An Overview of Unsupervised Deep Feature Representation for Text Categorization

Shiping Wang, Jinyu Cai, Qihao Lin, and Wenzhong Guo

***Abstract*— High-dimensional features are extensively accessible in machine learning and computer vision areas. How to learn an efficient feature representation for specific learning tasks is invariably a crucial issue. Due to the absence of class label information, unsupervised feature representation is exceedingly challenging. In the last decade, deep learning has captured growing attention from researchers in a broad range of areas. Most of the deep learning methods are supervised, which is required to be fed with a large amount of accurately labeled data points. Nevertheless, acquiring sufficient accurately labeled data is unaffordable in numerous real-world applications, which is suggestive of the needs of unsupervised learning. Toward this end, quite a few unsupervised feature representation approaches based on deep learning have been proposed in recent years. In this paper, we attempt to provide a comprehensive overview of unsupervised deep learning methods and compare their performances in text categorization. Our survey starts with the autoencoder and its representative variants, including sparse autoencoder, stacked autoencoder, contractive autoencoder, denoising autoencoder, variational autoencoder, graph autoencoder, convolutional autoencoder, adversarial autoencoder, and residual autoencoder. Aside from autoencoders, deconvolutional networks, restricted Boltzmann machines, and deep belief nets are introduced. Then, the reviewed unsupervised feature representation methods are compared in terms of text clustering. Extensive experiments in eight publicly available data sets of text documents are conducted to provide a fair test bed for the compared methods.**

***Index Terms*— Autoencoder, deconvolutional network, deep belief nets, deep learning, feature representation, text categorization, unsupervised learning.**

I. Introduction

With the development of Internet and multimedia technology, the amount of data comes with a rapid and steady growth, which contains a great deal of redundant, irrelevant, and inconsistent data. These undesired data may largely deteriorate the performance of specific learning tasks. Consequently, how to learn an efficient feature representation from massive data is extremely critical and urgent. It is known that an effective feature representation comes with satisfactory effects of learning tasks [1], [2]. Feature representation can be categorized as supervised or unsupervised methods. In supervised learning, it is about finding a low-dimensional representation by a trained model with a set of known class labels [3]–[5]. In contrast, unsupervised methods attempt to construct a feature representation by the evaluation of sample similarities [6]–[8].

The concept of deep learning was put forward by Hinton *et al.* [9] and Hinton and Salakhutdinov [10], and it has made persistent advances in recent years [11]–[14]. It is actually a multilayered neural network to simulate and analyze the operation mechanism of human brains. In such a scheme, deep learning attempts to represent, decode, and interpret all kinds of data, including images [15]–[18], sounds [19]–[21], texts [22]–[24], and videos [25]–[28]. Like other machine learning methods, deep learning can also be categorized as supervised and unsupervised approaches. As an example, convolutional neural networks [29]–[31] are supervised deep learning models, while deep belief nets tend to work in unsupervised scenes [32]–[34].

In contrast, shallow neural networks come with a smaller number of layers (below three layers) [35], [36]. This type of networks possesses favorable performances in some specific applications, especially for the cases of insufficient training data [37], [38]. However, the limitations of shallow neural networks are the weak approximation ability and generalization capacity of complex data distributions [39]. Conversely, deep neural networks consist of a large number of hidden layers, which are able to provide well-approximated solutions to varying complex functions [40], [41]. In other words, some complex nonlinear functions are expressible by deep neural networks but cannot be well approximated by any shallow neural network with the same number of neurons [42]. From a theoretic viewpoint, the former works with the lower bound of approximation performance. Simultaneously, deep nets exhibit considerable merits in learning hierarchical semantic feature representations, such as learning parts of objects [43], [44]. The semantic features act frequently as high-level characterizations of visual objects [45], [46].

Nonetheless, a variety of practical applications frequently suffer from the lack of sufficient accurately labeled data. Naturally, learning beneficial patterns and feature representations from this type of data is of great importance. Toward this end, there have been a number of unsupervised learning

Manuscript received November 28, 2018; revised February 26, 2019; accepted April 1, 2019. Date of publication April 26, 2019; date of current version June 10, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant U1705262 and Grant 61672159, in part by the Technology Innovation Platform Project of Fujian Province under Grant 2014H2005 and Grant 2009J1007, in part by the Fujian Collaborative Innovation Center for Big Data Application in Governments, and in part by the Fujian Engineering Research Center of Big Data Analysis and Processing. *(Corresponding author: Shiping Wang.)*

The authors are with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China, and also with the Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China (e-mail: shipingwangphd@163.com; guowenzhong@fzu.edu.cn).

Digital Object Identifier 10.1109/TCSS.2019.2910599

paradigms, such as autoencoder networks [47]–[50], restricted Boltzmann machines (RBMs) [51]–[54], deconvolutional networks [55]–[58], and deep belief nets [59]–[61]. These methods focus more on dimensionality reduction, aiming at providing a series of rewarding solutions to an unsupervised learning.

In this paper, we present a comprehensive overview of unsupervised deep feature representation methods for the problem of text categorization. In the first place, we begin with one widely used type of unsupervised neural networks, namely autoencoder networks, and describe their intrinsic mechanisms and distinguished invariants. Then, other typical representatives of unsupervised deep feature representations are introduced, including deconvolutional networks, deep belief nets, and RBMs. Finally, numerous experiments are conducted in eight publicly available data sets of text documents from machine learning repositories. Extensively experimental results demonstrate the respective benefits of varying unsupervised deep feature representation methods. It is expected that this paper would provide some enlightenments and insights for who are fresh in this area.

The remaining part of this paper is arranged as follows. A number of unsupervised deep feature representation approaches are presented in Section II. And the extensive comparative experiments of the reviewed methods are provided in Section III. Finally, this paper is concluded with remarking work in Section IV.

II. Overview of Unsupervised Deep Feature Representation

To begin with, some frequently used notations are revisited in this paper. For an unsupervised learning task, the set of data points is denoted by $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a d-dimensional column vector. The input data matrix is assumed to be $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$. For an arbitrary matrix $\mathbf{M} = [\mathbf{M}_{ij}]_{m\times n}$, its Frobenius norm (simply called F-norm) is defined as $||\mathbf{M}||_F = \sqrt{\sum_{j=1}^n \sum_{i=1}^m \mathbf{M}_{ij}^2}$, and its L_1-norm is denoted as $||\mathbf{M}||_1 = \sum_{j=1}^n \sum_{i=1}^m |\mathbf{M}_{ij}|$.

This section puts more emphasis on four categories of unsupervised deep feature representations, including autoencoders-like networks, deconvolutional networks, RBMs, and deep belief nets.

A. Autoencoder Network and its Variants

Autoencoder neural network is a well-known unsupervised feature representation method that learns low-dimensional hidden variables with the minimum reconstruction error to the input matrix. In order to capture the certain geometrical structures of the input data, a number of variations of autoencoder have emerged.

1) Autoencoder: An autoencoder neural network can be divided into two symmetric steps: encoder and decoder. The former aims to learn a low-dimensional feature representation of the input data, whereas the latter is to recover the data with the minimum reconstruction error. An encoder can be regarded as a feedforward bottom-up step, while a decoder can be viewed as a feedback top-down generative step. An illustration

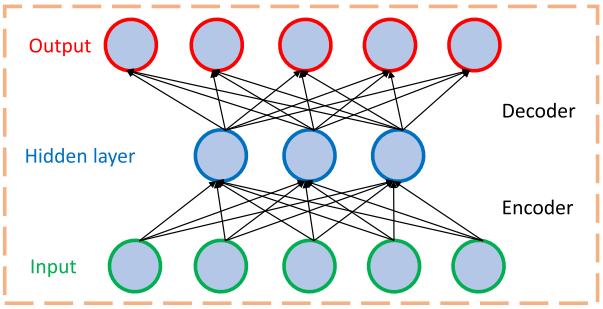

Fig. 1. Framework of autoencoder neural network.

of autoencoder networks is demonstrated in Fig. 1. From Fig. 1, it is observed that each data point $\mathbf{x} = [x^{(1)}; \cdots; x^{(d)}]$ in the input layer is encoded as $\mathbf{y} = [y^{(1)}; \cdots; y^{(r)}]$ with $r \ll d$ in the hidden layer, which is then decoded as $\widehat{\mathbf{x}} = [\widehat{x}^{(1)}; \cdots; \widehat{x}^{(d)}]$ in the output layer.

The output $\widehat{\mathbf{x}}$ is actually an estimator of the input sample $\mathbf{x}$, and loss$(\mathbf{x}, \widehat{\mathbf{x}})$ serves as the performance loss function. Specifically, the hidden layer $\mathbf{y}$ represents the activated neurons by a certain linear transformation of the input layer $\mathbf{x}$. The transformation matrix and bias from the input layer to the hidden layer are denoted by $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times r}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^r$, and those from the hidden layer to the output layer are denoted by $\mathbf{W}^{(2)} \in \mathbb{R}^{r\times d}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^d$. With the aforementioned notation, hidden units $\mathbf{y}$ can be computed by the following canonical form:

$$\mathbf{y} = f(\mathbf{x}) = h_{\mathbf{W}^{(1)},\mathbf{b}^{(1)}}(\mathbf{x}) = \sigma(\mathbf{W}^{(1)^T}\mathbf{x} + \mathbf{b}^{(1)}) \tag{1}$$

where σ is a predefined activation function, typically a logistic sigmoid function $\sigma(z) = (1/1 + e^{-z})$ for any $z \in \mathbb{R}$. Analogously, the output layer can be represented by

$$\begin{aligned}\widehat{\mathbf{x}} = g(f(\mathbf{x})) = h_{\mathbf{W}^{(2)},\mathbf{b}^{(2)}}(\mathbf{y}) &= \sigma(\mathbf{W}^{(2)^T}\mathbf{y} + \mathbf{b}^{(2)}) \\ &= \sigma(\mathbf{W}^{(2)^T}\sigma(\mathbf{W}^{(1)^T}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})\end{aligned} \tag{2}$$

in which the activation functions in encoding and decoding modes are assumed to be the same for model description simplicity. The mappings $f : \mathbb{R}^d \to \mathbb{R}^r$ and $g : \mathbb{R}^r \to \mathbb{R}^d$ are called the encoder and the decoder, respectively.

Without loss of generality, it is assumed that the encoder and the decoder share the same weighted matrix, i.e., $\mathbf{W}^{(1)} = \mathbf{W}^{(2)^T} = \mathbf{W}$ and $\mathbf{b} = [\mathbf{b}_f; \mathbf{b}_g]$ with $\mathbf{b}^{(1)} = \mathbf{b}_f$ and $\mathbf{b}^{(2)} = \mathbf{b}_g$. The optimization objective function of an autoencoder neural network is defined as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^n \ell(\mathbf{x}_i, g(f(\mathbf{x}_i))) = \sum_{i=1}^n \ell(\mathbf{x}_i, \widehat{\mathbf{x}}_i) \tag{3}$$

where $\ell(\cdot,\cdot)$ is a loss function to measure the mutual difference between two variables and $\widehat{\mathbf{x}}_i = g(f(\mathbf{x}_i)) = \sigma(\mathbf{W}^T\sigma(\mathbf{W}^T\mathbf{x}_i + \mathbf{b}_f) + \mathbf{b}_g)$. For simplicity, $\ell(\cdot,\cdot)$ is specified as the Euclidean distance. Therefore, the above-mentioned optimization objective is written as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^n ||\mathbf{x}_i - g(f(\mathbf{x}_i))||_2^2. \tag{4}$$

Algorithm 1 Algorithm for Autoencoder Neural Network

Input: Data points $\{\mathbf{x}_i\}_{i=1}^n$, the number of hidden units r, and learning rate α.

Output: Shared weighted matrix $\mathbf{W}$ and bias vectors $\mathbf{b}_f, \mathbf{b}_g$, denoted by $\mathbf{b} = [\mathbf{b}_f; \mathbf{b}_g]$.

Initialize $\mathbf{W}$ and $\mathbf{b}$;
Randomly generate mini-batches $\{\mathbf{X}_i\}_{i=1}^k$ with $\mathbf{X}_i \in \mathbb{R}^{d\times n_i}$ and $n = \sum_{j=1}^k n_j$, forming a partition of $\mathbf{X}$;
repeat
  for each epoch $(i = 1, \ldots, k)$ **do**
    Update $\mathbf{W}$ with $\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial}{\partial \mathbf{W}} J(\mathbf{W}, \mathbf{b})$;
    Update $\mathbf{b}$ with $\mathbf{b} \leftarrow \mathbf{b} - \alpha \frac{\partial}{\partial \mathbf{b}} J(\mathbf{W}, \mathbf{b})$;
  end for
until convergence
return $\mathbf{W}$ and $\mathbf{b}$.

Assuming that $\widehat{\mathbf{X}} = [\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_n]$, the optimization problem of autoencoder neural networks is then rewritten as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}) = ||\mathbf{X} - g(f(\mathbf{X}))||_F^2 = ||\mathbf{X} - \widehat{\mathbf{X}}||_F^2 \tag{5}$$

with $g(f(\mathbf{X})) = \widehat{\mathbf{X}} = \sigma(\mathbf{W}^T \sigma(\mathbf{W}^T \mathbf{X} + \mathbf{b}_f) + \mathbf{b}_g)$.

The aforementioned matrix characterization for the autoencoder network provides a batched optimization technique. The above-mentioned optimization problem can be solved by the mini-batch gradient descent method, as demonstrated in Algorithm 1.

2) Sparse Autoencoder: For certain learning tasks, most neurons of an autoencoder neural network are not activated. Assuming that a weighted matrix $\mathbf{W}$ and two bias vectors $\{\mathbf{b}^{(l)}\}_{l=1}^2$ will be learned, there are $d \times r + d + r$ parameters to be solved, where the parameter number may be greater than the known sample number, being indicative of the high risk of overfitting. Besides, the simplest form of autoencoders may lead to weight-decay, suggesting that the corresponding neurons are optimized by small weights. Sparse autoencoder neural network is based on the assumption that only a small number of neurons are activated [62], [63] when addressing specific learning tasks. It requires that both weighted matrices and output hidden units are constrained to be sparse. Therefore, the optimization problem of the sparse autoencoder neural network is formulated as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}) + \lambda ||\mathbf{W}||_1 + \mu \sum_{j=1}^{r} KL(\rho|\widehat{\rho}_j) \tag{6}$$

where λ and μ are the two regularized coefficients to balance the fitting term and the sparsity term, ρ is a predefined sparsity parameter, and $KL(\rho|\widehat{\rho}_j) = \rho \log((\rho/\widehat{\rho}_j)) + (1-\rho)\log((1-\rho/1-\widehat{\rho}_j))$ is the Kullback–Leibler (KL)-divergence to guarantee the sparsity of the output hidden variables. Herein

$$\widehat{\rho}_j = \frac{1}{n} \sum_{i=1}^{n} \sigma\left(\mathbf{w}_j^T \mathbf{X} + \mathbf{b}_j^{(1)}\right) \mathbf{x}_i \tag{7}$$

with $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{d\times r}$.

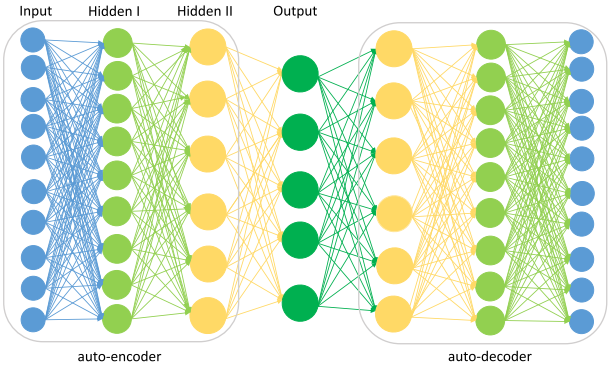

Fig. 2. Illustration of stacked autoencoder neural network wherein only two hidden layers are demonstrated.

3) Stacked Autoencoder: The aforementioned autoencoder neural network indicates that a low-dimensional representation $\mathbf{Y} \in \mathbb{R}^{r\times n}$ is available for a given data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$. However, the autoencoder network just contains one hidden layer, which implies weak abilities to learn a nonlinear feature representation, though nonlinear activation function is employed in each layer. Naturally, adding more hidden layers is a prior choice to learn nonlinear high-level features. A stacked autoencoder [64], [65] is to construct a network with multiple hidden layers based on greedy strategies. The stacked autoencoder network with m hidden layers can be simply regarded as a combination of m autoencoders. It can also be viewed as m pairs of encoder and decoder, as shown in Fig. 2.

In the hidden layers, the encoding parameter from the $(l-1)$th layer to the lth layer is denoted by $(\mathbf{W}^{(l)}, \mathbf{b}_f^{(l)})$, and its corresponding decoding parameter is written as $({\mathbf{W}^{(l)}}^T, \mathbf{b}_g^{(l)})$. Hence, the hyperparameter space of a stacked autoencoder can be denoted as $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^m$ with $\mathbf{b}^{(l)} = [\mathbf{b}_f^{(l)}; \mathbf{b}_g^{(l)}]$. With an input data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, the mth hidden layer to be solved can be represented as

$$\begin{aligned}\mathbf{Y} &= f_m(f_{m-1} \cdots (f_1(\mathbf{X}))) \\ &= \sigma\left({\mathbf{W}^{(m)}}^T \cdots \sigma\left({\mathbf{W}^{(1)}}^T \mathbf{X} + \mathbf{b}_f^{(1)}\right) + \mathbf{b}_f^{(m)}\right)\end{aligned} \tag{8}$$

and the reconstructed data matrix $\widehat{\mathbf{X}}$ is expressed as

$$\begin{aligned}\widehat{\mathbf{X}} &= g_1(g_2 \cdots (g_m(\mathbf{Y}))) \\ &= \sigma\left({\mathbf{W}^{(1)}}^T \cdots \sigma\left({\mathbf{W}^{(m)}}^T \mathbf{Y} + \mathbf{b}_g^{(m)}\right) + \mathbf{b}_g^{(1)}\right).\end{aligned} \tag{9}$$

Therefore, the overall cost loss of a stacked autoencoder neural network is $J(\mathbf{W}, \mathbf{b}) = ||\mathbf{X} - g_1(g_2 \cdots (g_m(f_m(f_{m-1} \cdots (f_1(\mathbf{X})))))||_F^2$, where $\mathbf{W} \triangleq \{\mathbf{W}^{(l)}\}_{l=1}^m$ and $\mathbf{b} \triangleq \{\mathbf{b}^{(l)}\}_{l=1}^m$. With a layerwise greedy strategy, the optimization objective function of a stacked autoencoder can be represented as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W}, \mathbf{b}) = \sum_{l=1}^{m} ||\mathbf{O}_{l-1} - g_l(f_l(\mathbf{O}_{l-1}))||_F^2 \tag{10}$$

where $\mathbf{O}_0$ is the input layer, i.e., $\mathbf{O}_0 = \mathbf{X}$, and $\mathbf{O}_l = f_l(\mathbf{O}_{l-1})$ is the lth layer output for any $l \in \{1, \ldots, m\}$.

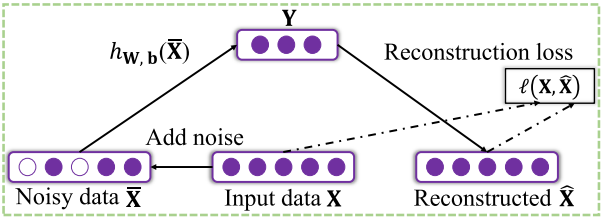

Fig. 3. Architecture of denoising autoencoder.

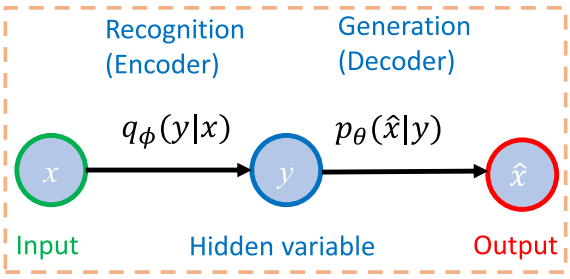

Fig. 4. Illustration of variational autoencoder neural networks.

4) Contractive Autoencoder: Beyond sparsity and multilayers, the robustness of the hidden variables is an indispensable point for a neural network. Toward this end, a measure was proposed to encourage the robustness of the learned low-dimensional feature representation $f(\mathbf{x})$. This measure is represented as the Frobenius norm of the Jacobian matrix $J_f(\mathbf{x})$ of a nonlinear mapping f [66], defined by

$$||J_f(\mathbf{x})||_F^2 = \sum_{i,j}\left(\frac{\partial h_j(\mathbf{x})}{\partial \mathbf{x}_i}\right)^2 \tag{11}$$

where $h_j(\mathbf{x})$ is the jth element of the low-dimensional hidden representation $h(\mathbf{x})$ encoded by f. Particularly, the Jacobian matrix can be represented by

$$J_f(\mathbf{x}) = \begin{bmatrix} \frac{\partial h_1(\mathbf{x})}{\partial \mathbf{x}_1} & \cdots & \frac{\partial h_r(\mathbf{x})}{\partial \mathbf{x}_1} \\ \vdots & & \vdots \\ \frac{\partial h_1(\mathbf{x})}{\partial \mathbf{x}_n} & \cdots & \frac{\partial h_r(\mathbf{x})}{\partial \mathbf{x}_n} \end{bmatrix} \tag{12}$$

where r is the number of hidden units. Adding a regularization term associated with the Jacobian matrix of nonlinear mappings, a contractive autoencoder neural network [67] is formalized as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W},\mathbf{b}) = \sum_{i=1}^{n} \ell(\mathbf{x}_i, g(f(\mathbf{x}_i))) + \lambda||J_f(\mathbf{x}_i)||_F^2 \tag{13}$$

where λ is a regularization parameter. By a batched matrix formulation, the aforementioned optimization objective problem is equivalently transformed into

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W},\mathbf{b}) = \ell(\mathbf{X}, g(f(\mathbf{X}))) + \lambda||J_f(\mathbf{X})||_F^2 \tag{14}$$

in which $J_f(\mathbf{X}) \in \mathbb{R}^{n\times n\times r}$ is a 3-D tensor of which the ith entry is equal to $J_f(\mathbf{x}_i)$ for any $i \in \{1,\ldots,n\}$.

5) Denoising Autoencoder: Deep neural networks tend to rely largely on accurately labeled input data, which implies that these networks frequently fail to learn effective discriminative features from partially destroyed data. Denoising autoencoder neural network [68], [69] aims at reconstructing clean data from given noisy data.

The architecture of denoising autoencoders is shown in Fig. 3. It is observed from Fig. 3 that a clean input data point $\mathbf{x}$ is preprocessed as corrupted $\overline{\mathbf{x}}$ by adding an amount of noises obeying certain probability distribution, such as binomial noises or Gaussian noises [70]. Each corrupted data point $\overline{\mathbf{x}}$ is mapped onto a low-dimensional hidden representation $\mathbf{y} = h_{\mathbf{W},\mathbf{b}}(\overline{\mathbf{x}}) = \sigma(\mathbf{W}^T\overline{\mathbf{x}} + \mathbf{b}_f)$ using an autoencoder. Using hidden units $\{\mathbf{y}_i\}_{i=1}^n$, we reconstruct the original input data points $\{\mathbf{x}_i\}_{i=1}^n$, instead of the corrupted data points $\{\overline{\mathbf{x}}_i\}_{i=1}^n$. Therefore, the optimization objective function of a denoising autoencoder neural network can be represented as

$$\min_{\mathbf{W},\mathbf{b}} J(\mathbf{W},\mathbf{b}) = \ell(\mathbf{X},\widehat{\mathbf{X}}) = \ell(\mathbf{X}, g(f(\overline{\mathbf{X}}))) \tag{15}$$

where $\widehat{\mathbf{X}} = \sigma(\mathbf{W}^T\sigma(\mathbf{W}^T\overline{\mathbf{X}}+\mathbf{b}_f))+\mathbf{b}_g$ with $\overline{\mathbf{X}} = [\overline{\mathbf{x}}_1,\ldots,\overline{\mathbf{x}}_n]$. Since $\overline{\mathbf{x}}$ is assumed to be a random variable by adding noise to $\mathbf{x}$, $\ell(\cdot,\cdot)$ tends to be defined as a probability loss function, such as cross entropy.

6) Variational Autoencoder: Given any high-dimensional input data point, autoencoders learn a low-dimensional hidden representation with minimum reconstruction error. It is observed that the learned hidden variable is closely related to the input data matrix, indicating that it fails to generate an arbitrary sample. Hence, the hidden layer is further constrained with certain conditions and regularizations, such that it can produce any sample at a relatively independent manner. Toward this end, a variational autoencoder neural network [71], [72] provides a probabilistic view to learn regularized hidden units.

Given an input data set $\{\mathbf{x}_i\}_{i=1}^n$ of n independent identically distributed (i.i.d) samples from a random variable $\mathbf{x}$, it is assumed that the given data points are generated by a given random process with an unobserved hidden random variable $\mathbf{y}$. The variational autoencoder aims to learn a hidden random variable (also called a latent variable) $\mathbf{y}$, as shown in Fig. 4. It is observed from Fig. 4 that variational autoencoders consist of two steps, namely a recognition model (a probabilistic encoder) and a generative model (a probabilistic decoder). The former serves as an encoder to learn the hidden variable $\mathbf{y}$ with conditional probability $q_\phi(\mathbf{y}|\mathbf{x})$, while the latter acts as a decoder to produce $\widehat{\mathbf{x}}$ with a generative model $p_\theta(\widehat{\mathbf{x}}|\mathbf{y})$ with minimum reconstruction loss to $\mathbf{x}$. Herein, parameters ϕ and θ need to be specified into certain distribution forms. When specifying the distribution (e.g., Gaussian distribution) that $p_\theta(\widehat{\mathbf{x}}|\mathbf{y})$ obeys and the estimated parameter θ, $p_\theta(\cdot)$ can be used to generate any adversarial sample $\widehat{\mathbf{x}}$.

Given an input random variable $\mathbf{x}$, variational autoencoders aim at learning a hidden random variable $\mathbf{y}$, satisfying the following two conditions: 1) the reconstruction loss $\ell(\cdot,\cdot)$ should be minimized and 2) the hidden variable $\mathbf{y}$ should obey a given distribution. According to Bayes' formula, the true

posterior density can be represented as

$$p_\theta(\mathbf{y}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{y})\, p_\theta(\mathbf{y})}{p_\theta(\mathbf{x})}. \tag{16}$$

Naturally, the conditional distribution $q_\phi(\mathbf{y}|\mathbf{x})$ should be the best estimation of the true posterior distribution $p_\theta(\mathbf{x}|\mathbf{y})$. The KL-divergence is employed to measure the difference between both distributions. Using maximum likelihood estimation, the marginal likelihood is expressed as

$$\log p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i). \tag{17}$$

Here, the marginal likelihood of each data point can be written as

$$\log p_\theta(\mathbf{x}_i) = D_{KL}(q_\phi(\mathbf{y}|\mathbf{x}_i)||p_\theta(\mathbf{y}|\mathbf{x}_i)) + \ell(\phi, \theta; \mathbf{x}_i) \tag{18}$$

where $\ell(\phi, \theta; \mathbf{x}_i)$ is a cost loss to be minimized, also called the variational lower bound of on the marginal likelihood of data point $\mathbf{x}_i$. Therefore, the optimization objective function of variational autoencoders is formulated as

$$\min_{\phi,\theta} -D_{KL}(q_\phi(\mathbf{y}|\mathbf{x}_i)||p_\theta(\mathbf{y})) + \mathbb{E}_{q_\phi(\mathbf{y}|\mathbf{x}_i)}[\log p_\theta(\mathbf{x}_i|\mathbf{y})] \tag{19}$$

where ϕ and θ to be optimized are called the variational and generative parameters, respectively. Frequently, we suppose that the hidden variable is a centered isotropic multivariate Gaussian, i.e., $p_\theta(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{I})$, and $p_\theta(\mathbf{x}|\mathbf{y})$ is a multivariate Gaussian for real-value data or Bernoulli for binary data. Simultaneously, the variational posterior is assumed to be a multivariate Gaussian with a diagonal covariance matrix, that is

$$q_\phi(\mathbf{y}|\mathbf{x}_i) = \mathcal{N}\big(\mathbf{y}; \mu_i, \sigma_i^2\mathbf{I}\big). \tag{20}$$

With the aforementioned parameterization trick, a variational autoencoder neural network is trained, and the corresponding hidden variables are computed.

7) Graph Autoencoder: Compared with other autoencoder variants, graph autoencoder includes both the data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ and the graph adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ as an input pair [73], [74]. By encoder function $f(\circ)$, the input two tuples $(\mathbf{X}, \mathbf{A})$ are mapped into a low-dimensional embedding $\mathbf{Y} \in \mathbb{R}^{k\times n}$ with $k \ll d$, while a pairwise decoder model $g(\circ)$ recovers $\mathbf{A}$ with the minimum reconstruction error [75].

In this section, we introduce a local invariant deep nonlinear mapping algorithm, namely a graph regularized autoencoder. With an input data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, this algorithm is to search a feature representation $\mathbf{Y} \in \mathbb{R}^{k\times n}$ with an encoder function $f(\circ)$, i.e., $\mathbf{Y} = f_{\mathbf{W},\mathbf{b}}(\mathbf{X}, \mathbf{A})$, in which k is the number of reduced dimensions and θ is a parameter to be solved. Naturally, there is a decoder function $g(\circ)$ to project the hidden units onto a reconstructed feature space, $\widehat{\mathbf{X}} = g(\mathbf{Y})$.

From a perspective of manifold learning, the local structure of the input feature space is preserved in the embedded low-dimensional representation. Accordingly, the graph regularized autoencoder aims at finding a best low-dimensional embedding in the hidden layer. Its optimization objective problem is defined as the form of

$$\arg\min_{\mathbf{W},\mathbf{b}} ||\mathbf{X} - g(f(\mathbf{X}))||_F^2 + \lambda tr(\mathbf{Y}\mathbf{L}_\mathbf{A}\mathbf{Y}^T) \tag{21}$$

Algorithm 2 Algorithm for Graph Regularized Autoencoders

Require: The original data $\mathbf{X}$, adjacency matrix $\mathbf{A}$, the number l of layers, and the numbers of units in the hidden layers.
Ensure: Weighted matrix $\mathbf{W}$ and bias vector $\mathbf{b}$ of each layer.
1: **for** $i \in \{1, \ldots, n\}$ **do**
2: Update $\mathbf{W}$ and $\mathbf{b}$ by solving Problem (22);
3: Obtain the i-th layer of data representation $\mathbf{Y}_i$;
4: **end for**
5: **return** $\mathbf{W}$ and $\mathbf{b}$.

where $\lambda > 0$ is suggestive of a regularization parameter of the training algorithm and $\mathbf{L}_\mathbf{A}$ is a regularizer of structured learning. In many circumstances, $\mathbf{L}_\mathbf{A}$ is specified as a graph Laplacian, i.e., $\mathbf{L}_\mathbf{A} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$.

As training all layers simultaneously in multiple-layer autoencoders may be stacked, graph regularized autoencoders train the multilayered neural networks layer-by-layer. It is worth pointing out that the above-mentioned optimization problem can be rewritten as

$$\arg\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{n} ||\mathbf{x}_i - g(f(\mathbf{x}_i))||_2^2 + \lambda \sum_{i=1}^{n} tr(\mathbf{y}_i \mathbf{L}_\mathbf{A} \mathbf{y}_i^T) \tag{22}$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$.

With the aforementioned analyses, the procedure for graph regularized autoencoders is summarized in Algorithm II.

8) Convolutional Autoencoder: A convolutional autoencoder is also comprised of encoding and decoding. Compared with other invariants of autoencoder neural networks, it is distinctive in feature generation using convolutional operations, instead of nonlinear activation functions of linear transformations.

For encoding, a convolution neural network architecture [76] is used to produce feature maps, which is widely used in image data processing. It is assumed to contain L layers where the former $L-1$ layers are convolutional, and the Lth layer is fully connected. Here, the hidden feature representation in the lth layer is denoted by $\mathbf{Y}^l$ for $l \in \{1, \ldots, L\}$. Suppose that the lth layer consists of p_l feature filters for any $l \in \{1, \ldots, L\}$. For example, as to the ith filter in the first layer, a convolutional operation with stride length r produces a latent feature map

$$\mathbf{y}^{i,1} = \sigma\left(\mathbf{X} * \mathbf{W}^{i,1} + \mathbf{b}_f^{i,1}\right) \tag{23}$$

where $\sigma(\cdot)$ is a specific nonlinear activation function and $*$ denotes the convolutional operator. In real-world applications, $\sigma(\cdot)$ is frequently represented by a rectified linear unit (ReLU) [77]. Concatenating all feature maps generated by all filters, the set of all feature maps in the lth layer is denoted $\mathbf{Y}^{(l)} = [\mathbf{y}^{(1,l)}, \ldots, \mathbf{y}^{(p_l,l)}]$. For simplicity, we denote $\mathbf{W}^l = [\mathbf{W}^{1,l}, \ldots, \mathbf{W}^{p_l,l}]$ and $\mathbf{b}_f^l = [\mathbf{b}_f^{1,l}, \ldots, \mathbf{b}_f^{p_l,l}]$. Hence, the aforementioned latent feature maps are formulated as the recursive form of

$$\mathbf{Y}^l = \sigma\left(\mathbf{Y}^{l-1} * \mathbf{W}^l + \mathbf{b}_f^l\right) \tag{24}$$

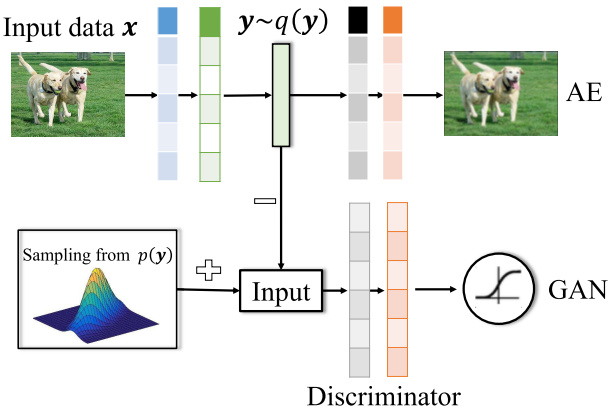

Fig. 5. Architecture of adversarial autoencoder neural network.

where $\mathbf{Y}^1 = \sigma(\mathbf{X} * \mathbf{W}^1 + \mathbf{b}_f^1)$ and $l \in \{1, \ldots, L\}$. As a result, the final hidden layer $\mathbf{Y}^L$ serves as a high-level semantic representation for learning tasks.

For decoding, the hidden low-dimensional feature representation is hierarchically reconstructed to the original feature space. Consequently, the estimated data matrix is represented by

$$\widehat{\mathbf{X}} \triangleq \widehat{\mathbf{X}}^L = g_L(g_{L-1} \cdots (g_1(\mathbf{Y}^L))) \tag{25}$$

with the recursive formulation of

$$\widehat{\mathbf{X}}^l = \sigma\left(\widehat{\mathbf{X}}^{l-1} * \widehat{\mathbf{W}}^l + \widehat{\mathbf{b}}_g^l\right) \tag{26}$$

in which $\widehat{\mathbf{X}}^1 = \sigma(\widehat{\mathbf{Y}}^L * \widehat{\mathbf{W}}^1 + \widehat{\mathbf{b}}_g^1)$ and $l \in \{1, \ldots, L\}$. With the formulated architecture, convolutional autoencoders are represented as the following optimization objective function:

$$\arg\min_{\mathbf{W},\mathbf{b}} \frac{1}{2} ||\mathbf{X} - \widehat{\mathbf{X}}||_F^2 \tag{27}$$

where a unified framework is provided for all variants of autoencoders.

9) Adversarial Autoencoder: Adversarial autoencoders use an "adversarial" mechanism to learn a low-dimensional representation. It is comprised of two main models, namely the generation model and the discriminative model. Frequently, it can be combined with generative adversarial networks [78], [79] and variational autoencoders. The structure of adversarial autoencoder networks [80] is shown in Fig. 5.

Let $\mathbf{x}$ be an input data point and $\mathbf{y}$ be a hidden unit of an adversarial autoencoder. Let $p(\mathbf{y})$ be an arbitrary prior distribution, $p(\mathbf{x}|\mathbf{y})$ be the decoding distribution, and $q(\mathbf{y}|\mathbf{x})$ be the encoding distribution. Meanwhile, we let $p(\mathbf{x})$ be the model distribution and $p_d(\mathbf{x})$ be the data distribution. By $q(\mathbf{y}|\mathbf{x})$, we can define $q(\mathbf{y})$ on a hidden encoding vector as

$$q(\mathbf{y}) = \int_{\mathbf{x}} q(\mathbf{y}|\mathbf{x}) p_d(\mathbf{x}) d\mathbf{x}. \tag{28}$$

Actually, $q(\mathbf{y})$ can be regarded as an aggregated posterior distribution on hidden units. An adversarial autoencoder matches $q(\mathbf{y})$ to $p(\mathbf{y})$ for regularization purposes as well as adversarial sample generators. In addition, the generator of adversarial network is the encoder $q(\mathbf{y}|\mathbf{x})$, which guarantees that the aggregated posterior distribution could confuse the discriminative network between $q(\mathbf{y})$ and $p(\mathbf{y})$.

From the perspective of optimization algorithms, an adversarial autoencoder could be divided into two phases in each mini-batch training process, solved by stochastic gradient descent (SGD). Both the phases are regarded as the reconstruction and the regularization, respectively. In the reconstruction phase, adversarial autoencoder attempts to minimize the reconstruction error of inputs by updating the encoding parameters ϕ and θ, defined by

$$\arg\min_{\theta,\phi} \|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2. \tag{29}$$

There are two tasks in the regularization phase. On one hand, the adversarial network distinguishes the true samples generated by the prior knowledge from the generated samples in the hidden layer through updating its discriminative network. On the other hand, it updates its generator to confuse the discriminative network, formulated as

$$\min_G \max_D \mathrm{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathrm{E}_{\mathbf{y}\sim p(\mathbf{y})}[\log(1 - D(G(\mathbf{y})))] \tag{30}$$

where $G(\circ)$ is the generator and $D(\circ)$ is the discriminator, respectively. Apart from this, $D(\mathbf{x})$ is a neural network to calculate the possibility that a variable $\mathbf{x}$ from the data space is a sampling from positive samples that we attempt to model instead of negative samples. Meanwhile, $G(\mathbf{y})$ is a mapping to project sample $\mathbf{y}$ in the prior $p(\mathbf{y})$ to the data space. The task of $G(\mathbf{y})$ is to infer the discriminative network into a trusted mechanism, such that it generates positive samples as much as possible in the training process. And the training is realized by updating the gradients of $D(\mathbf{x})$ with regard to $\mathbf{x}$.

Concerning the selection of the encoding $q(\mathbf{y}|\mathbf{x})$, adversarial autoencoders have several beneficial choices, such as deterministic, Gaussian posterior [81], and universal approximator posterior. This type of mechanism can generate rewarding models when addressing specific practical scenarios. All in all, an adversarial autoencoder aims at integrating the idea of confrontation into autoencoders, and its merit is the usage of modeling discrete true samples using a continuous infinite hidden variable. In other words, sufficient training data are generated by adversarial autoencoders, which is effective in filling the gap that generative adversarial networks could not produce discrete samples.

10) Residual Autoencoder: He *et al.* [82] proposed a residual learning framework for object recognition. The aim of this method is not to learn unreferenced functions, instead of residual functions about the layer inputs. Due to the multiple downsampling operations, deep residual neural networks may miss some beneficial details. Nevertheless, this type of neural networks can ease the training processing and is effective in exploiting substantially deeper networks. Let $\mathcal{H}(\circ)$ be the underlying function to map the original data point $\mathbf{x}$ into the reconstructed unit $\tilde{\mathbf{x}} = \mathcal{H}(\mathbf{x})$ with the same dimension. It is based on the assumption that any complicated nonlinear functions could be asymptotically approximated by multilayered neural networks with enough layers. Naturally, it is hypothesized that the residual function $\Delta\mathbf{x} \triangleq \mathcal{H}(\mathbf{x}) - \mathbf{x}$ can be

well approximated. Residual networks are to explicitly learn a surrogate of the residual function $\Delta\mathbf{x}$, rather than the mapping $\mathcal{H}(\mathbf{x})$. Combining the weighted matrices $\{\mathbf{W}_i\}_{i=1}^{l}$ to be learned with the layer number l, the residual function is then defined as $\Delta(\mathbf{x}; \{\mathbf{W}_i\}_{i=1}^{l})$. Therefore, the output vector of deep residual neural networks is computed by $\mathbf{y} = \tilde{\mathbf{x}} = \Delta(\mathbf{x}; \{\mathbf{W}_i\}_{i=1}^{l}) + \mathbf{x}$, where

$$\Delta\left(\mathbf{x}; \{\mathbf{W}_i\}_{i=1}^{l}\right) = \mathbf{W}_l^T \sigma\left(\cdots \mathbf{W}_2^T \sigma\left(\mathbf{W}_1^T \mathbf{x}\right)\right). \tag{31}$$

It is noted that the input feature vector $\mathbf{x}$ is assumed to have the same dimension with the output $\Delta(\mathbf{x}; \{\mathbf{W}_i\}_{i=1}^{l})$, which is not necessarily true in varying practical scenarios. Toward this end, a linear projection $\mathbf{W}_s$ by the shortcut connections is performed to uniform the dimensions, with the output defined by

$$\mathbf{y} = \Delta\left(\mathbf{x}; \{\mathbf{W}_i\}_{i=1}^{l}\right) + \mathbf{W}_s^T \mathbf{x}. \tag{32}$$

In a supervised learning, the given class label can serve as an estimated output $\mathbf{y} = \tilde{\mathbf{x}}$, which guides the searching directions of the optimization algorithms. Nonetheless, residual networks are tough to address unsupervised learning tasks due to the absence of class label information.

Residual autoencoders [83] aim at learning an efficient low-dimensional representation in an unsupervised manner. The residual function of an unsupervised task is defined as $\Delta\mathbf{x} = \mathbf{x} - \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is an estimate of the input feature vector $\mathbf{x}$. Therefore, the loss function of residual autoencoders is represented as the canonical form of

$$\mathbb{L} = \|\Delta\mathbf{x} - \Delta\hat{\mathbf{x}}\|_2^2 = \|(\mathbf{x} - \tilde{\mathbf{x}}) - \theta_{\mathbf{W},\mathbf{b}}(\tilde{\mathbf{x}})\|_2^2 \tag{33}$$

where $\Delta\hat{\mathbf{x}} = \theta_{\mathbf{W},\mathbf{b}}(\tilde{\mathbf{x}})$ is the estimated output that residual autoencoders produce. The principle of residual autoencoders is to make $\Delta\hat{\mathbf{x}}$ as close to the expected output as possible.

B. Deconvolutional Network

Deconvolutional network [84] is a framework that allows an unsupervised construction of hierarchical image representations. These feature representations are available to provide effective features for object recognition. Each hierarchical layer collects useful information to yield more complex representations for midlevel and high-level feature learning [58]. Taking image data as an example, it can automatically extract rich features that correspond to midlevel features, such as edge junctions, parallel lines, and curves and high-level features, such as rectangles and semantics. Compared with the convolutional autoencoders, each layer in deconvolutional networks is top-down. It manages to reconstruct the input data by a sum over convolutions of the feature maps with the learned filters.

The architecture of deconvolutional neural networks is shown in Fig. 6. Each deconvolutional network consists of pairs of convolution and deconvolution operations. The former is to generate features using convolution kernels, while the latter aims to reconstruct the input data using feature filters. Naturally, these two operations correspond to the encoder and the decoder of an autoencoder neural network, respectively.

As an example of single deconvolutional network, let the hidden variable be $\mathbf{y}^i$ consisting of c channels $\mathbf{y}_1^i, \ldots, \mathbf{y}_c^i$.

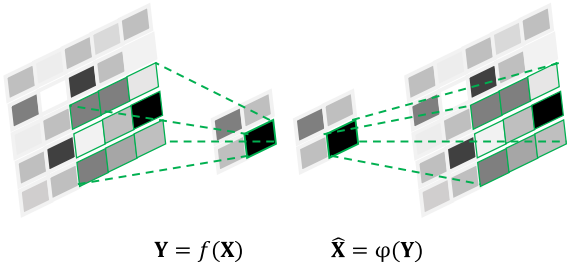

Fig. 6. Illustration of convolution and deconvolution.

In this network, each channel is represented as a linear sum of latent feature maps $\mathbf{z}_j^i$ convolved with filters $\mathbf{W}_{j,k}$, defined by

$$\mathbf{y}_k^i = \sum_{j=1}^{c'} \mathbf{z}_j^i * \mathbf{W}_{j,k} \tag{34}$$

where $k \in \{1, \ldots, c\}$ and c' is the number of latent feature maps. It is worth noting that c' is jointly determined by the input image size and filter size. For instance, there are $(W - h + 1) \times (H - h + 1)$ feature maps for an image of resolution $W \times H$ and a filter of $h \times h$ size. Accordingly, the optimization problem is represented by

$$\arg\min_{\mathbf{y}^i} \sum_{k=1}^{c} \left\| \mathbf{y}_k^i - \sum_{j=1}^{c'} \mathbf{z}_j^i * \mathbf{W}_{j,k} \right\|_2^2 + \lambda \sum_{j=1}^{c'} |\mathbf{z}_j^i|^p \tag{35}$$

where λ is a regularization parameter to keep a tradeoff between the first fitting accuracy (ACC) and the second regularization term. Besides, this regularization term is added to provide a well-defined solution to the underdetermined optimization problem. Then, the deconvolutional networks assume that Gaussian noises are imposed on the reconstruction term [85] and encourage certain properties of the learned feature maps by p-norm. As an example, λ balances the relative contributions of the reconstruction of $\mathbf{y}^i$ and the sparsity of the feature maps $\mathbf{z}_j^i$ when fixing $p = 1$. Finally, the model of deconvolutional networks is top-down. It is suggested from this network that an image is well restored when the latent feature maps are available. Unlike sparse autoencoders or deep belief nets, there is a tailored mechanism for feature map generation, whose effectiveness has been validated in varying practical applications.

C. Restricted Boltzmann Machine

A Boltzmann machine is a kind of stochastic recurrent neural network constructed by Ackley and Hinton [86] in 1985. The Boltzmann machine is named from the Boltzmann distribution in statistical mechanics. RBM is regarded as a special topological structure of the Boltzmann machine, proposed by Hinton *et al.* [87], [88] in 1986. The term "restricted" refers to the model constrained with the bipartite graph, which is indicative of the disconnectivity of neurons in the same layer. Therefore, RBMs consume less computational resources than traditional ones [89].

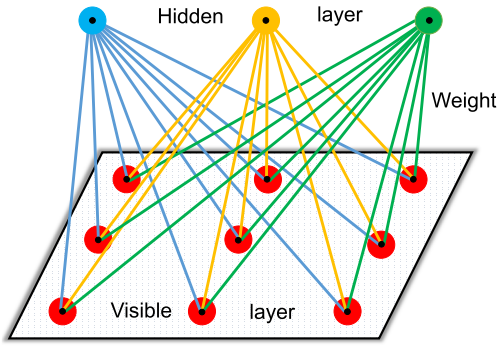

Fig. 7. Framework of RBM.

RBM has a visible layer in the input layer and a hidden layer for feature representations [90], [91], and its architecture is shown in Fig. 7. It is comprised of visible units and hidden units, where the former is the input data points and the latter corresponds to feature representations. Actually, RBMs come with binary-valued visible and hidden variables, aiming at modeling a probability distribution of the input data points.

Let $\mathbf{x}_i$ and $\mathbf{y}_j$ be the binary states of visible neuron i and hidden neuron j. An energy function $\mathrm{E}(\mathbf{x}, \mathbf{y})$ of the visible and hidden units $(\mathbf{x}, \mathbf{y})$ is formulated as

$$\mathrm{E}(\mathbf{x}, \mathbf{y}) = -\sum_i \mathbf{a}_i \mathbf{x}_i - \sum_j \mathbf{b}_j \mathbf{y}_j - \sum_i \sum_j \mathbf{x}_i \mathbf{y}_j \mathbf{W}_{ij} \tag{36}$$

where $\mathbf{a}_i$ and $\mathbf{b}_j$ are the biases and $\mathbf{W}_{ij}$ is the weighted parameter between $\mathbf{x}_i$ and $\mathbf{y}_j$. With the form of matrix representation, the aforementioned energy function is written as

$$\mathrm{E}(\mathbf{x}, \mathbf{y}) = -\mathbf{a}^T \mathbf{x} - \mathbf{b}^T \mathbf{y} - \mathbf{y}^T \mathbf{W} \mathbf{x}. \tag{37}$$

Alternatively, there is another energy function $\mathrm{E}(\mathbf{x}, \mathbf{y})$ to assign a probability to the pair $(\mathbf{x}, \mathbf{y})$ of visible and hidden vectors, defined by

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} e^{-\mathrm{E}(\mathbf{x}, \mathbf{y})} \tag{38}$$

where Z is the partition function that is formalized by summing over all possible pairs of visible and hidden vectors, that is

$$Z = \sum_{\mathbf{x}} \sum_{\mathbf{y}} e^{-\mathrm{E}(\mathbf{x}, \mathbf{y})}. \tag{39}$$

Besides, the marginal probability $p(\mathbf{x})$ of $p(\mathbf{x}, \mathbf{y})$ with respect to visible variable $\mathbf{x}$ is defined as

$$p(\mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{y}} e^{-\mathrm{E}(\mathbf{x}, \mathbf{y})}. \tag{40}$$

Since RBMs are structured with bipartite graphs, which is suggestive of no intralayer connection in the visible or hidden layers. Therefore, the conditional probabilities $p(\mathbf{x}|\mathbf{y})$ and $p(\mathbf{y}|\mathbf{x})$ are written as

$$p(\mathbf{x}|\mathbf{y}) = \prod_i p(\mathbf{x}_i|\mathbf{y}) \text{ and } p(\mathbf{y}|\mathbf{x}) = \prod_i p(\mathbf{y}_i|\mathbf{x}). \tag{41}$$

In the training process, RBMs are to search the optimal $\mathbf{W}$ value while maximizing the product of probabilities of a given training set $\mathbf{X}$, that is

$$\arg\max_{\mathbf{W}} \prod_{\mathbf{x} \in \mathbf{X}} p(\mathbf{x}) \tag{42}$$

which is equivalently written as

$$\arg\max_{\mathbf{W}} \mathbf{E}\left[\sum_{\mathbf{x} \in \mathbf{X}} \log p(\mathbf{x})\right]. \tag{43}$$

A widely used optimization algorithm for the above-mentioned problem is Gibbs sampling-based gradient descent method [92]. It is observed that $\mathrm{E}(\mathbf{x}, \mathbf{y})$ is a quadratic form, and hence, the derivative of the log probability of a training vector with regard to a weight is computed by

$$\frac{\partial \log p(\mathbf{x})}{\partial \mathbf{W}_{ij}} = \langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{data}} - \langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{model}} \tag{44}$$

where $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{data}}$ and $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{model}}$ represent the expectations under the distribution defined by the subscript that follows. The updated weights using stochastic descent method is solved by

$$\Delta \mathbf{W}_{ij} = \alpha(\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{data}} - \langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{model}}) \tag{45}$$

where $\alpha > 0$ is the learning rate. It is noted that for any visible unit $\mathbf{x}$, we have

$$p(\mathbf{x}_i = 1|\mathbf{y}) = \sigma\left(\mathbf{a}_i + \sum_j \mathbf{y}_j \mathbf{W}_{ij}\right) \tag{46}$$

where $\sigma(\circ)$ is the logistic sigmoid function. Analogously, for any hidden unit $\mathbf{y}$, we know

$$p(\mathbf{y}_j = 1|\mathbf{x}) = \sigma\left(\mathbf{b}_j + \sum_i \mathbf{x}_i \mathbf{W}_{ij}\right). \tag{47}$$

It is suggested that $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{data}}$ is easy to compute using the aforementioned formulations. Unfortunately, it is tough to obtain an unbiased sample of $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{model}}$, which would consume a large amount of computational cost if performing alternating Gibbs sampling by starting at any stochastic state of the visible units. Toward this end, Hinton [93] proposed a faster optimization method in 2002, replacing $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{model}}$ with $\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{recon}}$, indicating that the optimization problem is rewritten as

$$\Delta \mathbf{W}_{ij} = \alpha(\langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{data}} - \langle \mathbf{x}_i \mathbf{y}_j \rangle_{\text{recon}}). \tag{48}$$

Overall, RBMs have been capturing growing attention in recent years. Nevertheless, how to improve the training efficiency is still an open issue.

D. Deep Belief Nets

As a type representative of probabilistic generative model, deep belief nets were proposed by Hinton *et al.* [9] in 2006. This network consists of multiple layers of stochastic and latent units, where the latent variables are frequently binary-valued. The top two layers are connected with an undirected

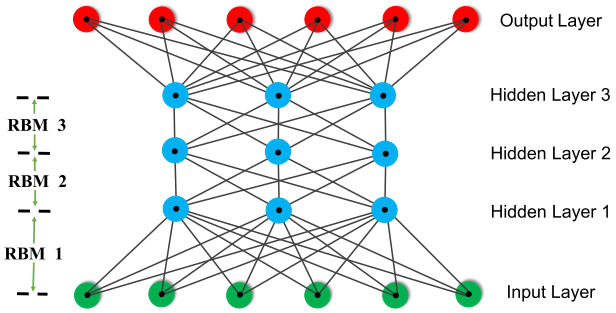

Fig. 8. Illustration of the architecture of deep belief nets.

and symmetric graph, constructing an associative memory. The architecture of deep belief nets is demonstrated in Fig. 8.

It is observed from Fig. 8 that deep belief nets are treated as one type of RBMs. The crucial procedure behind deep belief nets is to optimize $\mathbf{W}$ with the learned conditional distribution $p(\mathbf{x}|\mathbf{y}, \mathbf{W})$ and prior distribution $p(\mathbf{y}|\mathbf{W})$ from an RBM. Therefore, the probability of generating a visible unit $\mathbf{x}$ is formalized as

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{W}) p(\mathbf{x}|\mathbf{y}, \mathbf{W}). \tag{49}$$

While fixing $p(\mathbf{x}|\mathbf{y}, \mathbf{W})$, the weighted matrix $\mathbf{W}$ is updated by replacing $p(\mathbf{y}|\mathbf{W})$ with a better model of the aggregated posterior distribution over the hidden variables. Actually, the complementary prior at each layer guarantees that the posterior distribution could be factorial.

For feature representation and dimensionality reduction, deep belief nets are regarded as a nonlinear model as well. Indeed, deep belief nets act as a model that could produce complex nonlinear features in the last layer, in which these features are justified by minimal reconstruction error to the visible units.

III. Experiments Analyses

Extensive experiments are conducted to provide a fair comparison for the aforementioned unsupervised deep feature representation methods. The selective experimental databases are characterized as follows.

A. Data Sets

CNAE is a collection of 1080 text documents with nine categories from Brazilian companies for free text business characterizations.[1] Each document is represented as an 857-D feature vector, where each feature is the frequency of some certain word.

20Newsgroups consists of 18821 text documents, partitioned nearly evenly across 20 newsgroups, leading to 20 categories in total.[2] Each document corresponds to one different topic, forming a 61 188-D feature vector with word frequencies.

[1]https://archive.ics.uci.edu/ml/datasets/CNAE-9
[2]http://qwone.com/ jason/20Newsgroups/

TABLE I
Brief Description of the Tested Data Sets

ID	datasets	# samples	# features	# classes
1	CNAE	1,080	856	9
2	20Newsgroups	18,821	61,188	20
3	Reuters	21,578	18,933	65
4	RCV1	9,625	29,992	4
5	TOX	171	5,748	4
6	HAR	10,299	561	6
7	TDT2	9,394	36,771	30
8	DBWorld	64	4,702	2

Reuters is a subset of the database Reuters21578 of text documents. This data set is preprocessed as 8293 documents in 65 categories by discarding those data with multiple category labels.[3] Each document contains 18 933 distinct terms, corresponding to a feature vector.

RCV1 is cut from the distinguished RCV text document corpus, containing 9625 documents.[4] Each document is represented by a 29 992-D feature vector of distinct words.

TOX is a gene database of 171 genetic toxicology (mutagenicity) text documents with four categories from expert peer review of open scientific.[5] Each document is represented as a 5748-D feature vector.

HAR is a database from the recordings of 30 subjects performing activities of daily living.[6] The data set is composed of 10 299 documents in six categories, where each document is extracted as a 561-D feature vector.

TDT2 is a subset of NIST topic detection and tracking corpus. In this subset, those documents occurring two or more categories were eliminated, and only the top 30 categories were preserved, leaving totally 9394 documents of 36 771-D features.[7]

DBWorld is a data collection of 64 e-mails manually produced from DBWorld mailing list with two categories. Each document is represented as a 4702-D feature vector, in which each feature value corresponds to a precise word or stem in the entire data set vocabulary.[8]

With the aforementioned descriptions, a brief characterization to all tested data sets is summarized in Table I.

B. Parameter Settings

For all compared feature representation algorithms, the learning rate is fixed as $\alpha = 0.1$. As to those multilayered neural networks, the numbers of layers are set as 3, and the numbers of hidden variables in the first and second layers are tuned as 400 and 500, respectively. In order to provide a fair comparison, the number of units in the last hidden layer ranges in $\{20, 30, \ldots, 200\}$. Meanwhile, the numbers of epochs for all algorithms are fixed as 50, and the batch sizes are set as 100. Besides, the default settings are adopted for other algorithmic parameters.

[3]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html
[4]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html
[5]https://toxnet.nlm.nih.gov/newtoxnet/genetox.htm
[6]http://archive.ics.uci.edu/ml/datasets
[7]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html
[8]http://archive.ics.uci.edu/ml/datasets/DBWorld+e-mails

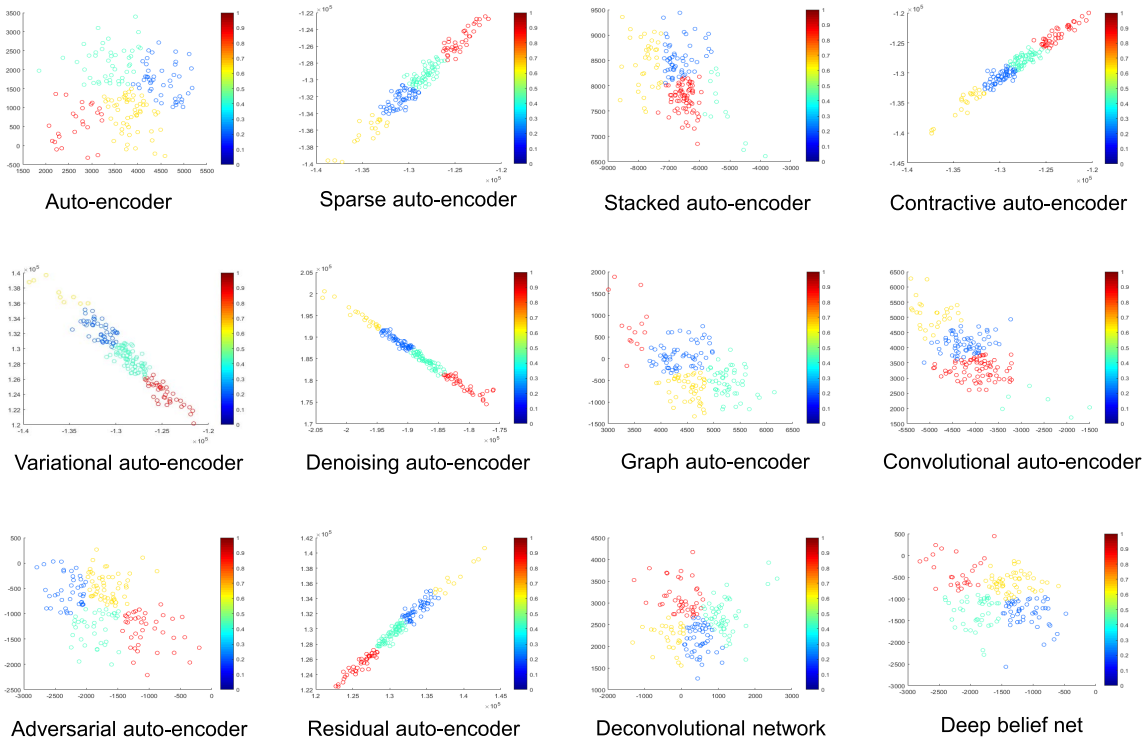

Fig. 9. Illustration of varying unsupervised deep feature representation methods on the data set TOX with four classes, where the dimensionality-reduced samples from the same class are marked with the same color. Naturally, the better method is that the samples having the same color are clustered together, while ones having different colors are scattered.

TABLE II

CLUSTERING ACC (MEAN% ± STD%) WITH DIFFERENT UNSUPERVISED FEATURE REPRESENTATION METHODS. THE BEST RESULTS ARE MARKED IN BOLD (THE HIGHER THE BETTER)

dataset/method	CNAE	20Newsgroups	Reuters	RCV1	TOX	HAR	TDT2	DBWorld
baseline	49.7±2.2	31.3±0.9	70.2±1.1	66.1±7.8	44.2±1.8	63.5±2.7	58.1±0.4	70.9±9.2
Auto-encoder	43.3±2.6	43.6±0.4	71.5±1.3	61.4±4.2	38.9±0.5	64.4±3.3	65.9±3.9	82.8±2.3
Sparse auto-encoder	46.4±1.2	**50.3±0.6**	72.9±3.1	77.3±3.4	38.4±0.4	66.1±1.7	68.8±2.1	73.1±1.8
Stacked auto-encoder	39.8±2.4	49.1±1.5	73.8±2.6	62.4±4.2	38.7±0.4	66.3±0.9	69.5±3.0	75.1±4.6
Contractive auto-encoder	47.8±3.2	38.7±2.2	**74.6±2.2**	69.5±4.2	41.0±2.4	64.6±2.9	66.9±3.7	75.0±4.5
Variational auto-encoder	54.7±6.2	42.3±0.3	68.2±3.7	70.6±4.2	46.5±1.5	70.8±3.2	**73.6±2.9**	79.7±4.9
Denoising auto-encoder	**55.2±0.9**	41.4±1.2	70.3±3.1	**85.2±4.9**	**48.8±1.2**	**71.3±4.0**	66.5±4.4	**85.9±5.6**
Graph auto-encoder	50.3±1.5	35.6±1.8	64.5±2.0	60.6±2.3	42.4±2.3	61.4±2.2	62.8±1.2	66.7±2.1
Convolutional auto-encoder	49.5±2.1	33.6±1.4	63.8±3.6	63.7±3.4	39.2±1.8	68.7±3.6	55.4±2.7	68.7±3.5
Adversarial auto-encoder	51.4±2.1	35.9±1.9	68.9±3.2	69.5±2.3	43.5±3.1	65.9±2.7	57.3±1.1	64.0±2.3
Residual auto-encoder	50.5±1.8	41.1±2.2	72.6±3.6	68.7±2.4	44.3±2.8	59.8±2.5	63.1±1.3	70.2±4.7
Deconvolutional network	46.8±2.1	39.7±1.8	66.4±2.5	72.6±3.1	48.7±1.9	61.0±1.7	64.7±2.2	67.4±3.8
Restricted Boltzmann Machine	50.4±0.7	32.3±1.3	69.8±2.7	67.3±2.7	40.9±2.2	63.8±3.6	59.3±2.1	72.3±2.2
Deep belief nets	48.7±0.3	36.1±0.2	73.2±2.9	67.5±1.2	41.4±3.7	65.4±2.1	66.9±4.0	71.6±2.4

All compared unsupervised methods are evaluated by their clustering performances of the feature representations in the last hidden layer. In these circumstances, the K-means method is employed for the clustering performance specification. Due to initialization sensitivity, the mean and standard deviation of 20 repeated experiments are reported.

C. Evaluation Metrics

Due to the absence of class label information, the clustering evaluation for unsupervised learning tasks is a tough issue. Toward this end, we adopt three evaluation metrics to assess the clustering performance, including clustering ACC, normalized mutual information (NMI), and adjusted rand index (ARI). Denote the set of data points as $\{\mathbf{x}_i\}_{i=1}^n$. Suppose that $\{\mathbf{l}_i\}_{i=1}^n$ is the given ground truth and $\{\hat{\mathbf{l}}_i\}_{i=1}^n$ is the predictive clustering labels. The clustering ACC is calculated by

$$\text{ACC} = \frac{\sum_{i=1}^n \delta(\mathbf{l}_i, \text{map}(\hat{\mathbf{l}}_i))}{n} \tag{50}$$

where map$(\circ)$ is a permutation mapping that best matches the predictive clustering labels to ground truths, and δ is a Dirac delta function, i.e., $\delta(\mathbf{l}_i, \mathbf{l}_j) = 1$ if $\mathbf{l}_i = \mathbf{l}_j$; otherwise, $\delta(\mathbf{l}_i, \mathbf{l}_j) = 0$.

TABLE III
NMI (MEAN% ± STD%) WITH DIFFERENT UNSUPERVISED FEATURE REPRESENTATION METHODS. THE BEST RESULTS ARE MARKED IN BOLD (THE HIGHER THE BETTER)

dataset/method	CNAE	20Newsgroups	Reuters	RCV1	TOX	HAR	TDT2	DBWorld
baseline	41.3±1.8	1.5±0.1	29.6±1.3	28.7±5.0	13.6±2.2	65.4±3.6	28.6±0.6	22.1±2.7
Auto-encoder	45.5±1.8	3.7±0.4	30.2±1.1	29.2±0.5	15.3±1.2	63.9±2.9	33.9±2.6	40.3±2.6
Sparse auto-encoder	46.6±2.4	**7.4±0.2**	31.6±3.2	33.4±0.9	14.5±0.8	65.8±0.9	31.2±1.1	29.0±1.4
Stacked auto-encoder	39.3±1.2	7.2±0.2	33.5±2.6	26.6±1.2	13.2±1.5	65.5±1.3	32.2±1.3	27.2±2.8
Contractive auto-encoder	52.4±2.6	5.8±0.3	31.5±2.6	30.7±1.3	12.5±1.4	63.9±2.2	31.9±1.8	21.6±0.9
Variational auto-encoder	**58.6±5.2**	2.3±0.2	28.8±3.1	27.3±0.7	**19.3±2.5**	70.8±4.1	37.1±1.9	27.1±2.7
Denoising auto-encoder	40.5±2.2	3.4±0.8	30.4±3.3	34.6±0.4	14.4±0.9	**71.9±2.9**	34.3±3.9	43.2±3.8
Graph auto-encoder	43.7±2.8	3.3±0.6	28.1±1.4	28.5±2.1	16.7±2.2	58.3±1.8	33.2±2.1	37.8±2.0
Convolutional auto-encoder	46.6±2.5	4.7±0.3	34.3±2.2	29.4±0.9	15.1±1.5	60.4±4.3	29.8±1.8	26.4±1.7
Adversarial auto-encoder	43.3±1.5	2.8±1.3	29.1±2.1	30.8±1.3	15.8±2.5	67.8±3.0	**43.6±1.2**	**44.3±2.5**
Residual auto-encoder	44.6±2.3	4.5±0.5	32.7±1.8	35.6±0.9	14.2±0.6	59.9±2.8	27.4±1.6	32.5±3.3
Deconvolutional network	47.2±3.4	2.6±0.4	**36.5±2.4**	31.4±0.6	12.9±1.2	62.4±2.7	30.3±1.7	23.3±1.1
Restricted Boltzmann Machine	45.8±1.6	2.3±0.7	27.6±1.1	33.5±0.9	14.9±1.8	68.7±2.2	37.5±1.4	37.6±1.9
Deep belief nets	50.1±2.2	4.2±0.2	28.7±1.6	**39.4±0.6**	14.6±2.7	70.6±2.5	38.7±2.3	40.6±2.4

TABLE IV
ARI (MEAN% ± STD%) WITH DIFFERENT UNSUPERVISED FEATURE REPRESENTATION METHODS. THE BEST RESULTS ARE MARKED IN BOLD (THE HIGHER THE BETTER)

dataset/method	CNAE	20Newsgroups	Reuters	RCV1	TOX	HAR	TDT2	DBWorld
baseline	23.8±3.2	0.1±0.1	17.5±2.1	19.6±1.2	11.6±1.7	52.1±4.9	10.2±0.2	19.9±1.5
Auto-encoder	27.2±2.9	2.6±0.2	18.4±0.5	20.3±1.3	14.6±0.5	54.4±3.2	15.7±2.5	42.2±1.8
Sparse auto-encoder	31.4±0.5	**8.6±0.2**	21.6±2.7	22.1±0.4	12.1±0.4	52.1±1.5	22.9±2.2	24.8±3.7
Stacked auto-encoder	28.2±0.4	7.4±0.4	20.1±1.4	27.8±1.1	19.6±0.7	52.8±2.4	20.2±2.7	20.7±4.5
Contractive auto-encoder	**34.6±3.3**	2.4±0.2	19.3±0.8	19.6±0.8	12.3±1.3	55.1±1.7	20.7±2.6	23.8±2.3
Variational auto-encoder	31.2±4.5	3.9±0.1	17.2±2.3	25.7±0.6	14.9±2.6	60.1±2.9	**26.1±2.9**	34.2±3.2
Denoising auto-encoder	24.6±0.7	2.2±0.2	18.1±1.7	20.3±0.8	**20.2±0.8**	60.2±3.5	15.2±2.8	**50.9±4.1**
Graph auto-encoder	28.9±1.5	5.5±0.1	16.4±1.1	19.2±0.7	10.7±2.3	61.6±2.5	17.8±1.6	27.2±0.8
Convolutional auto-encoder	29.2±2.2	1.3±0.2	21.5±1.9	18.6±0.8	12.5±1.6	57.8±3.1	14.4±2.3	28.4±1.6
Adversarial auto-encoder	23.3±0.6	1.4±0.3	**27.7±1.2**	**32.4±0.5**	11.4±3.4	56.1±2.9	12.6±2.3	23.8±1.3
Residual auto-encoder	25.9±0.8	2.2±0.4	18.3±1.2	20.1±1.2	13.7±0.7	59.3±2.6	19.4±1.4	27.6±2.2
Deconvolutional network	29.8±1.3	1.8±0.2	19.8±0.9	18.7±0.8	11.2±0.4	60.2±1.8	13.3±0.6	31.5±1.3
Restricted Boltzmann Machine	27.5±0.4	2.3±0.1	26.4±1.9	22.7±0.5	16.2±0.4	57.2±1.1	16.2±0.4	25.4±1.2
Deep belief nets	30.3±2.1	4.3±0.2	26.9±1.5	24.1±2.2	15.8±2.0	**63.7±1.4**	15.9±0.7	28.0±0.6

Given two random variables $\mathbf{P}$ and $\mathbf{Q}$, their NMI is computed by

$$\mathrm{NMI}(\mathbf{P},\mathbf{Q}) = \frac{I(\mathbf{P};\mathbf{Q})}{\sqrt{H(\mathbf{P})H(\mathbf{Q})}} \tag{51}$$

where $I(\circ;\circ)$ is the mutual information and $H(\circ)$ is the information entropy. Let the predictive clustering result be $\widetilde{\mathbb{C}} = \{\widetilde{\mathbf{C}}_i\}_{i=1}^{\widetilde{c}}$ and the ground truth be $\mathbb{C} = \{\mathbf{C}_j\}_{j=1}^{c}$. Herein, NMI can be defined by

$$\mathrm{NMI} = \frac{\sum_{i=1}^{\widetilde{c}}\sum_{j=1}^{c}|\widetilde{\mathbf{C}}_i \cap \mathbf{C}_j|\log\frac{n|\widetilde{\mathbf{C}}_i \cap \mathbf{C}_j|}{|\widetilde{\mathbf{C}}_i||\mathbf{C}_j|}}{\sqrt{\left(\sum_{i=1}^{\widetilde{c}}|\widetilde{\mathbf{C}}_i|\log\frac{|\widetilde{\mathbf{C}}_i|}{n}\right)\left(\sum_{j=1}^{c}|\mathbf{C}_j|\log\frac{|\mathbf{C}_j|}{n}\right)}}. \tag{52}$$

ARI is a distance metric that can measure the similarity between two data clusterings. Denote $n_{ij} = |\widetilde{\mathbf{C}}_i \cap \mathbf{C}_j|$, $a_i = \sum_{j=1}^{c} n_{ij}$, and $b_j = \sum_{i=1}^{\widetilde{c}} n_{ij}$ for any $i \in \{1,\ldots,\widetilde{c}\}$ and $j \in \{1,\ldots,c\}$. Consequently, ARI can be computed with

$$\mathrm{ARI} = \frac{\Sigma_{ij}\binom{n_{ij}}{2} - \left[\Sigma_i\binom{a_i}{2}\Sigma_j\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\Sigma_i\binom{a_i}{2} + \Sigma_j\binom{b_j}{2}\right] - \left[\Sigma_i\binom{a_i}{2}\Sigma_j\binom{b_j}{2}\right]/\binom{n}{2}}. \tag{53}$$

These three evaluation metrics are positively related to the clustering performance. In other words, higher values of ACC, NMI, and ARI are suggestive of a better clustering result as well as feature representation performance.

D. Experimental Visualizations

In this section, an experimental visualization is reported to provide an intuitive demonstration of low-dimensional representations learned by different deep neural networks. Taking the data set TOX as an example, the number of reduced dimensions is set as 2. Varying unsupervised deep feature representation methods are visualized in Fig. 9.

It is observed Fig. 9 that varying unsupervised deep feature representation approaches come with different visualization performances. The best feature representation, in theory, is a mapping with maximal between-cluster distance and minimum within-cluster distance simultaneously, where the circles marked with the same color can be regarded as one cluster. From this perspective, almost all tested feature representation methods perform favorably because all data points are projected onto a low-dimensional space with a large margin between the circles with different colors. Nonetheless, the learned low-dimensional distribution by different methods differs. As an example, the sparse autoencoder projects the original data points onto a nearly linear distribution, and so are contractive, variational, and residual autoencoders.

E. Experimental Results

In this section, the compared unsupervised deep feature representation approaches are tested on eight publicly available data sets of text documents. Meanwhile, we report their respective performances in terms of data clustering, as demonstrated in Tables II–IV.

From Tables II–IV, we have the following observations. In the first place, most of the unsupervised deep feature representation methods are capable of learning an effective low-dimensional representation, which is validated by the fact that the performance of the learned representation outperforms the baseline method that uses all original features for clustering. Then, the feature representation methods have a positive influence on the improvement of learning performance. As an example, the best compared method exhibits a definite superiority over the baseline method in the data sets, such as 20Newsgroups, RCV1, TDT2, and DBWorld. Certainly, feature representation techniques may fail to improve the clustering performance in some specific situations. For instance, stacked autoencoders work unfavorable in the data set CNAE, and deconvolutional networks perform undesirable in the data set Reuters. The learning performance fluctuation is partly due to the fixed parameter settings of tested neural networks for a fair experimental comparison. Finally, the majority of unsupervised feature representation methods could exceed the baseline. From the perspective of structured learning, certain feature representation method attempts to learn an effective low-dimensional embedding with specific patterns. In particular, sparse autoencoders favor the sparsity of the activated neurons in hidden layers, while adversarial autoencoders stress the robustness of the learned low-dimensional representation.

IV. Conclusion and Future Work

In this paper, we presented an overview of a number of unsupervised deep feature representation methods. These reviewed methods included autoencoders and their variants, deconvolutional networks, RBMs, and deep belief nets. In addition, extensive comparative experiments were conducted on eight publicly available data sets of text documents. These experiments provided a comprehensive comparison for all tested feature representation approaches, aiming at exhibiting respective benefits of varying deep feature learning methods in text clustering. It is expected that this paper would provide some insights and enlightenments for who is highly interested in unsupervised deep learning. In our future work, we will explore more efficient unsupervised deep learning methods with given data-driven scenes.

References

[1] A. Farasat, G. Gross, R. Nagi, and A. G. Nikolaev, "Social network analysis with data fusion," *IEEE Trans. Comput. Social Syst.*, vol. 3, no. 2, pp. 88–99, Jun. 2016.

[2] A. Argyriou, T. Evgeniou, and M. Pontil,"Multi-task feature learning," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2006, pp. 41–48.

[3] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 394–410, Mar. 2007.

[4] N. Foroutan and A. Hamzeh, "Discovering the hidden structure of a social network: A semi supervised approach," *IEEE Trans. Comput. Social Syst.*, vol. 4, no. 1, pp. 14–25, Mar. 2017.

[5] Y. Xiao, X. Li, H. Wang, M. Xu, and Y. Liu, "3-HBP: A three-level hidden bayesian link prediction model in social networks," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 2, pp. 430–443, Jun. 2018.

[6] J. G. Dy and C. E. Brodley, "Feature selection for unsupervised learning," *J. Mach. Learn. Res.*, vol. 5, pp. 845–889, Aug. 2004.

[7] S. Wang and W. Zhu, "Sparse graph embedding unsupervised feature selection," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 3, pp. 329–341, Mar. 2018.

[8] S. Wang and W. Guo, "Sparse multigraph embedding for multimodal feature representation," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1454–1466, Jul. 2017.

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

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

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

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 689–696.

[14] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn.*, Jun. 2013, pp. 1139–1147.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process Syst.*, Jun. 2015, pp. 91–99.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[18] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2016, pp. 21–37.

[19] L. Deng *et al.*, "Recent advances in deep learning for speech research at microsoft," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2013, pp. 8604–8608.

[20] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 2014, pp. 1562–1566.

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

[22] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2015, pp. 373–382.

[23] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Sep. 2015, pp. 649–657.

[24] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.

[26] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4694–4702.

[27] D.-A. Huang *et al.*, "What makes a video a video: Analyzing temporal information in video understanding models and datasets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7366–7375.

[28] D.-A. Huang, S. Buch, L. Dery, A. Garg, L. Fei-Fei, and J. C. Niebles, "Finding 'it': Weakly-supervised reference-aware visual grounding in instructional videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5948–5957.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017.

[30] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[31] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 1002–1014, Apr. 2018.

[32] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2007, pp. 873–880.

[33] N. Srivastava and R. Salakhutdinov, "Learning representations for multimodal data with deep belief nets," in *Proc. Int. Conf. Mach. Learn. Workshop*, Jul. 2012, pp. 79–87.

[34] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[35] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Jun. 2011, pp. 215–223.

[36] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 742–769, Feb. 2019.

[37] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2016, pp. 1–11.

[38] V. Kurková and M. Sanguineti, "Probabilistic lower bounds for approximation by shallow perceptron networks," *Neural Netw.*, vol. 91, pp. 34–41, Jul. 2017.

[39] S. Lin, "Limitations of shallow nets approximation," *Neural Netw.*, vol. 94, pp. 96–102, Oct. 2017.

[40] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, Jan. 2019.

[41] E. Sangineto, M. Nabi, D. Culibrk, and N. Sebe, "Self paced deep learning for weakly supervised object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 3, pp. 712–725, Mar. 2019.

[42] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 394–405, Feb. 2019.

[43] J. Yang, W. Xiong, S. Li, and C. Xu, "Learning structured and non-redundant representations with deep neural networks," *Pattern Recognit.*, vol. 86, pp. 224–235, Feb. 2019.

[44] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, "Deepcrack: Learning hierarchical convolutional features for crack detection," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1498–1512, Mar. 2019.

[45] I. González-Díaz, J. Benois-Pineau, J.-P. Domenger, D. Cattaert, and A. de Rugy, "Perceptually-guided deep neural networks for ego-action prediction: Object grasping," *Pattern Recognit.*, vol. 88, pp. 223–235, Apr. 2019.

[46] Y. Niu, Z. Lu, J.-R. Wen, T. Xiang, and S.-F. Chang, "Multi-modal multi-scale deep learning for large-scale image annotation," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1720–1731, Mar. 2019.

[47] C. Jia, M. Shao, S. Li, H. Zhao, and Y. Fu, "Stacked denoising tensor auto-encoder for action recognition with spatiotemporal corruptions," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 1878–1887, Apr. 2018.

[48] M. A. Kiasari, D. S. Moirangthem, and M. Lee, "Coupled generative adversarial stacked auto-encoder: Cogasa," *Neural Netw.*, vol. 100, pp. 1–9, Apr. 2018.

[49] M. Li, J. Hu, C. Xia, and Y. Zhang, "An implementation of picture compression with A CNN-based auto-encoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2018, pp. 2543–2546.

[50] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.

[51] D. Chen, J. Lv, and Z. Yi, "Graph regularized restricted boltzmann machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2651–2659, Jun. 2018.

[52] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, "Equivalence of restricted boltzmann machines and tensor network states," *Phys. Rev. B, Condens. Matter*, vol. 97, no. 8, Feb. 2018, Art. no. 085104.

[53] L.-W. Kim, "Deepx: Deep learning accelerator for restricted boltzmann machine artificial neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1441–1453, May 2018.

[54] L. Ning, R. Pittman, and X. Shen, "Lcd: A fast contrastive divergence based algorithm for restricted Boltzmann machine," *Neural Netw.*, vol. 108, pp. 399–410, Dec. 2018.

[55] Y. Chen, H. Gao, L. Cai, M. Shi, D. Shen, and S. Ji, "Voxel deconvolutional networks for 3D brain image labeling," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 1226–1234.

[56] J. Liu, Y. Wang, Y. Li, J. Fu, J. Li, and H. Lu, "Collaborative deconvolutional neural networks for joint depth estimation and semantic segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5655–5666, Nov. 2018.

[57] X. Yu and F. Porikli, "Imagining the unimaginable faces by deconvolutional networks," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2747–2761, Jun. 2018.

[58] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2018–2025.

[59] J. Qiao, G. Wang, X. Li, and W. Li, "A self-organizing deep belief network for nonlinear system modeling," *Appl. Soft Comput.*, vol. 65, pp. 170–183, Apr. 2018.

[60] S. N. Tran and A. S. d'A. Garcez, "Deep logic networks: Inserting and extracting knowledge from deep belief networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 246–258, Feb. 2018.

[61] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A cost-sensitive deep belief network for imbalanced classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 109–122, Jan. 2018.

[62] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Dec. 2013, pp. 8595–8598.

[63] K. Chen, J. Hu, and J. He, "A framework for automatically extracting overvoltage features based on sparse autoencoder," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 594–604, Mar. 2018.

[64] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1930–1943, Aug. 2013.

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

[66] Y. I. Manin, "Frobenius manifolds, quantum cohomology, and moduli spaces," *Amer. Math. Soc.*, vol. 38, no. 1999, p. 101–108, 2007.

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

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

[69] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 811–820.

[70] D. Slepian, "The one-sided barrier problem for gaussian noise," *Bell Syst. Tech. J.*, vol. 41, no. 2, pp. 463–501, Mar. 1962.

[71] Y. Pu *et al.*, "Variational autoencoder for deep learning of images, labels and captions," in *Proc. Adv. Neural Inf. Process. Syst.*, Sep. 2016, pp. 2352–2360.

[72] J. Walker, C. Doersch, A. Gupta, and M. Hebert, "An uncertain future: Forecasting from static images using variational autoencoders," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 835–851.

[73] T. N. Kipf and M. Welling. (2016). "Variational graph auto-encoders." [Online]. Available: https://arxiv.org/abs/1611.07308

[74] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu,"Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, Jul. 2014, pp. 1293–1299.

[75] R. van den Berg, T. N. Kipf, and M. Welling. (2017). "Graph convolutional matrix completion" [Online]. Available: https://arxiv.org/abs/1706.02263

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017.

[77] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, Jun. 2010, pp. 807–814.

[78] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Nov. 2014, pp. 2672–2680.
[79] Y. Yu, Z. Gong P. Zhong, and J. Shan, "Unsupervised representation learning with deep convolutional neural network for remote sensing images," in *Proc. Int. Conf. Image Graph.*, Dec. 2017, pp. 97–108
[80] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. (2015). "Adversarial autoencoders." https://arxiv.org/abs/1511.05644
[81] C. E. Rasmussen, "Gaussian processes in machine learning," in *Proc. Adv. lectures Mach. Learn.*, Feb. 2003, pp. 63–71.
[82] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[83] L. Tran, X. Liu, J. Zhou, and R. Jin, "Missing modalities imputation via cascaded residual autoencoder," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Aug. 2017, pp. 1405–1414.
[84] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2528–2535.
[85] M. Mäkitalo and A. Foi, "Noise parameter mismatch in variance stabilization, with an application to poisson–gaussian noise estimation," *IEEE Trans. Image Process.*, vol. 23, no. 12, pp. 5348–5359, Dec. 2014.
[86] D. H. Ackley and G. E. Hinton, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, no. 1, pp. 147–169, Jan./Mar. 1985.
[87] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," in *Proc. Parallel Distrib. Process., Explor. MicroStruct. Cogn.*, Feb. 1986, pp. 194–281.
[88] R. R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, Apr. 2009, pp. 448–455.
[89] J. Chu, H. Wang, H. Meng, P. Jin, and T. Li, "Restricted Boltzmann machines with gaussian visible units guided by pairwise constraints," *IEEE Trans. Cybern.*, to be published.
[90] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 791–798.
[91] N. Srivastava and R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2949–2980, 2014.
[92] C.-J. Kim and C. R. Nelson, *State-Space Models With Regime Switching: Classical and Gibbs-Sampling Approaches With Applications*. Cambridge, MA, USA: MIT Press, 1999.
[93] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

Shiping Wang received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2014.

He was a Research Fellow at Nanyang Technological University, Singapore, from 2015 to 2016. He is currently a Full Professor and Qishan Scholar with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His research interests include machine learning, computer vision, and granular computing.

Jinyu Cai received the B.S. degree in computer science and technology from Fuzhou University, Fuzhou, China, in 2018, where he is currently pursuing the M.S. degree with the College of Mathematics and Computer Science.

His research interests include machine learning, computer vision, and pattern recognition.

Qihao Lin received the B.S. degree in network engineering from Fuzhou University, Fuzhou, China, in 2018, where he is currently pursuing the M.S. degree with the College of Mathematics and Computer Science.

His research interests include machine learning and computer vision.

Wenzhong Guo received the B.S. and M.S. degrees in computer science from Fuzhou University, Fuzhou, China, in 2000 and 2003, respectively, and the Ph.D. degree in communication and information systems from Fuzhou University, in 2010.

He is currently a Full Professor with the College of Mathematics and Computer Science, Fuzhou University. Moreover, he leads the Network Computing and Intelligent Information Processing Laboratory, which is a Key Laboratory of Fujian Province, China. His research interests include cloud computing, mobile computing, and evolutionary computation.