning, there is an encouraging evidence that the learned representation could produce a better performance than the canonical, predefined image feature. It is therefore no need to pay more attention to designing the delicate handcrafted features, as studied in the preceding works [50]–[60].

B. Contributions

Though many studies on autoencoder have been done previously, a comprehensive summary is still absent. Seldom work is dedicated to the appropriate choice of network architecture to remote sensing, *i.e.*, the impact of related factors. This is adverse to the further development of the technique. This paper attempts to fill the gap. We pursue a comprehensive review of the related studies. Multiple comparative experiments are performed from the viewpoint of target recognition in SAR image. Some qualitative propositions can be then concluded. We go on to provide some hints for the readers about the logical choice of an appropriate network architecture according to the application at hand. The contribution therefore includes:

- *The review of autoencoder and its variants.* Many improved versions had been developed since the optimization algorithms of deep network were presented [61]. We pursue a systematic review of autoencoder and its variants, including the shallow framework and the neural network architecture.
- *The verification of network configuration from a comparative perspective.* Though studied widely, seldom work devotes to studying the effect of related factors, especially from a comparative perspective. We verify the performance of autoencoder with multiple quantitative experiments. The experimental results demonstrate a sound proof on some conclusions.

C. Organization

The rest of this paper is organized as follows. Section II provides the basic autoencoder, and some improved versions. A quantitative comparison of autoencoder and principal component analysis (PCA) is pursued. The deep architecture is described in Section III. The implementation of classification is detailed in Section IV. We simply review linear regression and logistic regression, followed by multi-class logistic regression. Section V delineates the application of autoencoder to target recognition in SAR images. A short review of related works is pursued, followed by the architecture of deep network formed by SAR images. Section VI provides multiple comparative studies, by which the impact of related factors can be studied. Section VII concludes this paper and outlines the future work.

II. The Basic Model

The thought of deep learning has not been harnessed until the efficient optimization algorithms are developed [61], [62]. The deep network is usually composed of several basic models. In this section, we review the basic autoencoder and the improved versions. All of the notations used in this article are listed in TABLE I.

TABLE I. The notations used in this article.

$\mathbf{x}, \tilde{\mathbf{x}}$	visible units, corrupted visible units
$\mathbf{h}, \tilde{\mathbf{h}}$	hidden units, dropped hidden units
d_v, d_h	dimension of visible layer, and hidden layer
$\mathcal{W}_v, \mathcal{W}_h$	weight of visible units, and hidden units
$\mathbf{b}_v, \mathbf{b}_h$	bias of visible units, and hidden units
$\mathcal{J}(\cdot), \mathcal{L}(\cdot)$	cost function and loss function
$\sigma(\cdot), \delta(\cdot)$	activation function
$\rho,\ \hat{\rho}$	desired and real average activation of hidden units
$*,\ \otimes$	convolution and element-wise product

A. AutoEncoder

Autoencoder is a typical unsupervised learning algorithm. It aims to setting the target values to be equal to the original input. As shown in Fig. 1, a generic model is usually composed of three separate phases.

- Encoder: a set of linear feed-forward filters parameterized by the weight matrix and bias. (Feed-forward neural network)
- Activation: a nonlinear mapping that transforms the encoded coefficients into the range [0,1].
- Decoder: a set of reverse linear filters that produce the reconstruction of the input. (Back-propagation)

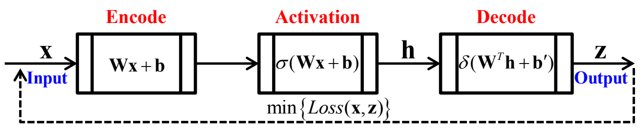


Fig. 1. The generic flowchart of autoencoder.

Autoencoder is a typical feed-forward neural network. It hooks together many of the simple "neurons". The output of a neuron can be the input of another. The parameters can be estimated by back-propagation algorithm. A "forward pass" is first run to compute the activations throughout the network, including the output value of the hypothesis. For each middle node, an "error term" that measures how much that node was responsible for any errors in the output is computed. For an output node, the difference between the network's activation and the true target value can be measured directly. It can be further used to renew the "error term".

Given the visible layer $\mathbf{x} \in \mathbb{R}^{d_v}$, autoencoder first maps it into a hidden layer $\mathbf{h} \in \mathbb{R}^{d_h}$ with a weight matrix $\mathcal{W}_v$, bias $\mathbf{b}$, and an activation function $\sigma(\cdot) : \mathbb{R} \mapsto [0, 1]$,

$$\mathbf{h} = \sigma(\mathcal{W}_v \mathbf{x} + \mathbf{b}) \tag{1}$$

The hidden units are then cast into the output layer $\mathbf{z} \in \mathbb{R}^{d_v}$ in a reverse fashion with weight matrix $\mathcal{W}_h$, bias $\mathbf{b}_h$ and activation $\delta(\cdot) : \mathbb{R} \mapsto [0, 1]$

$$\mathbf{z} = \delta(\mathcal{W}_h \mathbf{h} + \mathbf{b}_h) \tag{2}$$

To simplify the network architecture, the tied weights strategy $\mathcal{W}_v = \mathcal{W}_h = \mathcal{W}$ was usually employed. The parameters to be determined are $\{\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h\}$. The objective function to train an autoencoder is to minimize the cost function

$$\arg \min_{\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h} \mathcal{J}(\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h). \tag{3}$$

Given the training samples, the cost function is defined as,

$$\mathcal{J}(\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h) = \mathcal{L}(\mathbf{x}, \mathbf{z}) + \lambda g(\mathcal{W}) \tag{4}$$

where $\mathcal{L}(\mathbf{x}, \mathbf{z})$ is the loss function, and $g(\mathcal{W})$ is a regularization (weight decay). The comparative studies on loss function can be found in Section VI-A. The weight decay is the Frobenius norm of weight, $g(\mathcal{W}) = 0.5(\|\mathcal{W}_v\|_F^2 + \|\mathcal{W}_h\|_F^2)$. It tends to decrease the magnitude of the weights, and hence prevents over-fitting. The weight matrix and bias are obtained by the mini-batch stochastic gradient descent algorithm,

$$\begin{aligned}
\mathcal{W}^{(new)} &= \mathcal{W}^{(old)} - \alpha \frac{\partial}{\partial \mathcal{W}} \mathcal{J}(\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h) \\
\mathbf{b}_v^{(new)} &= \mathbf{b}_v^{(old)} - \alpha \frac{\partial}{\partial \mathbf{b}_v} \mathcal{J}(\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h) \\
\mathbf{b}_h^{(new)} &= \mathbf{b}_h^{(old)} - \alpha \frac{\partial}{\partial \mathbf{b}_h} \mathcal{J}(\mathcal{W}, \mathbf{b}_v, \mathbf{b}_h)
\end{aligned} \tag{5}$$

where α is the learning rate. $\mathcal{W}^{(old)}, \mathbf{b}_v^{(old)}, \mathbf{b}_h^{(old)}$ denote the old model parameters, and $\mathcal{W}^{(new)}, \mathbf{b}_v^{(new)}, \mathbf{b}_h^{(new)}$ are the renewed ones.

B. Sparse AutoEncoder

To exploit the inner structure of data, an additional regularization, the sparsity constraint on the hidden units is developed. A neuron whose output is close to 1 is activate, while the one whose output is close to 0 is inactive. Sparse autoencoder aims to limiting the neurons to be inactive most of the time.

Given a set of training samples, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$, the average activation of j-th hidden unit is $\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}[\tilde{\mathbf{h}}_j(\mathbf{x}_i)]$. Sparse autoencoder enforces the constraint $\hat{\rho}_j = \rho$, where ρ is the desired average activation value. Most activations of the hidden units are then near 0. The constraint is realized by a penalty

$$\sum_{j=1}^{d_h} \{\rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}\}. \tag{6}$$

It is the Kullback-Leibler (KL) divergence $\sum_{j=1}^{d_h} KL(\rho\|\hat{\rho}_j)$. The cost function of sparse autoencoder is

$$\mathcal{J}_{sparse} = \mathcal{L}(\mathbf{x},\mathbf{z}) + \beta \sum_{j=1}^{d_h} KL(\rho\|\hat{\rho}_j) \tag{7}$$

C. Denoising AutoEncoder

To deal with the small variation of input data, Pascal Vincent *et al.* propose a trick, namely denoising autoencoder [63]. They destruct the input data manually, and verify that even partially destroyed input could yield almost the same result. This is because a good representation should be capable to capture the stable structures in form of dependencies and regularities characteristic of the unknown distribution. The learned representation is robust towards the slight disturbance of the observed.

For input $\mathbf{x}$, it is first corrupted to get a partially destroyed version $\tilde{\mathbf{x}}$ by means of stochastic mapping. The random corruption can be implemented in two fashions:

- *Binary noise:* Randomly choose a fixed number of raw data, and force their values to be 0.
- *Gaussian noise:* Generate a number of Gaussian random values, and add them with the initial data.

The corrupted data $\tilde{\mathbf{x}}$ is then mapped to the hidden layer

$$\mathbf{h} = \sigma(\mathcal{W}_v\tilde{\mathbf{x}} + \mathbf{b}_v), \tag{8}$$

from which the reconstruction can be obtained

$$\mathbf{z} = \delta(\mathcal{W}_h\mathbf{h} + \mathbf{b}_h). \tag{9}$$

A diagram of denoising autoencoder is illustrated in Fig. 2.

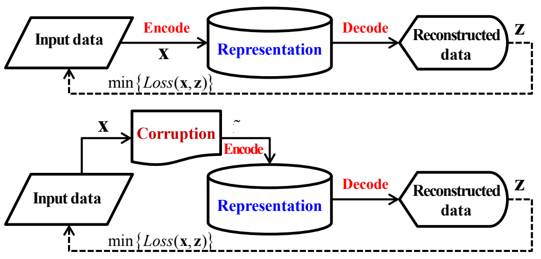


Fig. 2. The top chart displays the prototype of autoencoder, while the bottom one shows the denoising autoencoder. The input data is manually corrupted in a certain level. The corrupted data is then mapped to the hidden layer, from which the initial data can be reconstructed.

As proved in the preceding works, there is a low-dimensional manifold near which the data concentrate. The denoising autoencoder can be then viewed as a way to define and learn a manifold [64]. It aims to building a mapping from the low probability event $\tilde{\mathbf{x}}$ to the high probability event $\mathbf{x}$. This is achieved by promoting the probability $p(\mathbf{x}|\tilde{\mathbf{x}})$, similar to minimizing the loss function. The learned representation $\mathbf{h}$ can be interpreted as a coordinate system for points on the manifold. The tendency is much significant if the dimension of $\mathbf{h}$ is specified to be smaller than the dimension of $\mathbf{x}$. The process can be seen as that the learned representation captures the main change of the input. On the other hand, the latent state $\mathbf{h}$ can be viewed as the low-dimensional representation of input $\mathbf{x}$. The autoencoder can be also seen as a linear (without activation) or nonlinear (with activation) dimension reduction mapping.

D. AutoEncoder with Dropout

Dropout is a strategy popularly used in training large neural network with limited training resource [65]. It provides a way of approximately combining exponentially many different neural network architectures efficiently. "Dropout" refers to dropping out some hidden units in a neural network. Dropping an unit out means temporarily removing it from the network, along with all its all incoming and outgoing connections. The diagram flow is shown in Fig. 3.

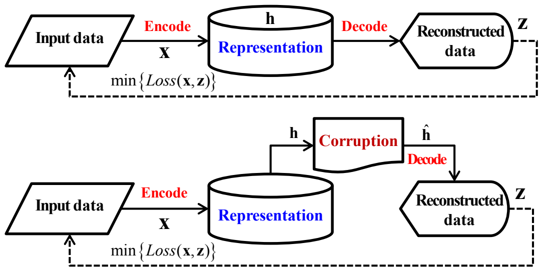


Fig. 3. The top flow shows the archetypal autoencoder, while the bottom draws autoencoder with dropout. The input data is mapped to the hidden layer, in which some neurons are deactivated. Only the remaining units are employed to reconstruct the input data.

Consider a simple neural network with three layer, the input layer $\mathbf{x}$, the hidden layer $\mathbf{h}$, and the output layer $\mathbf{z}$. The feed-forward operation with dropout is expressed as

$$\begin{aligned} \mathbf{h} &= \sigma(\mathcal{W}_v\mathbf{x} + \mathbf{b}_v) \\ \tilde{\mathbf{h}} &= \mathbf{h} \otimes Bernoulli(p) \\ \mathbf{z} &= \delta(\mathcal{W}_h\tilde{\mathbf{h}} + \mathbf{b}_h) \end{aligned} \tag{10}$$

where $\otimes$ is the element-wise product operation, and p is the proportion of hidden units to be dropped. The autoencoder with denoising and dropout is pictorially shown in Fig. 4.

E. Contractive AutoEncoder

Both denoising and dropout trick produce a stochastic fashion of neural network. All of the corrupted (or dropped) neurons are randomly determined. Differently, Salah Rifai *et al.* propose to train a deterministic fashion of neural network, contractive autoencoder [66]. They design a well chosen penalty term, the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input. The penalty could generate a localized space contraction which in turn yields the robust features on the activation layer.

To enhance the robustness of the learned representation, an intuitive idea is to penalize the sensitivity to the input. This thought is realized by the Frobenius norm of the Jacobian matrix $J_\sigma(\mathbf{x})$ of activations, $\mathbf{h} = \sigma(\mathcal{W}\mathbf{x} + \mathbf{b}_v)$. The penalty

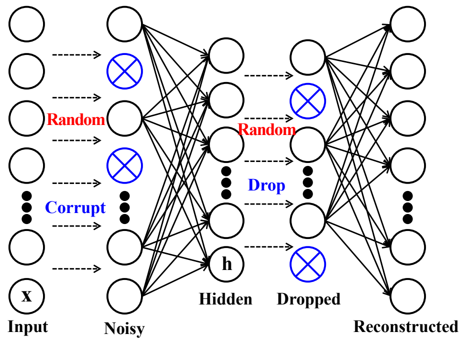


Fig. 4. The implementation flow of autoencoder with denoising and dropout. The circles in blue denote the manually corrupted (or dropped) neurons. All of them are randomly determined.

term is defined as the sum of square of all partial derivatives of the hidden nodes with regarding to input

$$\|J_\sigma(\mathbf{x})\|_F^2 = \sum_{ij}\Big(\frac{\partial h_j(\mathbf{x})}{\partial x_i}\Big)^2 \tag{11}$$

Considering the linear encoder, the penalty is degraded to the weight decay, $g(\mathcal{W})$. The Jacobian penalty encourages the mapping to the hidden layer to be contractive in the neighborhood of input. The objective function is then expressed as

$$\mathcal{J}_{contractive} = \mathcal{L}(\mathbf{x},\mathbf{z}) + \eta\|J_\sigma(\mathbf{x})\|_F^2. \tag{12}$$

Since the training examples congregate near a low-dimensional manifold, the variations present in the original data correspond to the local dimensions along the manifold. Specifically, the small variations in the input data correspond to the directions orthogonal to the manifold. The contractive autoencoder tries to make the features invariant in all directions by means of the F-norm of Jacobian matrix penalty. Simultaneously, it should be capable to reconstruct the input. It is therefore good at representing the data variations near a lower-dimensional manifold.

F. Variational AutoEncoder (VAE)

Variational autoencoder is a recently developed skill [67], [68]. It is not a precise concept of autoencoder model, but a generative model. It could generate the training samples which are not available in the gallery. Different from the archetypal autoencoder, the outputs of encoder and decoder are the samples drawn from a parameterized probability density function, as shown in Fig. 5.

From a perspective of coding theory, the hidden variables $\mathbf{h}$ can be interpreted as a latent representation or code. It refers to the recognition model $q_\phi(\mathbf{h}|\mathbf{x})$ as a probabilistic encoder. The data is encoded into a soft ellipsoidal region in the latent space, rather than a single point. Given an example $\mathbf{x}$, VAE produces a distribution (*e.g.*, Gaussian) over the possible values of the code $\mathbf{h}$, from which the datapoint $\mathbf{x}$ could be generated. The phase of encoding can be then expressed as

$$\text{Encoder}\begin{cases} \tilde{\mathbf{h}} & = \sigma(\mathcal{W}_0\mathbf{x} + \mathbf{b}_0) \\ \mu & = \mathcal{W}_1\tilde{\mathbf{h}} + \mathbf{b}_1 \\ \log\sigma^2 & = \mathcal{W}_2\tilde{\mathbf{h}} + \mathbf{b}_2 \\ \log q(\mathbf{h}|\mathbf{x}) & = \log\mathcal{N}(\mathbf{h};\mu,\sigma^2\mathbf{I}) \end{cases} \tag{13}$$

where the model parameters and the latent states are sampling form the statistical distribution $\mathbf{h} \sim q_\phi(\mathbf{h}|\mathbf{x})$.

In the phase of decoding, a similar vein can be obtained. We refer to $p_\theta(\mathbf{x}|\mathbf{h})$ as a probabilistic decoder. Given a code $\mathbf{h}$, it produces a distribution over the possible corresponding value of $\mathbf{x}$. The decoder is therefore a conditional generative model that estimates the probability of generating $\mathbf{x}$ given the latent variable $\mathbf{z}$,

$$\text{Decoder}\begin{cases} \tilde{\mathbf{h}}' & = \delta(\mathcal{W}_3\mathbf{h} + \mathbf{b}_3) \\ \mu' & = \mathcal{W}_4\tilde{\mathbf{h}}' + \mathbf{b}_4 \\ \log\sigma_1^2 & = \mathcal{W}_5\tilde{\mathbf{h}}' + \mathbf{b}_5 \\ \log p(\mathbf{x}|\mathbf{z}) & = \log\mathcal{N}(\mathbf{x};\mu',\sigma_1^2\mathbf{I}) \end{cases} \tag{14}$$

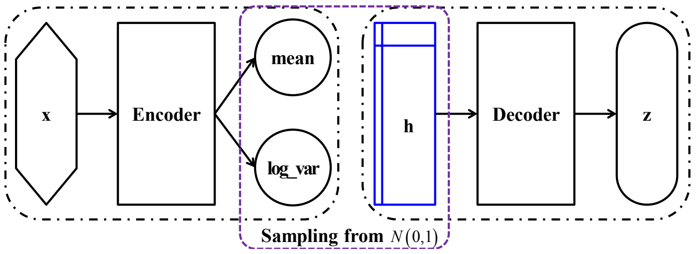


Fig. 5. Illustration of variational autoencoder. The model parameters and the latent states are sampling from a parameterized statistical distribution.

Variational autoencoder aims to approximating $p_\theta(\mathbf{x}|\mathbf{h})$ by the given distribution $q_\phi(\mathbf{h}|\mathbf{x})$. It is needed to balance the reconstruction accuracy and the goodness-of-fit of Gaussian distribution. The objective can be achieved by the neural network itself. The loss function is therefore composed of these items. The reconstruction accuracy can be measured by the mean square error, while the difference between the latent variables and the Gaussian distribute is typically quantified by the KL divergence,

$$\mathcal{J}_{VAE} = D_{KL}(q_\phi(\mathbf{h}|\mathbf{x})||p_\theta(\mathbf{x}|\mathbf{h})) + \text{MSE}(\mathbf{x},\mathbf{z}) \tag{15}$$

where ϕ and θ are the variational parameters and the generative parameters.

For the high-dimensional data, the similar examples may distribute in a high-dimensional manifold. The task of representation learning is actually to predict that manifold explicitly or implicitly. The observations can be then reconstructed from the low-dimensional latent variants. The latent variants could approximate the observations. Moreover, the similarity is varied smoothly.

G. Convolutional AutoEncoder

The fully connected autoencoder ignores the spatial structure of image. To handle the problem, J. Masci *et al.* present convolutional autoencoder [69]. They introduce some redundancy in the model parameters, forcing the learned representation to be global, spanning the entire visual field. Since the weights and bias are shared among all locations of input, the spatial locality can be preserved. Given the single channel image $\mathbf{x}$, the latent representation of the k-th feature map is given by

$$\mathbf{h}^k = \sigma(\mathbf{x} * \mathcal{W}^k + \mathbf{b}_v). \tag{16}$$

The bias $\mathbf{b}_v$ is broadcasted to the whole map. Since one bias per pixel would introduce too many degrees of freedom, one bias per latent map is employed. Then each kernel filter could specialize on features of the whole input.

Following the road-map of convolutional neural network, a max-pooling layer is connected with the convolutional layer. The sparsity over the representation can be then introduced. It erases the non-maximal values in non overlapping subregions. It could generate more broadly applicable feature, and avoid the trivial solutions such as having only one weight "on". The reconstruction is obtained using

$$\mathbf{z} = \delta(\sum_k \mathbf{h}^k * \tilde{\mathcal{W}}^k + \mathbf{b}_h) \tag{17}$$

where $\tilde{\mathcal{W}}$ denotes the flip of $\mathcal{W}$ over both dimensions of the weights. In the phase of reconstruction, the sparse latent coding decreases the average number of filters contributing to the decoding of each pixel, forcing filters to be more general. Consequently, the ℓ_1- or ℓ_2-regularization over the hidden layer is no need. The cost function is

$$\mathcal{J}_{conv} = \mathcal{L}(\mathbf{x}, \mathbf{z}) = \sum_i \{\|\mathbf{x}_i - \mathbf{z}_i\|_2^2\}. \tag{18}$$

H. Illustrative Examples

The generic autoencoder model is composed of a encoder and a decoder. The learned feature is the encoded coefficients. For the nonlinearly separable dataset, a network with much more hidden neurons is usually built (overcomplete representation). For the high-dimensional examples, the dimension can be reduced by training a network with less hidden neurons. Autoencoder can be therefore viewed as a dimension reduction trick. The network with nonlinear activation plays a similar role to the nonlinear mappings, such as locally linear embedding, Laplacian eigenmaps, while the network with linear activation resembles the linear skills, such as principal component analysis, linear discriminant analysis. In this subsection, we demonstrate some illustrative examples, by which PCA, autoencoder with linear activation, and with nonlinear activation are compared with.

To visually compare autoencoder and PCA, we perform a group of experiments. The images are of 96×96 pixels in size, and from three different classes. The details can be found in Section VI. We produce the 2-D representations for the bag of images by a trained autoencoder and PCA, and display the learned features in Fig. 6. As can be seen, the representations learned by autoencoder are scattered more separably, while the ones produced by PCA are overlapped. It is therefore much easier to differentiate these samples by the representations learned by autoencoder.

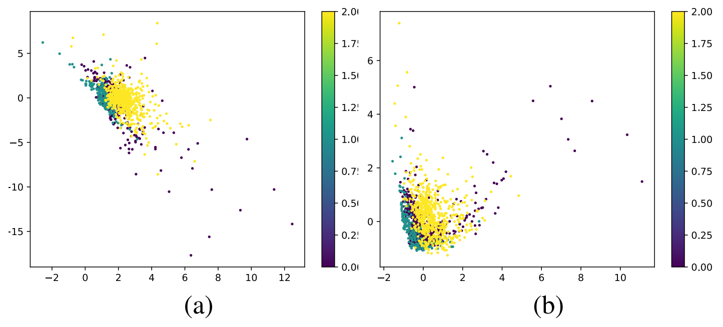


Fig. 6. The scattering map of the learned 2-D feature by (a) autoencoder and (b) principal component analysis.

We then evaluate the recognition performance. The learned representations are connected with a softmax layer to implement target classification. The experimental results are displayed in Fig. 7. As can be seen, the tendency is obvious. With the dimension of learned representation increased, the recognition accuracy is is changed. The 256-D representation always produces the best performance. On the other hand, the recognition accuracy obtained using PCA is consistently poorer than autoencoder, whether the linear activation or the ReLU activation are utilized. Simultaneously, we found the performance obtained using autoencoder with ReLU activation is always much better than the linear activation for all six dimensions of representation.

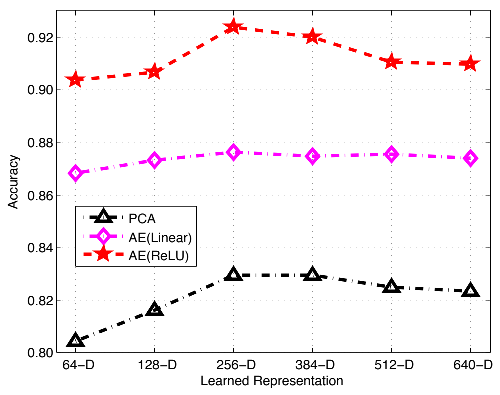


Fig. 7. The recognition performance obtained using autoencoder and PCA.

III. The Deep Model

A deep architecture is formed by the composition of multiple levels of representations. Likewise, the basic autoencoder can be concatenated to build the deep model, stacked autoencoder. It is composed of a visible layer, several hidden layers, and an output layer. The output of previous layer is wired to the input of the successive layer, as illustrated in Fig. 8. Stacked autoencoder inherits the benefits of deep network of greater expressive power. An autoencoder model aims to learning a good representation of the input. The first-hidden layer of stacked autoencoder tends to learn the first-order feature from the input data. Likely, the successive-hidden layer typically generates the high-level feature corresponding to pattern in the appearance of previous-level ones. It therefore captures a useful "hierarchical grouping" or "part-whole decomposition" of the input [70].

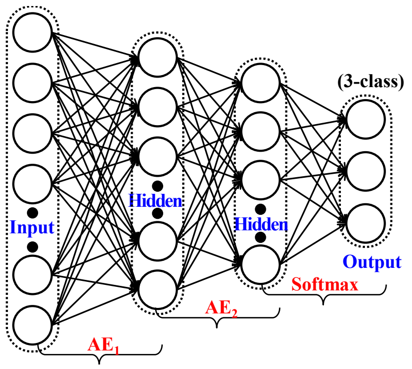


Fig. 8. Stacked autoencoder with two hidden layers. The successive layers are connected with the preceding layer by the model parameters. To implement classification, the last-hidden layer is usually connected with a softmax layer, whose output is the probability of the class taking each possible values.

For stacked autoencoder with N hidden layers, the weight and the bias of the k-th layer are denoted by $\mathcal{W}_v^{(k)}$, $\mathcal{W}_h^{(k)}$, $\mathbf{b}_v^{(k)}$, $\mathbf{b}_h^{(k)}$. The encoding is implemented layer by layer in a forward order,

$$\begin{aligned}\mathbf{h}^{(k)} &= \sigma(\mathbf{z}^{(k)})\\ \mathbf{z}^{(k+1)} &= \mathcal{W}_v^{(k)}\mathbf{h}^{(k)} + \mathbf{b}_v^{(k)}\end{aligned} \tag{19}$$

where $\mathbf{z}^{(k)} = \mathbf{x}^{(k-1)}$ is the output of previous layer. Similarly, the decoding is run layer by layer in a reverse order,

$$\begin{aligned}\mathbf{h}^{(n+k)} &= \delta(\mathbf{z}^{(n+k)})\\ \mathbf{z}^{(n+k+1)} &= \mathcal{W}_h^{(n-k)}\mathbf{h}^{(n+k)} + \mathbf{b}_h^{(n-k)}\end{aligned} \tag{20}$$

The information interested is contained in the transition period, *i.e.*, the activation of the deepest layer of hidden units. The model parameters are estimated by a layer-wise greedy strategy [61]. Specifically, the first-hidden layer is trained by the input data, resulting in the parameters $\mathcal{W}_v^{(1)}$, $\mathcal{W}_h^{(1)}$, $\mathbf{b}_v^{(1)}$, $\mathbf{b}_h^{(1)}$. The activations of the hidden neurons are then used to train the second-hidden layer, producing the parameters $\mathcal{W}_v^{(2)}$, $\mathcal{W}_h^{(2)}$, $\mathbf{b}_v^{(2)}$, $\mathbf{b}_h^{(2)}$. The procedure is transfered to the final-hidden layer. Each layer is trained individually, *i.e.*, those parameters of the network which is not involved are frozen. To boost the performance, fine-tuning via back-propagation is proposed. It improves the results by tuning the parameters of all layers. The implementation flow is summarized as follows:

- Compute the activations of each hidden layers.
- For certain layer n_l, compute the partial derivative

$$\delta^{(n_l)} = -(\nabla_{\mathbf{h}^{(n_l)}}\mathcal{J}) \cdot \sigma'(\mathbf{z}^{(n_l)})$$

- For $l = n_l - 1, n_l - 2, \ldots, 2$, set

$$\delta^{(l)} = ((\mathcal{W}^{(l)})^T\delta^{(l+1)}) \cdot \sigma'(\mathbf{z}^{(l)})$$

- Produce the desired partial derivatives:

$$\begin{aligned}\nabla_{\mathcal{W}^{(l)}}\mathcal{J}(\mathcal{W}, \mathbf{b}_h, \mathbf{b}_v) &= \delta^{(l+1)}(\mathbf{h}^{(l)})^T\\ \nabla_{\mathbf{b}^{(l)}}\mathcal{J}(\mathcal{W}, \mathbf{b}_h, \mathbf{b}_v) &= \delta^{(l+1)}\end{aligned}$$

Due to the representation power, stacked autoencoder has been widely used in the preceding works. Y. Chen *et al.* build a deep learning architecture by stacking autoencoder, with which the useful high-level features can be learned from the hyperspectral data [28]. J. Geng *et al.* refine the hand-engineered features by a contractive neural network [21]. S. Hao *et al.* encode the spectral values of the input pixel by stacked denoising autoencoder, and handle the corresponding image patch by a deep convolutional neural network [71]. X. Sun *et al.* learn a discriminative deep feature by a trained stacked autoencoder, where the constraint of the label consistency on a neighborhood region has been introduced [16]. X. Zhang *et al.* consider the spatial and spectral information jointly by a recursive autoencoder network [72]. A weighting scheme is developed to enhance the representation power. They determine the weights by the spectral similarity between the neighboring pixels and the investigated pixel. D. Zhang *et al.* build a stacked denoising autoencoder to learn the saliency prior knowledge from auxiliary annotated data sets and then transfer the learned knowledge to estimate the intra-saliency for each image in co-saliency data sets [73]. L. Cao *et al.* prove that the sparsity regularization and denoising mechanism seem to be mandatory for constructing interpretable feature representations [74]. J. Feng *et al.* propose a stacked marginal discriminative autoencoder model to handle the limited training samples [75]. The marginal samples obtained by k nearest neighbors between different classes are used to fine-tune the defined network. S. Paul and D. Kumar present a mutual information based segmented stacked autoencoder model to reduce the computational complexity [76]. A non-parametric dependency measure based spectral segmentation is defined to consider both linear and nonlinear inter-band dependency for spectral segmentation of the hyperspectral bands. X. Han *et al.* present an unsupervised convolutional sparse auto-encoder model to represent the spatial-spectral features around the central pixel within a spatial neighborhood window [77].

IV. Classification

Though autoencoder is an unsupervised learning skill, it can be appended by a softmax layer to achieve pattern classification. This section first provides a simple review of linear regression and its special form, logistic regression, followed by multi-class logistic regression (softmax classifier).

A. Linear Regression

Linear regression models the relationship between a scalar dependent variable y (response) and a set of explanatory variables $x_1, x_2, \ldots, x_m$,

$$y = h_\theta(\mathbf{x}) = \theta_0 + \theta_1 x_1 + \theta_2 \mathbf{x}_2 + \cdots + \theta_m x_m = \theta^T\mathbf{x} \tag{21}$$

where $\mathbf{x} = [1, x_1, x_2, \ldots, x_m]$ is an observed instance. $\theta = [\theta_0, \theta_1, \ldots, \theta_m]$ is the regression coefficients. Given a set of labeled training samples $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_M, y_M)\}$, the cost function is defined as

$$\mathcal{J}(\theta; (\mathbf{x}_i, y_i)_{i=1}^M) = \frac{1}{2M} \sum_{i=1}^{M} \Big(y_i - h_\theta(\mathbf{x}_i)\Big)^2. \tag{22}$$

The regression parameters θ is obtained by optimizing objective $\hat{\theta} = \arg\min_\theta \mathcal{J}(\theta; (\mathbf{x}_i, y_i)_{i=1}^M)$. It is solved by the gradient descent method,

$$\theta_j^{(new)} = \theta_j^{(old)} - \alpha \frac{\partial}{\partial \theta_j} \mathcal{J}(\theta) \tag{23}$$

where α is the learning rate. It is important to the solution of the regression parameters θ.

B. Logistic Regression

Logistic regression is a special form of linear regression where the dependent variable is categorical (biological). It typically produces the binary classification, $y \in \{0, 1\}$. The hypothesis concerning the label of sample $\mathbf{x}$ takes the form,

$$h_\theta(\mathbf{x}) = \sigma(\theta^T \mathbf{x}) = \frac{1}{1 + \exp(-\theta^T \mathbf{x})}. \tag{24}$$

It is obtained by a logarithm operation on the ratio of positive and negative hypotheses,

$$\begin{aligned} \log it(\mathbf{x}) &= \ln\Big(\frac{\mathcal{P}(y=1|\mathbf{x})}{\mathcal{P}(y=0|\mathbf{x})}\Big) = \ln\Big(\frac{\mathcal{P}(y=1|\mathbf{x})}{1-\mathcal{P}(y=1|\mathbf{x})}\Big) \\ &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_m x_m = \theta^T \mathbf{x} \end{aligned} \tag{25}$$

The cost function can be then expressed as

$$\mathcal{J}\big(\theta; (\mathbf{x}_i, y_i)\big) = -\frac{1}{m}\Big[\sum_{i=1}^{M} y_i \log h_\theta(\mathbf{x}_i) + (1-y_i)\log(1-h_\theta(\mathbf{x}_i))\Big]$$

C. Multi-class Logistic Regression

Multi-class logistic regression is a generalization of logistic regression to the case of multiple classes. The labels of logistic regression are binary, while softmax classifier allows multiple classes, $y \in \{1, 2, \ldots, K\}$. For a query sample $\mathbf{x} \in \mathbb{R}^d$, the output of softmax classifier is the probability of the class label taking one each of the K different possible values,

$$h_\theta(\mathbf{x}) = \begin{bmatrix} \mathcal{P}(y=1|\mathbf{x};\theta) \\ \mathcal{P}(y=2|\mathbf{x};\theta) \\ \vdots \\ \mathcal{P}(y=K|\mathbf{x};\theta) \end{bmatrix} = \frac{1}{\sum_j \exp(\theta_j^T \mathbf{x})} \begin{bmatrix} \exp(\theta_1^T \mathbf{x}) \\ \exp(\theta_2^T \mathbf{x}) \\ \vdots \\ \exp(\theta_K^T \mathbf{x}) \end{bmatrix}$$

where $\theta = [\theta_1, \theta_2, \ldots, \theta_K]$ are the model parameters. The probability function is

$$\mathcal{P}(y|\mathbf{x};\theta) = \prod_{j=1}^{K} \Big(\frac{\exp(\theta_j^T \mathbf{x})}{\sum_{i=1}^{K} \exp(\theta_i^T \mathbf{x})}\Big)^{I\{y=j\}} \tag{26}$$

where $I\{\cdot\}$ is a indicator function. The likelihood function is

$$\mathcal{LK}(\theta; (\mathbf{x}_i, y_i)) = \prod_{i=1}^{N} \prod_{j=1}^{K} \Big(\frac{\exp(\theta_j^T \mathbf{x}_i)}{\sum_{k=1}^{K} \exp(\theta_k^T \mathbf{x}_i)}\Big)^{I(y_i=j)} \tag{27}$$

Taking the logarithm operation, it can be then written as

$$\log \mathcal{LK}(\theta) = \sum_{i=1}^{N} \sum_{j=1}^{K} I\{y_i = j\} \log\Big(\frac{\exp(\theta_j^T \mathbf{x}_i)}{\sum_{k=1}^{K} \exp(\theta_k^T \mathbf{x}_i)}\Big)$$

The cost function is

$$\mathcal{J}(\theta) = -\Big[\sum_{i=1}^{N} \sum_{j=1}^{K} I\{y_i = k\} \log \frac{\exp(\theta_k^T \mathbf{x}_i)}{\sum_{j=1}^{K} \exp(\theta_j^T \mathbf{x}_i)}\Big] \tag{28}$$

V. Application to Target Recognition

With the development of integrated circuit technology, huge high-resolution images are collected by various kinds of sensors. It is urgent to achieve image interpretation automatically. Target recognition[1] in SAR images is a typical research topic of image interpretation. Though studied widely, it is still an open problem due to the complicated battlefield environment and the mutable imaging conditions. The conventional methods used in the optical sensor images is not effective to radar image. The researchers gradually recommend learning the effective representation from the data itself via deep neural network [78]–[82]. In this section, we provide a simple review of the preceding works on the hand-engineered features. The achievement of target recognition in SAR image via an autoencoder neural network is then presented.

A. The Previous Works

The performance of radar target recognition is mainly dependent on two key issues, how to design an effective feature from a radar image, and how to implement classification with the defined feature. We give a simple summary of the hand-engineered features developed for SAR images, followed by the typical classification methods popularly used.

1) Handcrafted feature: In SAR imaging process, there are multiple scattering mechanisms contributing to the backscattered signal [83]. The interaction of electromagnetic wave and object includes the direct backscatter, the single-direction double bounce, and the return-direct multi-bounce. Due to the complicated scattering mechanisms, it is much difficult to extract feature from radar image. The conventional features developed for the optical sensor images are not effective any more. Therefore, some delicate features specific to SAR image have been presented.

- *Raw intensity.* In the early works, the raw intensity values, or the enhanced image are directly employed to produce a feature [84], [85]. The defined feature is then input to a trained classifier. The performance is therefore mainly dependent on a discriminative classifier, rather than the feature itself. Though easily defined, the recognition performance is limited, especially in the non-literal settings.

[1]In this paper, the term "target" refers to the ground, military vehicles, such as main-battle tank, armored personnel carrier, car. We aim to differentiating these military vehicles by the radar images. We use the term "target recognition" to classify a scenery of radar image as one of several classes.

- *Geometry feature.* This kind of features are obtained by separating the target and the radar shadow, from which some region (or shape) descriptors can be obtained [86], [87], such as the shape context descriptor of radar shadow and target, various kinds of statistical moment. This family of features are usually needed the fine segmentation of target, an open problem in radar images.
- *Projection coefficients.* Some researchers propose to project the initial image into a designed linear subspace, and define a feature by the transformed coefficients. The representative trick includes principal component analysis, locality preserving projection, non-negative matrix factorization [53]. There is considerable ambiguity about what these features actually mean.
- *Filter banks.* To deal with the extended conditions such as translation, distortion, some researchers propose to define the feature by a family of filter banks. The representative includes Fourier transform, wavelet, and Gabor filters [88], [89]. These methods are usually effective to the certain dataset or task, and hence difficult to be extended to other generalized applications.
- *Scattering center models.* This family of feature refers to the received returns by the interactions between the incident radio wave and the physical structure of object. The thought has been led by L. Potter and R. Moses [90]. They present a framework for feature extraction predicated on parametric models for the radar returns. The models are motivated by the scattering behavior predicted by the geometrical theory of diffraction. Then some improved methods have been developed [91]–[95].

2) *Typical Classifier:* The generated feature is input to a trained classifier to predict the class membership. The methods popularly used include kNN classifier, kernel-based classifier, Bayesian inference, and sparse representation. *KNN classifier* predicts the identity by measuring the similarity between the probe and the training. The key is to define an appropriate metric, such as mean square error [96], KL-divergence [97], [98], Wishart distance [99]. *Kernel-based classifier* projects the initial data into an abstract feature space, whose dimensionality can be much high or even infinite. The class separability can be then enhanced in the feature space. One of the most representative method is SVM with the linear kernel, the Sigmoid kernel, the Gaussian radial function, or the polynomial kernel. Q. Zhao and J. Principe present an application of SVMs for SAR automatic target recognition [85]. They demonstrate the advantage of SVM compared with the conventional classifiers in both closed and open sets. Y. Sun *et al.* propose to classify the SAR images by using the adaptive boosting algorithm with the radial basis function network as the base learner [50]. R. Kemker and C. Kanan implement the hyperspectral image classification by feeding the learned representation into a SVM classifier [43]. *Bayesian inference.* This family of classifiers predict the identification with the Bayesian theory, such as maximum a posterior probability [51], maximum likelihood [100], generalized likelihood ratio [101]. *Sparse representation-based classifier* is a recently developed method. It considers the classification problem as the multiple linear regression models, and address the problem by the theory of sparse representation [84].

B. Target Recognition via AutoEncoder

Though great efforts have been pursued, feature generation from radar image is still much challenging due to the unique imaging mechanism. The handcrafted and predefined features are usually effective to the certain dataset or task, and yet could not be transferred to the other dataset. In the preceding studies, the researchers mainly focus on how to design a delicate feature, by which target recognition can be achieved. Seldom works are devoted to learning the latent representation from the original data itself. To break this tendency and improve the performance, a good choice is to rely on the advanced learning skills, such as deep neural network [32], [102], by which an effective representation can be learned.

This paper considers target classification in radar images. To avoid the multiple complicated procedures in the conventional methods, this paper proposes to learning an effective representation by an unsupervised learning skill, autoencoder neural network [103]. The learned representation can be then input to a third-party trained classifier, or a softmax layer to predict the class membership of the query. To form an autoencoder model, we reshape the radar image to be a single array, whose entries are the nodes of visible layer. The visible layer is then cast into the hidden layer by the model parameters (weights and bias) and the activation function. Each visible node is connected with all of the hidden neurons, some of which are deactivated or dropped. The network is allowed only to transmit forward, and hence also called feed-forward neural network. The deep neural network architecture is built by concatenating the basic autoencoder models hierarchically. To estimate the model parameters, a generic method is to optimize a loss function composing of the reconstruction error and some regularization terms. The deviation is allowed only to transmit from the end to the beginning, *i.e.*, back-propagation. The latent representation is obtained by training the network layer by layer [61]. To implement target classification, the learned representation is then input to a softmax layer, or a third-party classifier, by which the identification can be predicted. The whole procedure is pictorially shown in Fig. 9. The implementation flow is summarized as twofold,

- Generate the representation by training an autoencoder neural network.
- Predict the identification via a softmax (or third-party) classifier.

VI. Experiments and Discussions

This section verifies the performance of autoencoder and its variants from a comparative perspective. The experiments are pursued on MSTAR database, a collection consisting of X-band SAR images in one-foot resolution. Images are taken at several different operating conditions, including depression angle, aspect view, configuration, occlusion and articulation. The depression angle refers to the angle between the line of sight (from the radar to the target) and the horizontal plane at the radar. For each target, images are captured at different

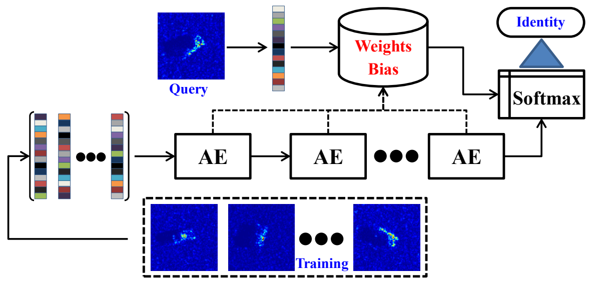


Fig. 9. Target recognition in SAR images via the autoencoder network.

depression angles over a full $0 \sim 359°$ range of aspect view. The initial images are of 128×128 pixels around in size, and are standardized by cropping the center patches.

We organize two kinds of experiment plans, standard verification and extended evaluation. The first devotes to target recognition under the standard settings, with which the impacts of related factors can be studied. The second aims to the non-literal conditions, from which the quantitative comparison with state-of-the-art algorithms is performed. The network models are realized by Keras and Theano library. All of the experiments are pursued on Python 3.6 of Win 7. The source codes are publicly released in the website https://ganggangdong.github.io/homepage.

A. Fundamental Validation

We first consider the fundamental evaluations. The effect of related factors are studied by changing some configurations, including the size of visible layer, the size of hidden layer, activation, loss function, optimizer, the variants of autoencoder, and the depth of network. Images of BMP2, BTR70, and T72 are used. The details on training and testing are shown in TABLE II.

TABLE II. NUMBER OF ASPECT VIEW FOR BMP2, BTR70, AND T72.

Target	SeriesNo.	Training	Testing
BMP2	SN_9563	233	195
	SN_9566	—	196
	SN_c21	—	196
BTR70	SN_c71	233	196
BMP2	SN_132	232	196
	SN_812	—	195
	SN_s7	—	191
Total		698	1365

1) The Size of Visible Layer: The size of input layer refers to the dimension of input data. It influences the effectiveness of the learned representation in some degree. To study the effect, we pursue a set of experiments. The initial images are of 128×128 pixels around in size. To produce different size of input data, we standardize the images by cropping the center 48×48, 56×56, 64×64, 72×72, 80×80, 88×88, 96×96, 104×104, and 112×112 patches, corresponding to 2304-, 3136-, 4096-, 5184-, 6400-, 7744-, 9216-, 10816-, and 12544-node visible layer. Fig. 10 displays a set of the cropped images. The cropped patches are used to train an autoencoder. The number of hidden unit is changed in a range [400, 1200].

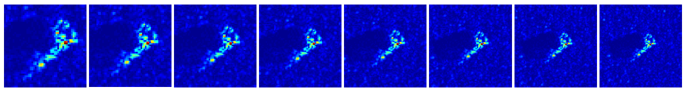


Fig. 10. The center 48×48, 56×56, 64×64, 72×72, 80×80, 88×88, 96×96, and 104×104 pixels patches generated from the original image.

The results are reported as some statistics (the minimum, median, and minimum, the 25th and 75th percentiles) of different settings, as box-plotted in Fig. 11. As can be seen, the input data does play an important role to the effectiveness of learned feature. The recognition accuracies are gradually increased when the size of visible layer increased from 2304-D to 12544-D. The lowest accuracy is obtained by 48×48-pixel patches, in which part of radar shadow and most of the background have been excluded. Contrarily, the best rate is produced by 96×96- and 112×112-pixel patches. The most stable performance is obtained using 96×96-pixel patches, while the least stable performance is generated by 56×56-pixel input. The recognition performance is much poor when the size of visible layer is too small. The results demonstrate that 96×96-pixel patches is suitable for our task of target recognition. This set of experiments also confirm that both radar shadow and background have some discriminative power. The performance can be improved by considering target, radar shadow, and background jointly.

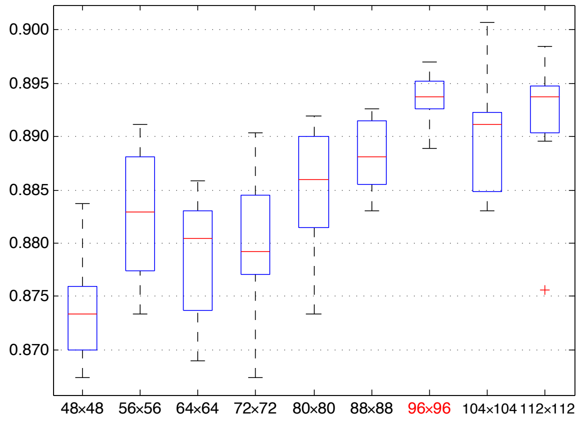


Fig. 11. The recognition accuracy over different size of visible layer.

2) The Size of Hidden Layer: Autoencoder neural network is an unsupervised learning trick. The learned representation is actually the encoded coefficients (hidden neurons). The dimension of learned feature is therefore determined by the size of hidden layer. To study which number of hidden unit is appropriate to the task of target recognition, we pursue a group of experiments. The 88×88- and 96×96-pixel patches are employed as the input, while the hidden layer is changed from 300 to 2000.

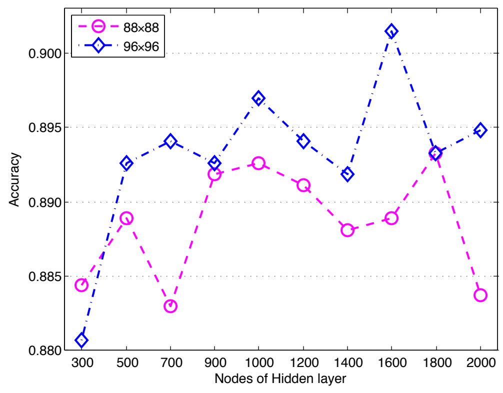


Fig. 12. The recognition accuracy across the number of hidden unit.

The results are reported as the overall recognition rate, as shown in Fig. 12. We can see the recognition rates are irregularly varied with the size of hidden layer increased. Two kinds of input data demonstrate similar trend. For 96×96-pixel input, the lowest rate is produced by 300-hidden-unit network. Correspondingly, 300-, 700-, and 2000-hidden-unit networks produce the poor performance for 88×88-pixel input. The recognition accuracies are not proportional to the size of hidden layer. The best recognition rate is obtained by 1600-hidden-node network with 96×96 pixels input. The results prove that it is important to tune an appropriate size of hidden layer according to our mission at hand. Neither small nor large size of hidden layer could learn an effective representation.

3) Activation: The activation function is a nonlinear mapping. It projects the encodings (or decodings) into a certain range [0,1], and hence converts a linear encoder (or decoder) to be a nonlinear one. The typical activation includes sigmoid function, hyperbolic tangent, and rectify linear unit.

- Sigmoid: $\sigma(x) = \frac{1}{1+\exp(-x)}$
- Hyperbolic tangent (tanh): $\sigma(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$
- Rectify linear unit (ReLU): $\sigma(x) = \max(x, 0)$

To study the effect of activation, a set of experiments are performed. We set the size of visible layer as 96×96, and vary the units of hidden layer from 400 to 1600. Fig. 13 draws the recognition accuracy across the size of hidden layer. Three activation functions are compared with.

The results demonstrated in Fig. 13 (a) and (b) are slightly different. When the cross entropy function is used to measure the deviation, ReLU function produces the best performance. The accuracies are gradually decreased with the number of hidden unit increased. The sigmoid function generates a much poor performance. On the contrary, hyperbolic tangent function provides great advantage when mean square error function is employed to quantify the deviation. The improvement is significant in terms of the recognition accuracy. The performance obtained using sigmoid function is varied sharply, even increasing from 0.8200 to 0.8852. Hyperbolic tangent function always produces the most stable recognition performance. From this round of experiments, we could conclude that the hyperbolic tangent function is better to be conjunction with the loss of mean square error.

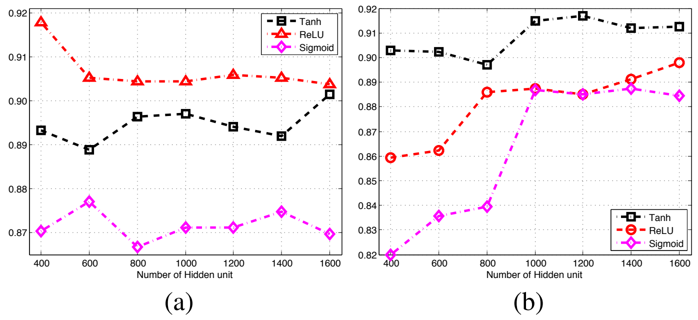


Fig. 13. The comparison of activation function with the loss function of (a) cross entropy and (b) mean square error.

4) The Loss Function: The loss function is used to quantify the deviation of the reconstructed sample from the initial input data. An ideal autoencoder neural network aims to building an identity function, with which the input data can be perfectly reconstructed. The deviation of a reconstruction amount from the original value is expected to be as minimal as possibly. The metrics popularly used are mean square error (MSE) and cross entropy.

- MSE: $\mathcal{L}(\mathbf{x}, \mathbf{z}) = \frac{1}{2d_v}\sum_{i=1}^{d_v}\left\{(x_i - z_i)^2\right\}$
- Cross Entropy:
  $\mathcal{L}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{d_v}\left\{x_i \cdot \log(z_i) + (1-x_i)\cdot\log(1-z_i)\right\}$

To study the impact of loss function, a set of experiments are pursued. We set the size of visible layer as 96×96, and change the number of hidden unit from 400 to 1600. The experimental results are shown in Fig. 14, where two loss functions are compared with.

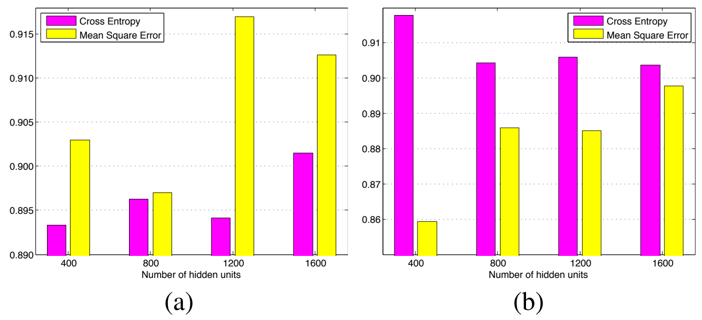


Fig. 14. The comparison of loss function with the activation of (a) hyperbolic tangent and (b) rectify linear unit.

As can be seen, Fig. 14 (a) and (b) demonstrate the totally different tendency. The recognition accuracy obtained using MSE function is much better than cross entropy function when the hyperbolic tangent is used to achieve activation. The bigger the hidden layer size, the better the performance, when MSE function is employed to measure the deviation. The 1200-hidden-neuron network produces the best performance. On the contrary, the performance produced by cross entropy function is much better than MSE function when ReLU function is employed to implement activation. The recognition accuracy is gradually decreased with the number of hidden unit increased. The 400-hidden-unit network generates the best performance. The results prove that large-scale hidden layer network is suitable to be conjunction with MSE function, while small-scale hidden layer network is appropriate to be configured with the cross entropy function.

5) Optimizer: Have determined the architecture of neural network, the next problem is to solve the model parameters (the weights, the bias) by an optimization scheme. The most popularly used method is the gradient descent optimization algorithm. In the community of deep learning, many variants of gradient descent have been presented. The representative include stochastic gradient descent (SGD), RMSprop, adagrad, adadelta, adaptive moment estimation (adam), adamax, and Nesterov-accelerated adaptive moment estimation (Nadam). The comprehensive review on optimization can be found in the preceding work [104]. To verify the performance of these algorithms, we pursue a set of experiments. The results are reported as some statistics (minimum, median, maximum, the 25th and 75th percentiles) of 10 sample runs, as drawn in Fig. 15[2].

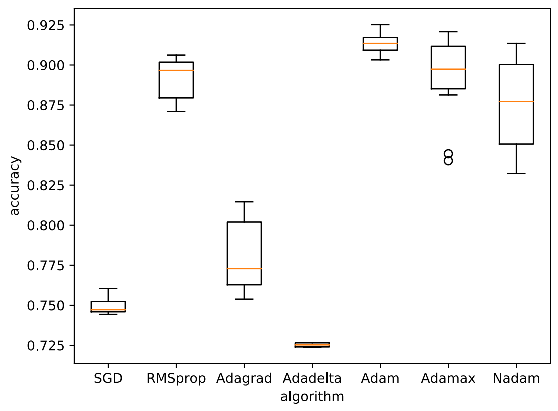


Fig. 15. The recognition performance obtained using different optimizers.

As can be seen from Fig. 15, the family of optimization algorithms demonstrate sharply different recognition performance. The best performance is obtained by 'adam' optimizer, as also widely studied in the preceding works. The poorest performance is generated by 'adadelta' algorithm. Nearly 20-percent drop of recognition accuracy has been produced. The result is conform to the preceding study [104]. The recognition accuracy produced by 'SGD', 'adadelta', 'adam' optimizer are much more robust than the remaining algorithms. The fluctuation of recognition rate produced by 'nadam' algorithm is more drastic than the remaining. It can be therefore concluded that the 'adam' algorithm is a good and general choice to solve the model parameters.

6) Sparse AutoEncoder: Sparse autoencoder introduces a sparse constraint, KL divergence between the average activation hidden units and the desired value, with which the inner structure of input can be exploited. To validate the performance of sparse constraint, we conduct a set of experiments. The input layer is set as 96×96-pixel patches, while the number of hidden units are specified as 800. We manually change the desired activation value from 0.01 to 0.8. Two different structures of network are configured. The first scheme achieves the nonlinear mapping with hyperbolic tangent function, and measures the reconstruction error by cross entropy function. The second scheme implements the activation by the sigmoid function, and quantifies the loss by MSE function. The experimental results are drawn in Fig. 16.

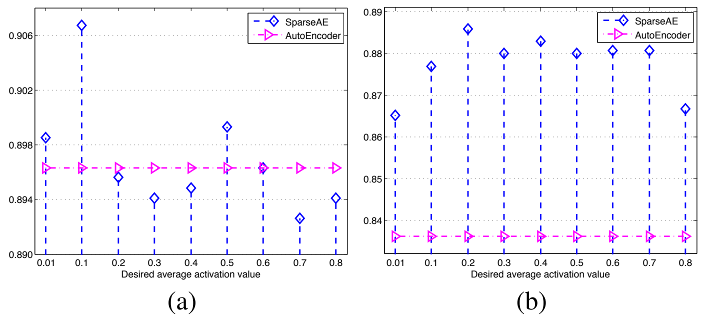


Fig. 16. The comparison of autoencoder with and without sparse constraint. (a) hyperbolic tangent activation and cross entropy loss, (b) sigmoid activation and mean square error loss.

As can be seen from Fig. 16 (a) and (b), two totally different kinds of tendency are displayed. Only sparse autoencoder with the desired sparsity value of 0.01, 0.1 and 0.5 outperform the prototype of autoencoder when the hyperbolic tangent activation and cross entropy loss function are configured. On the contrary, the recognition accuracy obtained using sparse autoencoder with all desired sparsity values (from 0.01 to 0.8) are better than autoencoder when the sigmoid activation is conjunction with MSE loss function. The results prove that sparse autoencoder is effective to the network configuration of sigmoid activation and MSE loss. The desired sparsity value should be tuned according to the task at hand. An inappropriate sparsity constraint may degrade the recognition performance.

7) Denoising AutoEncoder: Denoising autoencoder is developed to deal with the slight disturbance of observed input. It trains the network by the manually corrupted data. To verify the performance, a group of experiments are conducted. The size of visible layer is set as 96×96, while 600- and 1200-hidden-neuron networks are built. We enhance the corruption level gradually from 0.01 to 0.5. The recognition accuracies across the level of corruption are drawn in Fig 17, where denoising autoencoder and the prototype of autoencoder are compared with.

As can be seen from Fig. 17, two configurations of network produce the similar trend. Significant improvement has been obtained when the input data are manually destructed. The best recognition rate is produced by denoising autoencoder with 600-hidden-neuron, 3.04% better than the prototype of autoencoder. Similarly, the best recognition rate for denoising autoencoder is 0.9237 when 1200-hidden-unit network is built,

[2]This round of experiments are performed by Keras library with tensorflow backend. A 1024-hidden-neuron network is built, followed by a softmax layer on the top architecture.

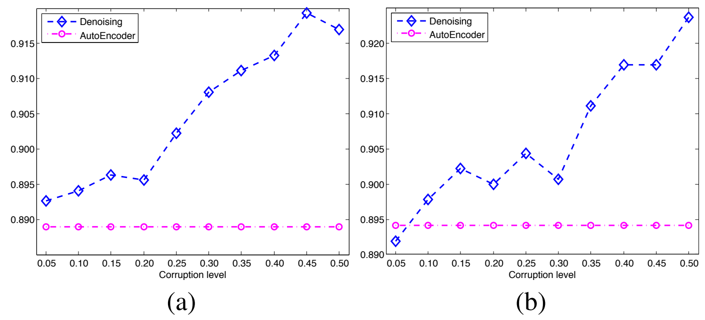


Fig. 17. The comparison of autoencoder with and without denoising trick. (a) 600-hidden-node network, (b) 1200-hidden-node network.

2.93% better than the prototype of autoencoder. Only the accuracy of denoising with 0.05-level is slightly lower than autoencoder. The similar conclusion can be concluded in the preceding study [63]. The results prove that the effectiveness of learned representation can be promoted by manually corrupting the input data.

8) Dropout: Denoising trick corrupts the input data manually, by which the slight perturbance can be allowed. Differently, dropout skill abandons the hidden neurons randomly. It aims to handling the small training samples. To verify the performance, we pursue a group of experiments. The visible layer is set as 96×96, while 600- and 1200-hidden-neuron network are configured. We promote the level of dropout gradually from 0.05 to 0.4. The results are reported in Fig 18, where the recognition accuracy over the level of dropout is plotted.

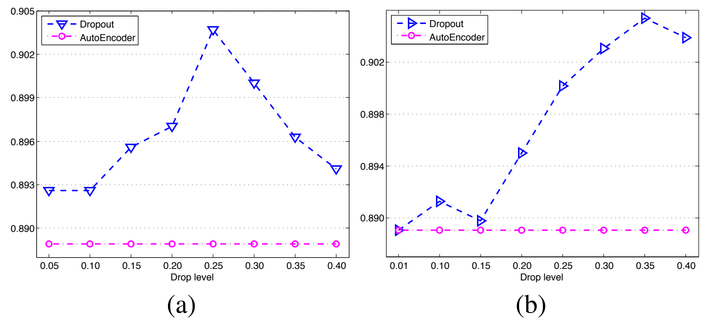


Fig. 18. The performance obtained using autoencoder with and without dropout. Two structural networks with (a) 600-hidden-neuron and (b) 1200-hidden-neuron are built. The recognition accuracy of the former network forms a unimodal sequence, while the performance of the latter network is monotonically increased.

As can be seen from Fig. 18, autoencoder with and without dropout trick demonstrates the similar trend in comparison to autoencoder with and without denoising, as shown in Fig. 17. Autoencoder with all levels of dropout performs much better than the prototype of autoencoder. The best recognition accuracy is produced by the 600-hidden-unit network with dropout trick, 2.07% better than autoencoder without dropout. Similarly, for the 1200-hidden-neuron network, the best recognition rate for autoencoder with dropout is 0.9104, 1.63% better than the archetypal autoencoder. Simultaneously, it can be observed that the curves drawn in Fig. 18 (a) and (b) are slightly different. The recognition accuracy is approximately monotonically increased for the 1200-hidden-unit network, while the recognition rate produced by the 600-hidden-neuron network forms a unimodal sequence.

9) Contractive AutoEncoder: Different from denoising and dropout tricks, where the corrupted (or dropped) nodes are randomly determined, contractive autoencoder presents a deterministic constraint. The new constraint term is defined as the Frobenius norm of Jacobian matrix of hidden neurons, with which a robust representation can be learned. To validate the performance, a group of experiments are performed. We set the visible layer as 96×96, and change the number of hidden neurons from 400 to 800. The experimental results are shown in Fig. 19.

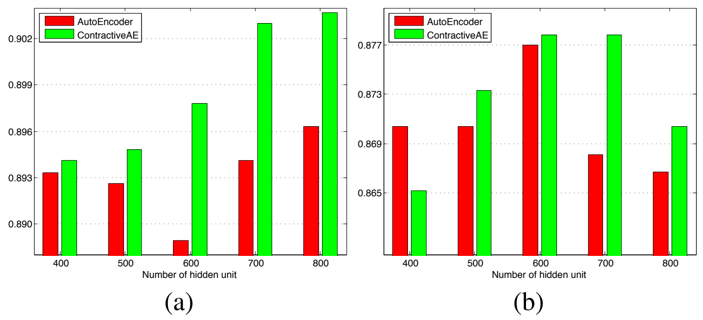


Fig. 19. The comparison of autoencoder with and without Jacobian constraint. (a) hyperbolic tangent activation and (b) sigmoid activation.

We found the similar conclusion can be concluded from Fig. 19 (a) and (b). The performance obtained using contractive autoencoder is consistently better than the archetypal autoencoder except for the 400-hidden-neuron network with sigmoid activation. The improvement is much more significant if the hyperbolic tangent function is employed to achieve activation. The recognition accuracy is monotonically increased with the number of hidden neurons increased. Contrarily, the recognition rate of the network with sigmoid activation forms an approximate unimodal pattern. On the other hand, though the recognition performance can be improved by contractive autoencoder, the computational cost is much more intensive than autoencoder. The memory consumption of contractive autoencoder is much huge than autoencoder [3].

10)Variational AutoEncoder: Variational autoencoder is usually composed of a generative model and a recognition model. It can be viewed as a special form of autoencoder, where the recognition model plays the role of an encoder, and the generative model is regarded as the decoder. The most arresting characteristics of variational autoencoder consists in the model parameters, which are sampled from a certain statistical distribution. Variational autoencoder imposes the constraint terms on the hidden neurons, and hence is similar to contractive

[3]Our 2Gb GPU (GeForce GTX 750 Ti) memory could only build the network within 800 hidden neurons. The network with more hidden neurons makes the GPU memory overflow.

autoencoder. To verify the performance, we perform a set of experiments. We manually change the dimension of the latent representation. The experimental results are given in Fig. 20, where some statistics (minimum, median, maximum, the 25th and 75th percentiles) of 10 sample runs are reported[4].

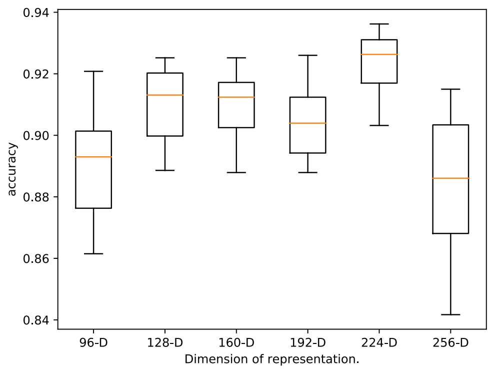


Fig. 20. The recognition performance obtained using variational autoencoder.

As can be seen from Fig. 20, the recognition performance obtained is slightly varied. Though the recognition accuracy is gradually promoted with the dimension of learned representation increased, the improvement is much slight. The 128-D, 160-D, and 192-D learned representation even produce the similar performance. The performance obtained by 224-dimension latent representation is much better than the remaining. Moreover, the recognition accuracy is much more stable than the baseline. The 256-dimension learned representation generates the poorest performance. Moreover, the fluctuation of recognition accuracy is much sharper than the others. It could be therefore concluded that the recognition performance is not proportional to the dimension of learned representation.

11) Convolutional AutoEncoder: Convolutional autoencoder (CAE) is developed by combining convolutional neural network and autoencoder. Different from the archetypal autoencoder, convolutional autoencoder shares the kernel weights and bias among all locations in the input, and hence could preserve the spatial locality. To evaluate the performance, we conduct a group of experiments. We manually change the size of convolutional kernel, and pursue a quantitative comparison with the deep CNN. Three kinds of convolutional kernels, 3×3, 5×5, and 7×7 are verified. The experimental results are displayed in Fig. 21, where some statistics (maximum, mean, minimum, percentiles) are reported[5]. The convolutional neural network is employed as the baseline. To avoid over-fitting, 3-layer CNN and CAE architectures are constructed.

As can be seen from Fig. 21, the recognition performance for convolutional autoencoder is changed for different size

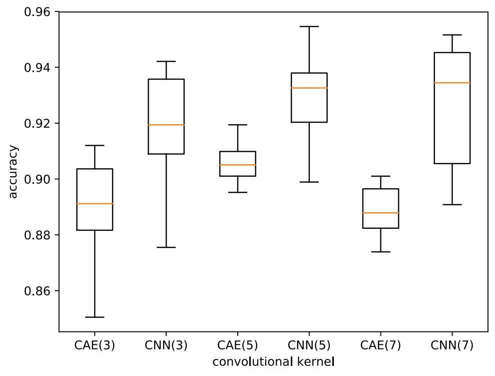


Fig. 21. The recognition accuracy obtained using convolutional autoencoder and convolutional neural network. Three kinds of convolutional kernels, 3×3, 5×5, and 7×7 are tested.

of convolutional kernel. The recognition accuracy obtained using 5×5 convolutional kernel is much better than 3×3 and 7×7. In addition, CAE(5) produces the most robust recognition performance. As for 3-layer convolutional neural network, it always outperforms the convolutional autoencoder, whichever convolutional kernel is employed. The result is not surprising due to their working mechanism. The deep CNN is initially developed for the task of large-scale classification, such as ImageNet, GoogLeNet, VGGNet. It aims to achieving pattern classification. Differently, the convolutional autoencoder is a variant of autoencoder. Though it inherits some advantages of convolutional neural network, the fundamental purpose is to learn a good representation. A good representation will not necessarily produce an ideal classification accuracy. On the other hand, we found that the performance obtained using convolutional autoencoder is much more stable than convolutional neural network.

12) The Depth of Network: The neural network architecture of autoencoder is formed by concatenating the basic model layer by layer. The output (hidden state) of previous layer is wired to the inputs of the successive layer. To study which depth of network is appropriate to our task, we conduct a groups of experiments. Several different structural networks are configured. The sigmoid function is used to achieve activation. The experimental results are given in TABLE III, where the overall recognition rate is reported.

TABLE III. THE RECOGNITION ACCURACIES OBTAINED USING STACKED AUTOENCODER.

Input Layer	$96 \times 96 = 9216$−Node					
AutoEncoder	1000	1000	1000	1000	1000	1000
Stacked AE$^{(2)}$	↓	400	400	400	400	400
Stacked AE$^{(3)}$	↓	↓	1000	1000	1000	1000
Stacked AE$^{(4)}$	↓	↓	↓	500	500	500
Stacked AE$^{(5)}$	↓	↓	↓	↓	1000	1000
Stacked AE$^{(6)}$	↓	↓	↓	↓	↓	500
Output Layer	Softmax classifier (3−class)					
Accuracy	0.8711	0.9119	0.9119	0.9193	0.9237	0.9244

As can be seen, the performance obtained using stacked

[4]The set of experiments are implemented by Keras library. The model parameters are sampled from the normal distribution $\mathcal{N}(0, 1)$.

[5]The experiments are pursued by Keras library with tensorflow backend. We build three convolutional layers, followed by the 2×2 max-pooling trick. For convolutional autoencoder, an up-sampling layer is followed by the deconvolutional layer. For 3-layer CNN, the learned representation is flattened to input a softmax classifier, a fully connected layer.

autoencoder is much better than the single-hidden-layer model. The recognition accuracy for two-hidden-layer stacked autoencoder is 4.08% better than the archetypal autoencoder. The recognition accuracies are gradually improved with the network deepened. The deeper the network, the better the performance. However, the recognition performance reaches a plateau when the depth of network is beyond 3-hidden-layer. Stacked autoencoder with six-hidden-layer only outperforms the one with two-hidden-layer 1.25%. We can therefore conclude that although the representation power can be inherited by stacked autoencoder, the recognition performance could not be proportional to the depth of network. For the task of classification with limited training samples, the deep architecture usually sinks into the over-fitting. It is therefore important to configure a network with an appropriate structure.

B. Extended Evaluation

The related factors of autoencoder have been studied previously. All of the experiments are performed under the standard conditions. This section devotes to the extended evaluation under the non-literal settings, including the changes of aspect view, configuration, and depression angle, articulation and occlusion. Two learning strategies popularly used, support vector machine (SVM) and sparse representation-based classification (SRC) [105], are used to provide the baseline performance. The input features are directly resulting from the raw intensity values. Linear kernel is employed to implement target classification. In the preceding work [106], a family of correlation pattern recognition were reviewed. The representative application to radar target recognition include optional tradeoff synthetic discriminant function filter (OTSDF) [107] and minimum noise and correlation energy filter (MINACE) [108]. A set of correlation filters were generated by the Fourier transformed coefficients. The inference is reached according to the correlation response to the generated filters. Sparse representation of monogenic signal (MSRC) [89], [109] is a recently developed strategy for target recognition in radar images. The monogenic signal is used to characterize target signature, while sparse signal modeling is utilized to implement classification. All of these methods employ the global, predefined, handcrafted features for target recognition, and hence are specified as the baseline.

1) Configuration Change: The configuration refers to the additional or removal of discrete components on the target, such as physical difference, structural modification. All of the variants can be categorized as a single class in military sense. Exemplars on configuration variation are listed in TABLE IV.

TABLE IV. TYPICAL EXAMPLES ON CONFIGURATION CHANGE.

Category	**Examples**
Version Variants	Smoke grenade, Launchers, Side skirts
Configuration Variants	Two cables, Fuel barrels
Structural Changes	Dented fenders, Broken antenna mount

To evaluate the performance under configuration change, images of four targets, BMP2, BTR60, T72 and T62 are utilized, among which BMP2, BTR60 are armored personnel carriers, while T72, T62 are main-battle tanks. Both two pairs of vehicles demonstrate much similar scattering phenomenology. Moreover, BMP2 and T72 still have several configuration variants, noted by the series number. The detail is tabulated in TABLE V. The standards, BMP2_SN_9563 and T72_SN_132 taken at 17° depression angle are used to train the algorithms, while the variants, BMP2_SN_9566, BMP2_SN_c21, and T72_SN_812, T72_SN_s7 captured at 15° depression angle are specified for testing. The configurations used for training are not contained in the query set. The task of target recognition is therefore much challenging than the previous experiments.

TABLE V. ASPECT VIEWS OF DIFFERENT CONFIGURATIONS.

Target	SeriesNo.	Training	Testing
BMP2	SN_9563	233	—
	SN_9566	—	196
	SN_c21	—	196
BTR60	k10yt7532	256	195
BMP2	SN_132	232	—
	SN_812	—	195
	SN_s7	—	191
T62	A51	299	273
Total		1020	1246

TABLE VI reports the experimental results, where Stacked AE$^{(n)}$ denotes stacked autoencoder with n hidden layers [6]. As can be seen, the whole performance is much poorer than the standard evaluation in the previous experiments. The recognition accuracy is 0.8443 for autoencoder, lower than the standard verification. The drop of recognition accuracy can be attributed to the hard settings. In this round of experiments, both the configuration and the depression angle are significantly different between the images available for training and those for testing. The performance obtained using autoencoder is comparable to (or slightly better than) SVM with raw intensity values and the correlation filters, OTSDF and MINACE. Some improvement has been achieved by stacked autoencoders. The recognition accuracy is 0.8540 for Stacked AE$^{(2)}$, 0.8581 for Stacked AE$^{(3)}$, 0.8694 for Stacked AE$^{(4)}$, and 0.8726 for Stacked AE$^{(5)}$, 2.34%, 2.75%, 3.88%, 4.20% better than the single-hidden-layer autoencoder. Though the performance can be improved by stacked autoencoders, the improvement is slight. The recognition accuracy is only comparable to sparse representation with shallow representation and 3-layer CNN, and lower than the preceding works [89], 0.8748. Much more effort should be pursued to further promote the performance under the configuration changes.

2) Articulation and Occlusion: In the field of radar image interpretation, articulation and occlusion generally refer to the relative movement between different attached parts on the target. It is usually designated as continuous, such as tank turret rotation and gun elevation, or discrete, such as opening and closing the hatches and doors. A pair of examples are shown in Fig. 22.

[6]The recognition accuracy of SVM-Raw and SRC-Raw are consistent with the results reported in our preceding works [88], [89].

TABLE VI. PERFORMANCE ON CONFIGURATION CHANGES.

Algorithm	*Handcrafted Feature & Network Architecture*	*Accuracy*
SVM-Raw	The raw intensity values	0.8331
SRC-Raw	The raw intensity values	0.8708
MSRC	The monogenic signal	0.8748
OTSDF	The Fourier transformed coefficients	0.8373
MINACE	The Fourier transformed coefficients	0.8354
AutoEncoder	9216→1200→4	0.8443
Stacked AE$^{(2)}$	9216→1200→500→4	0.8540
Stacked AE$^{(3)}$	9216→1200→500→1200→4	0.8581
Stacked AE$^{(4)}$	9216→1200→500→1200→500→4	0.8694
Stacked AE$^{(5)}$	9216→1200→500→1200→500→600→4	0.8726
ConvNN	3-layer convolutional neural network	0.8664

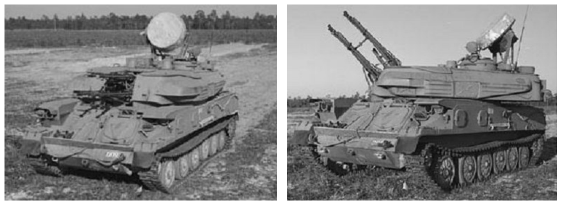


Fig. 22. Target articulation, turret straight and turret articulated of ZSU23/4.

To validate the performance under the operating of articulation and occlusion, we pursue a set of experiments. Images of three military vehicles, ZSU23/4, 2S1, and BRDM_2 are employed, among which BRDM_2 and ZSU23/4 have the several articulated variants. The standards taken at 17° depression angle are used for training, while the variants collected at 45° depression angle are employed for testing. The details are tabulated in TABLE VII. Images available for training and those for testing are taken in two significantly different operating conditions. Moreover, the target may be on different states, such as moving the gun. The task of target recognition is therefore much more difficult than the previous ones.

TABLE VII. ASPECT VIEWS OF STANDARD AND ARTICULATED.

Target	SeriesNo.	**Training**	**Testing**	
			Standard	*Articulated*
2S1	b01	299	303	—
BRDM_2	E-71	298	303	120
ZSU23/4	d08	299	303	119
Total		896	1148	

The results is reported as the overall recognition rate, as listed in TABLE VIII[7]. The recognition performance is much poorer than the above experiments. The best recognition accuracy is even lower than 0.75. The sharp drop of recognition accuracy is caused mainly by the harsh experiment setting. Images used for training are taken at 17° depression angle, while the ones used for testing are collected at 45° depression angle. A drastic change of 28° exists between the images available for training and the ones for testing. In addition, BRMD_2 and ZSU23/4 still have several articulated variants, as detailed in TABLE VII. The recognition accuracy obtained using the correlation filters (OTSDF and MINACE) are only around 0.4, much lower than the remaining. The performance obtained using autoencoder neural network is much better than the baseline, even though the experimental setting is much tough. The single-hidden-layer autoencoder produces the recognition accuracy of 0.6727, 14.66%, 13.61%, and 3.33% better than the competitors, SVM-Raw, SRC-Raw, and MSRC. It is slightly lower than 3-layer convolutional neural network, 0.6871. The best performance, 0.7436, is obtained by Stacked AE$^{(4)}$. It is 0.46%, 3.0%, 4.72%, 7.09% better than Stacked AE$^{(5)}$, Stacked AE$^{(3)}$, Stacked AE$^{(2)}$, and autoencoder. The recognition accuracy obtained using Stacked AE$^{(4)}$ is better than Stacked AE$^{(5)}$. The phenomenon proves that the classification accuracy is not necessarily proportional to the depth of neural network. It is therefore necessary to tune an appropriate structure of neural network according to our task at hand.

[7]Again, the recognition rates of SVM-Raw and SRC-Raw are resulting from the preceding works [88], [89]

TABLE VIII. PERFORMANCE ON ARTICULATION AND DEPRESSION VARIATIONS.

Algorithm	*Architecture*	*Accuracy*
SVM-Raw	Raw intensity values	0.5261
SRC-Raw	Raw intensity values	0.5366
MSRC	Monogenic signal	0.6394
OTSDF	Fourier transformed coefficients	0.4486
MINACE	Fourier transformed coefficients	0.3998
AutoEncoder	9216→2400→3	0.6727
Stacked AE$^{(2)}$	9216→2400→400→3	0.6964
Stacked AE$^{(3)}$	9216→2400→400→800→3	0.7136
Stacked AE$^{(4)}$	9216→2400→400→800→600→3	0.7436
Stacked AE$^{(5)}$	9216→2400→400→800→600→500→3	0.7390
ConvNN	3-layer convolutional neural network	0.6871

C. Discussion

This section devotes to the verification of autoencoder and the variants. Two family of experimental plans, fundamental validation and extended evaluation are pursued. The former aims to studying the impacts of related factors and verifying the configuration of neural network. The latter intends evaluating the performance under the extended operating conditions, such as the depression angle and configuration changes, articulation and occlusion. The comparative studies demonstrate that

- the improvement can be obtained by stacked autoencoder compared to the basic model.
- the configuration of network plays an important role for representation learning.
- the autoencoder neural network could handle the nonliteral experimental settings in some degree.

Though autoencoder and the deep models could handle the extended operating conditions, this problem is far more to be solved. It is much difficult to tune a suitable network structure. In addition, the performance is needed to be promoted further.

VII. CONCLUSION

In this article, we attempt to pursue a systematic review of autoencoder and its variants. Much more attention are paid to those applications to remote sensing and radar image interpretations. The contributions of the existing works are categorized from four viewpoints, the implementation of information fusion, the development of cost function, the renew of learning process, and the achievement of transfer learning. Some comparative studies are performed from the perspective of target recognition in SAR images. The performance are validated by two kinds of experimental plans, fundamental verification under the standard setting and extended evaluation in the non-literal conditions. The quantitative comparison leads to interesting hints about the logical configuration of deep network for the task at hand. The main conclusion concluded from our study include four-fold:

- The latent representation learned by deep neural network does outperforms those handcrafted, shallow, predefined features, as proved in the preceding works.
- The drastic fluctuation on recognition performance can be produced if the network are configured unsuitably. A neural network with inappropriate configurations may degrade the performance.
- There is no certain network structure that could provide the best performance consistently. It is therefore important to tune the network flexibly according to the task at hand.
- The performance can be further improved by deep neural network, especially for target recognition under the extended operating conditions.

This article pursues an attempt on the application of autoencoder to target recognition in SAR images. Though some improvement have been achieved, much more difficult works should be pursued further. The most urgent problem is how to handle the real-world sources of variability. In the future research, we aim to dealing with target recognition under the non-literal conditions. Some new tricks will be develoepd, including the initialization via pre-training, the update of learning algorithm, and an adaptive architecture of neural network, the tuning skill to network structure. Thought the relative studies have been done in the preceding works [2], [43], [110], the further research in SAR target recognition is yet to be uncovered.

ACKNOWLEDGMENT

This work was supported by the National Science Fund for Distinguished Young Scholars of China under Grant 61525105. The authors would like to thank the handling associate editor and those anonymous reviewer for the great contribution to this paper. Dr. Ganggang Dong would like to thank Dr. Zhouhan Lin for his kind help.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[2] A. Romero, C. Gatta, and G. Camps-Valls, “Unsupervised deep feature extraction for remote sensing image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 3, pp. 1349–1362, March 2016.

[3] F. Zhang, B. Du, and L. Zhang, “Saliency-guided unsupervised feature learning for scene classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 2175–2184, Apr. 2015.

[4] Y. Li, X. Huang, and H. Liu, “Unsupervised deep feature learning for urban village detection from high-resolution remote sensing images,” *ISPRS J. Photogrammetry and Remote Sens.*, vol. 83, no. 8, pp. 567–579, August 2017.

[5] G. Hinton, S. Osindero, and Y. Teh, “A fast learning algorithm for deep belief nets,” *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

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

[7] G. Hinton and R. Zemel, “Autoencoders, minimum description length and Helmholtz free energy,” in *Adv. in Neural Inf. Process. Syst.*, Denver, Colorado, USA, 1994, pp. 3–10.

[8] N. L. Roux and Y. Bengio, “Representational power of restricted Boltzmann machines and deep belief networks,” *Neural Computation*, vol. 20, no. 6, pp. 1631–1649, June 2008.

[9] S. Lawrence, C. Giles, A. C. Tsoi, and A. Back, “Face recognition: a convolutional neural-network approach,” *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan. 1997.

[10] D. T. Grozdic and S. T. Jovicic, “Whispered speech recognition using deep denoising autoencoder and inverse filtering,” *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 25, no. 12, pp. 2313–2322, Dec. 2017.

[11] Y. Dai and G. Wang, “Analyzing tongue images using a conceptual alignment deep autoencoder,” *IEEE Access*, vol. 6, no. 3, pp. 1137–1145, March 2018.

[12] D. Park, Y. Hoshi, and C. C. Kemp, “A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, July 2018.

[13] J. Yu, C. Hong, Y. Rui, and D. Tao, “Multitask autoencoder model for recovering human poses,” *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 5060–5068, June 2018.

[14] M. Ma and C. S. X. Chen, “Deep coupling autoencoder for fault diagnosis with multimodal sensory data,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1137–1145, March 2018.

[15] G. Cheng, J. Han, L. Guo, Z. Liu, S. Bu, and J. Ren, “Effective and efficient midlevel visual elements-oriented land-use classification using VHR remote sensing images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 53, pp. 4238–4249, Aug. 2015.

[16] X. Sun, F. Zhou, J. Dong, F. Gao, Q. Mu, and X. Wang, “Encoding spectral and spatial context information for hyperspectral image classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 12, pp. 2250–2254, Dec. 2017.

[17] X. Ma, H. Wang, and J. Geng, “Spectral spatial classification of hyperspectral image based on deep auto-encoder,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 9, pp. 4073–4085, Sep. 2016.

[18] Y. Chen, L. Jiao, Y. Li, and J. Zhao, “Multilayer projective dictionary pair learning and sparse autoencoder for PolSAR image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, pp. 6683–6694, Dec. 2017.

[19] F. Lv, M. Han, and T. Qiu, “Remote sensing image classification based on ensemble extreme learning machine with stacked autoencoder,” *IEEE Access*, vol. 5, pp. 9021–9031, 2017.

[20] W. Xie, L. Jiao, B. Hou, W. Ma, J. Zhao, S. Zhang, and F. Liu, “POLSAR image classification via Wishart-AE model or Wishart-CAE model,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3604–3615, May. 2017.

[21] J. Geng, H. Wang, J. Fan, and X. Ma, “Deep supervised and contractive neural network for SAR image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2442–2459, Apr. 2017.

[22] E. Li, P. Du, A. Samat, Y. Meng, and M. Che, “Mid-level feature representation via sparse autoencoder for remotely sensed scene clas-

sification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 3, pp. 1068–1081, March 2017.

[23] B. Hou, H. Kou, and L. Jiao, "Classification of Polarimetric SAR images using multilayer autoencoders and superpixels," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 7, pp. 3072–3081, July 2016.

[24] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral-spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 12, pp. 2438–2442, Dec. 2015.

[25] W. Zhou, Z. Shao, C. Diao, and Q. Cheng, "High-resolution remote-sensing imagery retrieval using sparse features by auto-encoder," *Remote Sensing Letters*, vol. 6, no. 10, pp. 775–683, Aug. 2015.

[26] E. Othmana, Y. Bazi, N. Alajlan, H. Alhichri, and F. Melgani, "Using convolutional features and a sparse autoencoder for land-use scene classification," *Int. J. Remote Sens.*, vol. 37, no. 10, pp. 1977–1995, 2016.

[27] Z. Lin, Y. Chen, X. Zhao, and G. Wang, "Spectral-spatial classification of hyperspectral image using autoencoders," in *9th Int. Conf. on Information, Communications and Signal Processing*, Dec. 2013.

[28] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.

[29] J. Geng, J. Fan, H. Wang, X. Ma, B. Li, and F. Chen, "High-resolution SAR image classification via deep convolutional autoencoders," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 11, pp. 2351–2355, Nov. 2015.

[30] M. Kang, K. Ji, X. Leng, X. Xing, and H. Zou, "Synthetic aperture radar target recognition with feature fusion based on a stacked autoencoder," *MDPI Sensors*, vol. 17, no. 192, 2017.

[31] P. Planinsic and D. Gleich, "Temporal change detection in SAR images using log cumulants and stacked autoencoder," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, pp. 297–301, Feb. 2018.

[32] L. Zhang, W. Ma, and D. Zhang, "Stacked sparse autoencoder in PolSAR data classification using local spatial information," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, pp. 1359–1363, Sept. 2016.

[33] S. Malek, F. Melgani, Y. Bazi, and N. Alajlan, "Reconstructing cloud-contaminated multispectral images with contextualized autoencoder neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, pp. 2270–2282, April 2018.

[34] S. Deng, L. Du, C. Li, J. Ding, and H. Liu, "SAR automatic target recognition based on euclidean distance restricted autoencoder," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 7, pp. 3323–3333, Jul. 2017.

[35] K. Liang, H. Chang, Z. Cui, S. Shan, and X. Chen, "Representation learning with smooth autoencoder," in *Asian Conference on Computer Vision*. Singapore: Springer, 2014, pp. 72–86.

[36] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3263–3277, Dec. 2015.

[37] Y. Sun, J. Li, W. Wang, A. Plaza, and Z. Chen, "Active learning based autoencoder for hyperspectral imagery classification," in *IEEE Int'l Geosci. Remote Sens. Symp.*, Beijing, China, July 2016, pp. 469–472.

[38] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[39] Y. Zhou and Y. Wei, "Learning hierarchical spectral-spatial features for hyperspectral image classification," *IEEE Trans. Cybernetics*, vol. 46, no. 7, pp. 1667–1678, July 2016.

[40] H. Wu, B. Liu, W. Su, W. Zhang, and J. Sun, "Deep filter banks for land-use scene classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 12, pp. 1895–1899, Dec. 2016.

[41] H. Kim and A. Hirose, "Unsupervised fine land classification using quaternion autoencoder-based polarization feature extraction and self-organizing mapping," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, pp. 1839–1851, March 2018.

[42] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *ACM Proc. 24th Int. Conf. Mach. Learn.*, June 2007, pp. 759–766.

[43] R. Kemker and C. Kanan, "Self-taught feature learning for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2693–2705, May 2017.

[44] X. Yao, J. Han, G. Cheng, X. Qian, and L. Guo, "Semantic annotation of high-resolution satellite images via weakly supervised learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, pp. 3660–3671, June 2016.

[45] Z. Shao, L. Zhang, and L. Wang, "Stacked sparse autoencoder modeling using the synergy of airborne LiDAR and satellite optical and SAR data to map forest above-ground biomass," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 12, pp. 5569–5582, Dec. 2017.

[46] H. Li and S. Misra, "Prediction of subsurface NMR T2 distributions in a shale petroleum system using variational autoencoder-based neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, pp. 2395–2397, Dec. 2017.

[47] A. Elshamli, G. W. Taylor, A. Berg, and S. Areibi, "Domain adaptation using representation learning for the classification of remote sensing images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 9, pp. 4198–4209, Sept. 2017.

[48] S. De, L. Bruzzone, A. Bhattacharya, F. Bovolo, and S. Chaudhuri, "A novel technique based on deep learning and a synthetic target database for classification of urban areas in PolSAR data," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 1, pp. 154–170, Jan. 2018.

[49] L. Windrim, R. Ramakrishnan, A. Melkumyan, and R. J. Murphy, "A physics-based deep learning approach to shadow invariant representations of hyperspectral images," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 665–677, Feb. 2018.

[50] Y. Sun, Z. Liu, S. Todorovic, and J. Li, "Adaptive boosting for SAR automatic target recognition." *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, pp. 112–125, Jan. 2007.

[51] U. Srinivas, V. Monga, and R. G. Raf, "SAR automatic target recognition using discriminative graphical models," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 50, no. 1, pp. 591–606, Jan. 2014.

[52] A. A. Popescu, I. Gavat, and M. Datcu, "Contextual descriptors for scene classes in very high resolution SAR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 1, pp. 80–84, Jan. 2012.

[53] M. Liu, Y. Wu, P. Zhang, Q. Zhang, Y. Li, and M. Li, "SAR target configuration recognition using locality preserving property and Gaussian mixture distribution," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, no. 2, pp. 268–272, March 2013.

[54] M. Li, Y. Guo, M. Li, G. Luo, and X. Kong, "Coupled dictionary learning for target recognition in SAR images," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 6, pp. 791–795, June 2017.

[55] B. Ding, G. Wen, X. Huang, C. Ma, and X. Yang, "Target recognition in synthetic aperture radar images via matching of attributed scattering centers," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 7, pp. 3334–3347, July 2017.

[56] A. M. Cheriyadat, "Unsupervised feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, pp. 439–451, Jan. 2014.

[57] Y. Yang and S. Newsam, "Geographic image retrieval using local invariant features," *IEEE Trans. Geosci. Remote Sens.*, vol. 51, pp. 818–832, Feb. 2013.

[58] X. Zheng, X. Sun, K. Fu, and H. Wang, "Geographic image retrieval using local invariant features," *IEEE Geosci. Remote Sens. Lett.*, vol. 10, pp. 652–656, July 2013.

[59] W. Shao, W. Yang, and G. Xia, "Extreme value theory-based calibration for the fusion of multiple features in high-resolution satellite scene classification," *Int. J. Remote Sens.*, vol. 34, pp. 8588–8602, Dec. 2013.

[60] D. M. McKeown, S. D. Cochran, S. J. Ford, J. C. McGlone, J. A.

Shufelt, and D. A. Yocum, “Fusion of HYDICE hyperspectral data with panchromatic imagery for cartographic feature extraction,” *IEEE Trans. Geosci. Remote Sens.*, vol. 37, pp. 1261–1277, May 1999.

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

[62] M. Ranzato, C. Poultney, S. Chopra, and Y. Cun, “Efficient learning of sparse representations with an energy-based model,” in *Adv. in Neural Inf. Process. Syst.*, Vancouver, Canada, 2007, pp. 1137–1144.

[63] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *ACM Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 1096–1103.

[64] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

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

[66] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, “Contractive auto-encoders: Explicit invariance during feature extraction,” in *ACM Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, USA, 2011, pp. 833–840.

[67] D. Kingma, S. Mohamed, D. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” in *Adv. in Neural Inf. Process. Syst.*, Montreal, Canada, 2014, pp. 3581–3589.

[68] D. Rezende, S. Mohamed, and D. Wierstra, “Stochastic back-propagation and approximate inference in deep generative models,” in *ACM Proc. 31th Int. Conf. Mach. Learn.*, June 2014, pp. 1278–1286.

[69] J. Masci, U. Meier, D. Cirean, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in *International Conference on Artificial Neural Networks*, 2011, pp. 52–59.

[70] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen. (2013) UFLDL Tutorial: Stacked Autoencoders. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders

[71] S. Hao, W. Wang, Y. Ye, T. Nie, and L. Bruzzone, “Two-stream deep architecture for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, 2018.

[72] X. Zhang, Y. Liang, C. Li, N. Huyan, L. Jiao, and H. Zhou, “Recursive autoencoders-based unsupervised feature learning for hyperspectral image classification,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 11, pp. 1928–1932, 2017.

[73] D. Zhang, J. Han, J. Han, and L. Shao, “Cosaliency detection based on intrasaliency prior transfer and deep intersaliency mining,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 6, pp. 1163–1176, Jun. 2016.

[74] L. Cao, W. Huang, and F. Sun, “Building feature space of extreme learning machine with sparse denoising stacked-autoencoder,” *Neurocomputing*, vol. 174, no. Part A, pp. 60–71, January 2016.

[75] J. Feng, L. Liu, X. Zhang, R. Wang, and H. Liu, “Hyperspectral image classification based on stacked marginal discriminative autoencoder,” in *IEEE Int’l Geosci. Remote Sens. Symp.*, July 2017, pp. 3688–3671.

[76] S. Paul and D. Kumar, “Spectral-spatial classification of hyperspectral data with mutual information based segmented stacked autoencoder approach,” *ISPRS J. Photogrammetry and Remote Sens.*, vol. 138, no. 1, pp. 265–280, April 2018.

[77] X. Han, Y. Zhong, and L. Zhang, “Spatial-spectral unsupervised convolutional sparse auto-encoder classifier for hyperspectral imagery,” *ISPRS J. Photogrammetry and Remote Sens.*, vol. 83, no. 3, pp. 195–206, March 2017.

[78] Z. Lin, K. Ji, M. Kang, X. Leng, and H. Zou, “Deep convolutional highway unit network for sar target classification with limited labeled training data,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, pp. 1091–1095, July 2017.

[79] Q. Song and F. Xu, “Zero-shot learning of SAR target feature space with deep generative neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, pp. 2245–2249, Dec. 2017.

[80] Z. Zhang, H. Wang, F. Xu, and Y.-Q. Jin, “Complex-valued convolutional neural network and its application in polarimetric SAR image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, pp. 7177–7188, Dec. 2017.

[81] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan, “Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, pp. 3322–3337, June 2017.

[82] S. Chen, H. Wang, F. Xu, and Y.-Q. Jin, “Target classification using the deep convolutional networks for SAR images,” *IEEE Trans. Geosci. Remote Sens.*, vol. 54, pp. 4806–4817, Aug. 2016.

[83] E. Keydel, S. Lee, and J. Moore, “MSTAR extended operating conditions - a tutorial,” in *Algorithms for Synthetic Aperture Radar Imagery III, SPIE*, Jun. 1996, pp. 228–242.

[84] J. Thiagarajan, K. Ramamurthy, P. Knee, and *et al.*, “Sparse representation for automatic target classification in SAR images.” in *Int’l Sym. Communcitaion, Control and Signal Process.*, Mar. 2010, pp. 1–4.

[85] Q. Zhao and J. C. Principe, “Support vector machines for SAR automatic target recognition,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 2, pp. 643–654, 2001.

[86] S. Papson and R. M. Narayanan, “Classification via the shadow region in SAR imagery,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 48, no. 2, pp. 969–980, Apr. 2012.

[87] J. Zhu, X. Qiu, Z. Pan, Y. Zhang, and B. Lei, “An improved shape contexts based ship classification in SAR images,” *Remote Sensing*, vol. 9, no. 2, 2017.

[88] G. Dong, G. Kuang, N. Wang, and W. Wang, “Classification via sparse representation of steerable wavelet frames on Grassmann manifold: Application to target recognition in SAR image,” *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2892–2904, Jun. 2017.

[89] G. Dong and G. Kuang, “Classification on the monogenic scale space: Application to target recognition in SAR image,” *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2527–2539, Aug. 2015.

[90] L. Potter and R. Moses, “Attributed scattering centering for SAR ATR,” *IEEE Trans. Image Process.*, vol. 6, no. 1, pp. 79–91, Jun. 1997.

[91] J. Zhou, Z. Shi, X. Cheng, and Q. Fu, “Automatic target recognition of SAR images based on global scattering center model,” *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 10, pp. 3713–3729, Oct. 2011.

[92] H. Liu, B. Jiu, F. Li, and Y. Wang, “Attributed scattering center extraction algorithm based on sparse representation with dictionary refinement,” *IEEE Trans. Antennas Propag.*, vol. 65, no. 5, pp. 2604–2614, May 2017.

[93] M. Koets and R. Moses, “Feature extraction using attributed scattering center models on SAR imagery,” in *Proc. SPIE Algorithms for Synthetic Aperture Radar Imagery VI*, April 1999, pp. 104–115.

[94] E. Ertin and R. L. Moses, “Through-the-wall SAR attributed scattering center feature estimation,” *IEEE Trans. Geosci. Remote Sens.*, vol. 47, pp. 1338–1348, May 2009.

[95] M. Martorella, E. Giusti, A. Capria, F. Berizzi, and B. Bates, “Automatic target recognition by means of polarimetric ISAR images and neural networks,” *IEEE Trans. Geosci. Remote Sens.*, vol. 47, pp. 3786–3794, Nov. 2009.

[96] L. Novak, G. Owirka, and W. Brower, “Performance of 10- and 20-target MSE classifiers,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 36, pp. 1279–1289, Oct. 2000.

[97] A. M. Atto, E. Trouve, Y. Berthoumieu, and G. Mercier, “Multidate divergence matrices for the analysis of SAR image time series,” *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 1922–1938, April 2013.

[98] J. Inglada and G. Mercier, “A new statistical similarity measure for change detection in multitemporal SAR images and its extension

to multiscale change analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 5, pp. 1432–1445, May 2007.

[99] A. C. Frery, A. D. C. Nascimento, and R. J. Cintra, "Analytic expressions for stochastic distances between relaxed complex Wishart distributions," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 2, pp. 1213–1226, Feb 2014.

[100] P. Guccione, A. M. Guarnieri, and M. Zonno, "Azimuth antenna maximum likelihood estimation by persistent point scatterers in SAR images," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 2, pp. 947–955, Feb. 2014.

[101] P. Iervolino and R. Guida, "A novel ship detector based on the generalized-likelihood ratio test for SAR imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3616–3630, May 2017.

[102] P. Ghamisi, J. Plaza, Y. Chen, J. Li, and A. J. Plaza, "Advanced spectral classifiers for hyperspectral images: A review," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 1, pp. 8–32, March 2017.

[103] H. Kamyshanska and R. Memisevic, "The potential energy of an autoencoder," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1261–1273, Jun. 2015.

[104] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv: 1609.04747*, 2016.

[105] J. Wright, A. Yang, A. Ganesh, and *et al.*, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

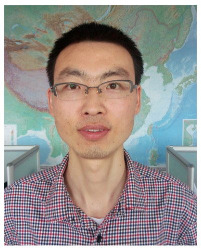
[106] V. K. Kumar, M. Savvides, and C. Xie, "Correlation pattern recognition for face recognition," *Proc. IEEE*, vol. 94, no. 11, pp. 1963–1976, Nov. 2006.

[107] R. Singh and B. Kumar, "Performance of the extended maximum average correlation height filter and the polynomial distance classifier correlation filter for multiclass SAR detection and classification," in *Algorithms for SAR Imagery IX, SPIE*, vol. 4727, Aug. 2002, pp. 265–279.

[108] R. Patnaik and D. Casasent, "MINACE filter classification algorithms for ATR using MSTAR data," in *Automatic Target Recognition XV, Proc. SPIE*, vol. 5807, Aug. 2005, pp. 100–111.

[109] G. Dong, N. Wang, and G. Kuang, "Sparse representation of monogenic signal: with application to target recognition in SAR images," *IEEE Signal Process. Lett.*, vol. 21, no. 8, pp. 952–956, Aug. 2014.

[110] M. S. Seyfioglu, A. M. zbayoglu, and S. Z. Gurbuz, "Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities," *IEEE Trans. Aerosp. Electron. Syst.*, 2018.



Ganggang Dong received M.S. and Ph.D. degrees in information and communication engineering from National University of Defense Technology, Changsha, China, in 2012 and 2016, respectively. Since 2014, he has authorized more than 20 scientific papers in peer-reviewed journals and conferences, including IEEE TIP, IEEE TGRS, IEEE GRSM, IEEE JSTARS, IEEE GRSL, IEEE SPL. His research interests include the applications of compressed sensing and sparse representation, pattern recognition, manifolds learning, and deep neural network.

Guisheng Liao (M'96–SM'16) was born in Guiling, China. He received the B.S. degree from Guangxi University, Nanning, China, in 1985, and the M.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 1990 and 1992, respectively. He is a Professor with Xidian University, where he is also the Dean of the School of Electronic Engineering currently. He has been a Senior Visiting Scholar with The Chinese University of Hong Kong, Hong Kong. His research interests include synthetic aperture radar (SAR), space-time adaptive processing, SAR ground moving target indication, and distributed small satellite SAR system design. Prof. Guisheng Liao is a member of the National Outstanding Person and the Cheung Kong Scholars in China.

Hongwei Liu (M'00) received the B.Eng. degree in electronic engineering from Dalian University of Technology, Dalian, China, in 1992, and the M.Eng. and Ph.D. degrees in electronic engineering from Xidian University, Xi'an, China, in 1995 and 1999, respectively. He is currently a Director and a Professor with the National Laboratory of Radar Signal Processing, Xidian University. His research interests include radar automatic target recognition, radar signal processing, and adaptive signal processing.

Gangyao Kuang(M'11) received the B.S. and M.S. degrees from the Central South University of Technology, Changsha, China, in 1998 and 1991, respectively, and the Ph.D. degree from the National University of Defense Technology, Changsha, in 1995. He is currently a Professor and Director of the Remote Sensing Information Processing Laboratory, National University of Defense Technology. His current interests mainly include remote sensing, SAR image processing, change detection, SAR ground moving target indication, and classification with polarimetric SAR images.