Graph Regularized Auto-Encoders for Image Representation

Yiyi Liao, *Student Member, IEEE*, Yue Wang, *Student Member, IEEE*, and Yong Liu, *Member, IEEE*

***Abstract*— Image representation has been intensively explored in the domain of computer vision for its significant influence on the relative tasks such as image clustering and classification. It is valuable to learn a low-dimensional representation of an image which preserves its inherent information from the original image space. At the perspective of manifold learning, this is implemented with the local invariant idea to capture the intrinsic low-dimensional manifold embedded in the high-dimensional input space. Inspired by the recent successes of deep architectures, we propose a local invariant deep nonlinear mapping algorithm, called graph regularized auto-encoder (GAE). With the graph regularization, the proposed method preserves the local connectivity from the original image space to the representation space, while the stacked auto-encoders provide explicit encoding model for fast inference and powerful expressive capacity for complex modeling. Theoretical analysis shows that the graph regularizer penalizes the weighted Frobenius norm of the Jacobian matrix of the encoder mapping, where the weight matrix captures the local property in the input space. Furthermore, the underlying effects on the hidden representation space are revealed, providing insightful explanation to the advantage of the proposed method. Finally, the experimental results on both clustering and classification tasks demonstrate the effectiveness of our GAE as well as the correctness of the proposed theoretical analysis, and it also suggests that GAE is a superior solution to the current deep representation learning techniques comparing with variant auto-encoders and existing local invariant methods.**

***Index Terms*— Auto-encoders, graph regularization, local invariance.**

I. Introduction

ALTHOUGH the dense original image can provide intuitive visual representation, it is well known that this raw representation may cover the hidden semantic patterns which need to be recognized by those image based learning tasks. On the other side, the performance of the machine learning method also strongly depends on the corresponding data representation to which they are applied. Thus constructing an appropriate image representation becomes a fundamental problem in visual analysis.

Given an image data matrix $X \in \mathbb{R}^{n\times m}$, each column of X corresponding to an image, finding its underlying representation is to build an embedding from image space to representation space, $H = F(X)$ ($H \in \mathbb{R}^{l\times m}$), where each column vector of H is the representation of the corresponding image. In general, the embedding function F can be an iterative algorithm such as sparse coding and nonnegative matrix factorization [1], [2], or an inference encoder with linear or non-linear form, like principal component analysis and auto-encoder [3], [4]. As the problem of embedding learning is mostly over-parameterized, an additional regularizer is usually employed to constrain the underlying structure of the representation space. Therefore, the model of the embedding and the design of the regularizer are two focuses in representation learning.

A usual regularizer follows the local invariant idea, which regularizes the representation space to preserve the local structure of the neighborhood from the input space. It is inspired by the *manifold assumption* [5] that the intrinsic dimension of image data is considered to be much lower than the dimension of pixels in images. Then the local invariant regularizer aims to capture the intrinsic low-dimensional manifold embedded in the high-dimensional space. With this regularizer, many successful dimension reduction methods have been proposed such as Locally Linear Embedding (LLE) [6], ISOMAP [7], and Graph regularized Nonnegative Matrix Factorization (GNMF) [8]. The lower-dimensional representations found by these algorithms showed excellent performance on clustering. However, most of these methods do not have an explicit encoder for F. As a result, it is unclear how to leverage the learned embedding to the external test data without retraining. In addition, the models for these algorithms are considered to be shallow [9], leading to limited expressive capacity for capturing the variations in relatively complex data.

In the recent decade, stacked auto-encoders, inspired by the greedy unsupervised learning strategy on deep models training [4], [10], [11], demonstrated its remarkable effectiveness in representation learning. Based on the better expressive capability of deep architecture, some researchers proposed that adding regularizer to stacked auto-encoders, such as sparse auto-encoder (SAE) and contractive auto-encoder (CAE), can further improve the performance in image based learning tasks [12], [13]. Besides, in our early study [14], the local invariant idea is implemented as a graph regularizer by encouraging the representation to preserve the local connectivity in the input space, which is called graph regularized auto-

Manuscript received December 3, 2015; revised June 6, 2016; accepted August 24, 2016. Date of publication August 31, 2016; date of current version April 18, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant U1509210, and in part by the Natural Science Foundation of Zhejiang Province under Grant LR13F030003. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Christine Guillemot. *(Corresponding author: Yong Liu.) (Yiyi Liao and Yue Wang are co-first authors.)*

Y. Liao, Y. Wang and Y. Liu are with the State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Hangzhou 310027, China (e-mail: yyliao@iipc.zju.edu.cn; wangyue@iipc.zju.edu.cn; yongliu@iipc.zju.edu.cn).

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

Digital Object Identifier 10.1109/TIP.2016.2605010

encoder (GAE). It is also investigated in another following study [15]. Unfortunately, these works did not theoretically reveal the intrinsic influence of the graph regularizer, also not clarify its relationships to other auto-encoder regularizers.

In this paper, we present the design of the GAE as well as the efforts toward understanding GAE's underlying mechanism theoretically and experimentally. A stochastic gradient algorithm is also designed for efficient training on large datasets. The contributions are summarized as follows:

- With the graph regularization imposed to the auto-encoders, our GAE can preserve the local connectivity and provide powerful expressive capacity at the same time, which extends the conventional manifold learning algorithms into the context of deep architecture, and equip them with encoders for fast inference of representation on external test data.
- To understand the intrinsic structure of the representation learned by GAE, we give a theoretical analysis on the graph regularization. We analyze the local property of the input space, as well as the effects of constraining the local property on hidden space. Furthermore, we also clarify the relationship between the GAE and other popular variants of auto-encoders, SAE and CAE on the learned representations.
- Substantial experimental results are presented on both clustering and classification problems. GAE demonstrates competitive performance comparing with other state-of-the-art methods on both unsupervised and supervised learning tasks, validating the effectiveness and the theoretical analysis of the proposed GAE. Besides, the learned representations and parameters are visualized for further analyzing, which provides an insightful way for understanding the influence of the hyper-parameters in the graph regularizer.

The remainder of this paper is organized as follows: Section II gives a review on the related works. In Section III, we introduce our graph regularized auto-encoder for image representation learning tasks. We uncover the intrinsic effects of the graph regularizer using theoretical analysis in Section IV. Then comparison experiments on both clustering and classification tasks are presented in Section V. Finally, we provide a conclusion and future works in Section VI.

II. Background and Related Works

In the conventional efforts on image based learning tasks, most of researchers focus on representing the image with hand-craft features. They have achieved significant successes with the fixed representation function $H = F(X)$ [16]–[19]. However, these features usually should be specifically designed for different tasks. There are another series of methods that learn the representation with trainable structures, which attracted much attention due to the successful unsupervised learning criterion on deep architectures [4], [10], [11]. Given sufficient amount of training data, these deep learning methods that discover and extract features automatically can usually achieve better results on a series of tasks, such as object classification [13], [20], [21], visual tracking [22], [23], human pose recovery [24] and human re-identification [25], [26].

Deep architectures can be considered as incomplete or over-complete settings, depending on whether the dimension of the representation space is smaller or larger than that of input space. In this paper, we focus on the incomplete deep architecture to find the lower-dimensional embedding from the input space, which can also be regarded as a dimension reduction model. At the perspective of dimension reduction, the learned representation should have a lower dimension than the original one, i.e. $l < n$, and express the intrinsic property of data better at the same time. The representation capability of the lower dimension can usually be measured by the performance of the classification or clustering on H. One of the usual frameworks to model the dimension reduction problem is an optimization problem to minimize the following cost function,

$$C = \Phi(X, H) + \Psi(H) \tag{1}$$

where the first term measures the approximation of H to X, and the second term constrains the structure of the representation space. Under the *manifold assumption* [5], the aim of $\Psi(H)$ is to preserve the local property of input space in the representation space.

As a popular model to construct deep architectures, the auto-encoder [27], [28] and its variants show their powerful expressive capacity on the unsupervised representation learning. The SAE [12], [29]–[31] imposes a sparsity penalty on the hidden activations. It restricts the SAE to represent the data manifold with limited activations, thus prevents SAE from reconstructing all the input data points including the noises [32]. Different with sparse auto-encoder, the CAE [13], [20] tries to penalize the sensitivity to the input explicitly by constraining the Jacobian matrix of the encoder mapping. This method achieves state-of-the-art classification performance on a range of datasets. Denoising auto-encoder (DAE) [21], [33], [34] reconstructs the clear input from the input with small perturbations, which can be regarded as mapping a point's neighborhood to that point. Thus there is also a requirement on local insensitivity. Generally, the insensitive representation is often accompanied by the saturation, as the saturated regions of the nonlinear activation in auto-encoders are invariant to a range of inputs. Therefore the saturated auto-encoder [32] explicitly encourages the hidden activations in the saturated regions. The authors [32] also point out that the saturated auto-encoder can achieve low reconstruction error on the low-dimensional data manifold, and the reconstruction error will be raised up fast when the input moves away from the data manifold.

From all these kind of auto-encoders we can see, the representation should be insensitive to the small perturbation of the input. Alain and Bengio [35] point out the insensitivity could be assured by the penalty term corresponding to the Jacobian matrix of the encoder activations with respect to the input, which means the representation should be invariant to the neighborhood of the sampled data. This idea is also reflected in manifold learning, which aims at preserving the local neighborhood information of the input space to the hidden space. There are many successful manifold learning algorithms such as Locally Linear Embedding (LLE) [6],

ISOMAP [7], and Laplacian Eigenmap [36], as well as their applications in real problems [37]–[39], which all implement the idea of local invariance that nearby points are likely to have similar embeddings. However, as Bengio and LeCun [9] say, these methods can't perform well on generalization because of curse of dimensionality. To relieve this problem, GNMF [8] adds a reconstruction term while keeping the local invariance. The success of GNMF also confirms the importance of keeping reconstruction term in the cost function, as auto-encoders do.

There are also some works on graph regularized neural network architectures for special tasks [40], [41]. These works validate the effectiveness of introducing the local invariant constraints into deep architectures rather than auto-encoders. The GAE are also studied in [15] and [42]. In these papers as well as our early paper, the performance of GAE on supervised and unsupervised learning tasks are demonstrated. However, no theoretical analysis was included so that the understanding of underlying mechanism of GAE is limited. In this paper, we focus on the contribution of theoretic analysis and its experiment validation as a complement to our previous works of GAE [14].

III. Graph Regularized Auto-Encoder

We follow the cost functions generally employed in auto-encoders. The reconstruction term corresponds to the first term $\Phi(X, H)$ in (1) since it requires the hidden representation reconstructs the original input, which means H should preserve essential information of X. While only considering the reconstruction term usually can't lead to a good result, many works [12], [13], [21], [29] then focus on designing the additional constraint term, i.e. $\Psi(H)$. In this section, we focus on designing the second term $\Psi(H)$ to constrain the representation space, and introduce our graph regularization that implements locally geometrical invariance.

A. Single Layer Auto-Encoder Regularized With Graph

In our Graph regularized Auto-Encoder (GAE), both the decoder and the encoder use sigmoid as their activation functions. Denote sigmoid function as $f(x) = 1/(1 + e^{-x})$. Then the encoder and decoder can be presented as follow:

$$H = f(WX + b)$$
$$\hat{X} = f(W^T H + c)$$

where H and $\hat{X}$ denote the hidden representation and the reconstruction respectively. We call such architecture with one encoder connected by one decoder as single layer auto-encoder since it has only one hidden layer. In our GAE, we consider the auto-encoder with tied weights, which means the weight matrix in the decoder is the transpose of the weight matrix in the encoder.

Under the manifold assumption, data samples are considered lying on the vicinity of a low-dimensional manifold in the high-dimensional space. As the representation should discover the latent structure of the original data, the geometrical structure of the data will be the ideal underlying structure in representation learning. A reasonable assumption is that if two data points x_i, x_j are close in the intrinsic geometry of the data distribution, then their corresponding representations, h_i and h_j, should be also close to each other. This idea is usually denoted as local invariance [6], [36], [43], which plays an essential role in designing of dimension reduction algorithms.

In our GAE, we introduce the locality preserved constraint to the auto-encoder. To achieve this goal, we add a graph regularization as an additional cost to reconstruction cost as follow:

$$J = \frac{1}{m}\sum_{i=1}^{m}\|x_i - \hat{x}_i\|^2 + \frac{\lambda}{m}\sum_{i=1}^{m}\sum_{j=1}^{m} v_{ij}\|h_i - h_j\|^2 \quad (2)$$

where m is the number of samples, λ is the weight coefficient of graph regularization, v_{ij} is the weight between input samples x_i and x_j, and their corresponding representations, h_i and h_j. Then we have the adjacency graph $V = [v_{ij}]_{m\times m}$, which can be easily captured from the data distribution. As we will introduce later, the adjacency graph V, which plays an important role in the GAE, is a sparse matrix since we only connect each sample to its few nearest neighbors. Hence we normalize the graph regularization term by dividing the total number of the samples as same as the reconstruction term. More details about the designing of the adjacency graph V will be discussed in Section III-C. Based on the idea of local invariance, the graph regularization requires the lower-dimensional representations keeping the same geometrical structure as the original data. When two samples are close in the original space, which means the weight of similarity v_{ij} is large, then the h_i and h_j will be pushed to be closer in the representation space. Ideally, the representation space will be insensitive to small disturbance of the input with graph regularization. Further theoretical analysis to reveal its intrinsic influence to the hidden representation is presented in Section IV.

The cost function then can be expressed as the following matrix form:

$$J = \frac{1}{m}\|X - \hat{X}\|_F^2 + \frac{\lambda}{m} tr(HLH^T) \quad (3)$$

where $tr(\cdot)$ denotes the trace of a matrix, L is the Laplacian matrix with the form:

$$L = D_1 + D_2 - 2V \quad (4)$$

$D_1 = [d_{ij}^1]_{m\times m}$ and $D_2 = [d_{ij}^2]_{m\times m}$ are diagonal matrices with $d_{ii}^1 = \sum_j v_{ij}$ and $d_{jj}^2 = \sum_i v_{ij}$. Then the second term is calculated as follow:

$$\begin{aligned}
&\sum_i\sum_j v_{ij}\|h_i - h_j\|^2 \\
&= \sum_i h_i^T \sum_j v_{ij} h_i + \sum_j h_j^T \sum_i v_{ij} h_j - 2\sum_i\sum_j h_i v_{ij} h_j \\
&= tr(HD_1H^T) + tr(HD_2H^T) - 2tr(HVH^T) \\
&= tr(HLH^T)
\end{aligned}$$

Denote $\theta = \{W, b, c\}$, then we can optimize the auto-encoder regularized with graph as (2) and get the solution

as follow:

$$\hat{\theta} = \arg\min(\frac{1}{m}\|X - \hat{X}\|_F^2 + \frac{\lambda}{m} tr(HLH^T)) \quad (5)$$

The optimization problem given in (5) can be solved by the stochastic gradient descent algorithm presented in Section III-D.

B. Multi-Layer Auto-Encoders Regularized With Graph

The auto-encoder was proposed in the context of the neural network, which is later applied to train the deep structure of networks to obtain better performance in representation learning. We also implement the locally invariant constraint into the multi-layer auto-encoders by adding the graph regularized terms.

Use H_i to denote the data representation of the ith layer, and its corresponding reconstruction is denoted as $\hat{X}_i$. The input data of the ith layer is the data representation of $i-1$ th layer.[1] That is:

$$H_i = f(W_i H_{i-1} + b_i)$$
$$\hat{X}_i = f(W_i^T H_i + c_i)$$

Here, $\theta_i = \{W_i, b_i, c_i\}$, and then the objective function of the ith layer of the GAE is,

$$\hat{\theta}_i = \arg\min(\frac{1}{m}\|H_{i-1} - \hat{X}_i\|^2 + \frac{\lambda}{m} tr(H_i L H_i^T))$$

It is worth to mention that all of the L for different layers are generated from the original input data X, since we expect the hidden representation is able to keep the same local information as the original space.

This deep network can be trained with unsupervised layer-wise pre-training for representation learning [4]. The output of the last hidden layer is regarded as the learned representation by GAE, which can be used for classification or clustering. Usually, we fine-tune the deep network combined with the classifier in classification tasks.

C. Graph Regularizer Design

As mentioned above, the performance of data representation regularized with graph mainly relies on the design of the adjacency graph V since it encodes the local property of the data space. In this section, we introduce the design of adjacency graph based on neighborhood information. We first define the connected relationship between samples and then calculate their weights of the connected edges.

For choosing connected edges, there are two kinds of methods employed in our GAE, which are introduced as follows,

- **KNN-graph:** If x_j lies in x_i's k-nearest neighbor set, then x_i and x_j are connected on the adjacency graph.
- **ϵ-graph:** The data sample x_i's ϵ-neighbor set contains all the data samples whose distances to x_i are less than ϵ. If x_j lies in x_i's ϵ-nearest neighbor set, i.e. $\|x_i - x_j\| < \epsilon$, there is a connected edge between these two data samples.

[1]If $i = 1$, then the input data is the original input data X, and the training process is back to single layer GAE.

We denote $x_j \in N(x_i)$ if x_j is connected to x_i in the adjacency graph. Note that we don't constrain the adjacency graph to be symmetric. In the case of KNN, the adjacency graph can be asymmetric, while the ϵ-graph is always constructed in symmetry. The asymmetry might help alleviate the influence of unexpected connections. After choosing the connected edges, we calculate the weight v_{ij} depending on the similarity between x_i and x_j. Based on the cost function in (2), the connected weight should be large when the distance between two samples is small. Then we employ two weight calculation methods as follows,

- **Binary:** If $x_j \in N(x_i)$, then $v_{ij} = 1$, otherwise $v_{ij} = 0$.
- **Heat kernel:** If $x_j \in N(x_i)$, then

$$v_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\sigma}}$$

otherwise $v_{ij} = 0$. Here σ is a user-designed constant. v_{ij} is also bounded in $[0, 1]$ in this setting.

D. Model Learning

The layer-wise pre-training method proposed by Hinton and Salakhutdinov [4] is widely used for training deep architectures. For pre-training a single-layer GAE, our goal is to solve the optimization problem in (5).

Since the graph regularization contains the weight matrix V of training samples, it will be time-consuming to train all of the data at the same time. Here we use stochastic gradient descent for pre-training. The training algorithm is given in Algorithm 1. For each training batch, we don't select a random set of data directly since the random batch cannot reveal the global neighborhood information. Instead, we select a small random set at first, and then add all their neighbor samples into this set as our training batch.

In practice, it is time-consuming to loop over each sample or each neighbor pair as shown in Algorithm 1. Here we provide the gradient for each batch in matrix form, which means the gradient of a batch can be calculated without looping. Let $Z^{(l)}$ denote the total weighted sum of inputs in layer l, $X^{(l)}$ denote the output of the active function in layer l, which means $X^{(l)} = f(Z^{(l)})$ for $l = 2, 3$. Furthermore, denote $X^{(1)} = X_{batch}$ for uniformity.[2] Since the cost J is composed of reconstruction error and graph regularization, which can be denote as $J = J_{rec} + J_{graph}$, the gradients can also be computed as two parts separately. Here we compute the "error term" $\delta^{(l)} = \frac{\partial J}{\partial Z^{(l)}}$ at first, which corresponds to the influence of $Z^{(l)}$ to the error. For the reconstruction term, $\delta_{rec}^{(l)}$ is given as follow:

$$\delta_{rec}^{(3)} = -2(X^{(1)} - X^{(3)}) \bullet f'(Z^{(3)})$$
$$\delta_{rec}^{(2)} = (W\delta_{rec}^{(3)}) \bullet f'(Z^{(2)})$$

where $\bullet$ denotes the Hadamard product. For the graph regularization, the $\delta_{graph}^{(l)}$ is given as:

$$\delta_{graph}^{(3)} = 0$$
$$\delta_{graph}^{(2)} = (2\lambda X^{(2)}(L + L^T)) \bullet f'(Z^{(2)})$$

[2]In our paper, the single-layer GAE means it has only one hidden layer. Each single-layer GAE actually has 3 layers, where $l = 1, 2, 3$ denotes the input layer, hidden layer and output layer respectively.

Algorithm 1 Stochastic Gradient Descent for Pre-Training Single Layer GAE

Input: Input dataset X, size m_1, randomly initialized $\theta = \{W, b, c\}$
Output: $\theta = \{W, b, c\}$

Calculate the weight matrix V of all the input X;
while *not stopping criterion* **do**
  Randomly choose m_1 samples from X and find their m_2 neighbors, denote these $m_1 + m_2$ samples as X_{batch};
  Perform a feedforward pass on X_{batch}, calculate the cost J;
  Set $\Delta W = 0$, $\Delta b = 0$, $\Delta c = 0$;
  for *sample* $i \in X_{batch}$ **do**
    Calculate the gradient of reconstruction term ΔW_{rec}, Δb_{rec} and Δc_{rec} with respect to sample i;
    $\Delta W = \Delta W + \frac{1}{m_1+m_2}\Delta W_{rec}$;
    $\Delta b = \Delta b + \frac{1}{m_1+m_2}\Delta b_{rec}$;
    $\Delta c = \Delta c + \frac{1}{m_1+m_2}\Delta c_{rec}$;
  end
  for *neighbor pairs* $i, j \in X_{batch}$ **do**
    Calculate the gradient of graph regularization ΔW_{graph} and Δb_{graph} and Δc_{graph} with respect ro samples i, j;
    $\Delta W = \Delta W + \lambda\frac{1}{m_1+m_2}\Delta W_{graph}$;
    $\Delta b = \Delta b + \lambda\frac{1}{m_1+m_2}\Delta b_{graph}$;
    $\Delta c = \Delta c + \lambda\frac{1}{m_1+m_2}\Delta c_{graph}$;
  end
  $W = W - \Delta W$;
  $b = b - \Delta b$;
  $c = c - \Delta c$.
end

Then we have $\delta^{(2)} = \delta^{(2)}_{rec} + \delta^{(2)}_{graph}$ and $\delta^{(3)} = \delta^{(3)}_{rec} + \delta^{(3)}_{graph}$. Finally, the gradients is computed using the error term:

$$\Delta W = \frac{1}{m_1 + m_2}\left((\delta^{(2)})(X^{(1)})^T + (\delta^{(3)})(X^{(2)})^T\right)$$
$$\Delta b = \frac{1}{m_1 + m_2}\sum\nolimits_j \delta^{(2)}_{ij}$$
$$\Delta c = \frac{1}{m_1 + m_2}\sum\nolimits_j \delta^{(3)}_{ij}$$

where ΔW is calculated using $\delta^{(2)}$ and $\delta^{(3)}$ together since we use tied weights. With the gradient computing in matrix form, the GAE can be trained in relatively high efficiency.

IV. Theoretical Analysis on Graph Regularization

It is intuitive to find in (2) that the graph regularization aims to achieve the local invariance. Hence the hidden representation will be insensitive to the small changes of the input. In this section, we further discover the underlying theoretical mechanism of our GAE. The relationships among GAE, CAE and SAE are also investigated based on the theory.

To find out the effect of the graph regularization on the hidden representation, we conduct the mathematical analysis in the case of the continuous space. Given a continuous input sample $x_i \in \mathbb{R}^m$ generated from the probability density function $p(x_i)$, we denote its corresponding neighbors as x_j with the conditional probability density $p(x_j|x_i)$. Here we use the ϵ-neighborhood to describe the local property of any given data points. If we have finite number of samples from the neighborhood, they can form the graph as mentioned in SectionIII-C. Without loss of generality, the connection weight between each pair of data is set binary, i.e. $v_{ij} = 1$ for connected neighboring pair x_i and x_j, and $v_{ij} = 0$ for others. Then the conditional density $p(x_j|x_i)$ defined in the ϵ-neighborhood of x_i is given as,

$$p(x_j|x_i) = \frac{p(x_i)1_{\|x_i - x_j\| < \epsilon}}{Z(x_i)}$$

where $Z(x_i)$ is the normalizing term to ensure that $p(x_j|x_i)$ is a valid probability density function. We can give the conditional mean and covariance of x_j based on $p(x_j|x_i)$ as,

$$\mu_{x_j|x_i} = \int x_j p(x_j|x_i)dx_j$$
$$\Sigma_{x_j|x_i} = \int (x_j - \mu_{x_j|x_i})(x_j - \mu_{x_j|x_i})^T p(x_j|x_i)dx_j$$

Then we can rewrite our graph regularization term as shown in (6), as shown at the bottom of the next page.

Notice that we employ the Taylor expansion around h_j as

$$h_j = h_i + J_i(x_i - x_j) + o(\sigma^2)$$

where J_i means the Jacobian matrix of the encoder mapping at the particular sample x_i, then the approximate equation at third line in (6) is obtained by omitting the second-order term $o(\sigma^2)$. For the equation on sixth line, we use the property of quadratic form that the expectation of the quadratic form $x^T\Lambda x$ is

$$E[x^T\Lambda x] = tr[\Lambda\Sigma] + \mu^T\Lambda\mu$$

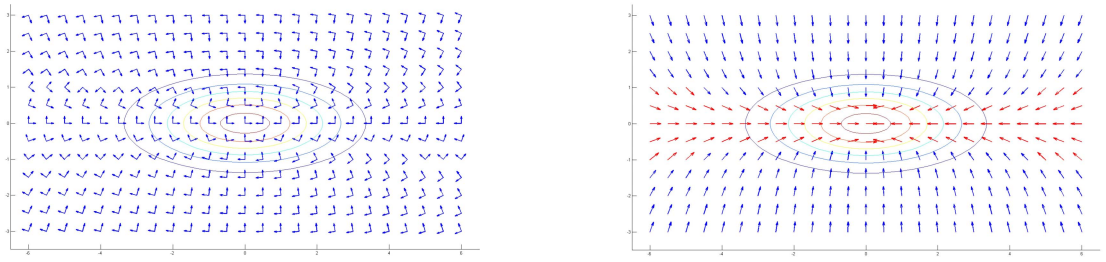
where x is a vector, μ and Σ denote the expected value and variance matrix of x respectively. Furthermore, $\Sigma_{x_j|x_i}$ and $(x_j - \mu_{x_j|x_i})(x_j - \mu_{x_j|x_i})^T$ are both positive-definite matrices, then the Cholesky decomposition can be applied so that $\Sigma_{x_j|x_i} + (x_j - \mu_{x_j|x_i})(x_j - \mu_{x_j|x_i})^T = L_i L_i^T$ on the tenth line in (6), where L_i is a lower triangular matrix with positive diagonal entries. The derived result in (6) shows that GAE tries to minimize the Frobenius norm of the weighted Jacobian matrix. It is in accordance with our original intention for learning local invariant representation.

A. Local Property in Input Space

Intuitively, the weight matrix L_i encodes the local property around the sample x_i. Let's denote $R_i = L_i L_i^T$. To reveal the local property of the input space, we diagonalize R_i by eigenvalue decomposition as:

$$R_i = Q_i \Lambda_i Q_i^T$$

where Q_i includes the eigenvectors of R_i, and Λ_i is the diagonal matrix composed of eigenvalues.



(a) (b)

Fig. 1. Eigenvectors of sampled points. Left sub-figure shows all of the two eigenvectors for the two-dimensional input. The eigenvectors are orthogonal to each other since R is real positive. Right sub-figure shows the largest eigenvector on each sampled point. Red eigenvectors represent that the projections of the sample points on horizontal axis are larger, while blue eigenvectors represent that the projections of the sample points on vertical axis are larger. The elliptical contours corresponds to the two-dimensional Gaussian distribution, the sample points in each ellipse have the same value of the probability density.

Note the tenth line of (6), it tends to minimize the expectation of $tr[J_i L_i L_i^T J_i^T]$ over the input space. Denote each row of Jacobian matrix as J_{pi}, one can see that the cost minimized is equivalent to

$$\begin{aligned} tr[J_i L_i L_i^T J_i^T] &= \sum_p J_{pi} R_i J_{pi}^T \\ &= \sum_p J_{pi} Q_i \Lambda_i Q_i^T J_{pi}^T \end{aligned}$$

It means that the R_i is propagated to each hidden unit by projecting J_{pi} on Q_i with the weight $\sqrt{\Lambda_i}$. As Q_i and Λ_i vary with respect to the input sample x_i, the weight imposed to the Jacobian matrix also varies from an input sample to another.

To qualitatively uncover the local property contained in R_i, a simple numerical example is carried out for illustration. For simplicity, we assume that our input data generated from a two-dimensional Gaussian distribution, i.e. $p(x_i) \sim N(\mu, \sigma)$ with $\mu = [0, 0]^T$ and $\sigma = [3, 0; 0, 0.5]$. With known input data distribution, we can calculate the weighted matrix for any given x_i. The elliptical contours shown in Figure 1 present this two-dimensional probability density.

We sample points x_i from the continuous input space with constant intervals along both axes. As for the conditional probability density $p(x_j|x_i)$, we set $\epsilon = 0.3$ in this example, then we have

$$p(x_j|x_i) = \frac{p(x_i) 1_{\|x_i - x_j\| < 0.3}}{Z(x_i)}$$

The corresponding R_i is calculated on these sampled points. Figure 1(a) shows the eigenvectors of R_i on each sampled point x_i. In this case, each R_i has two eigenvectors since its rank is 2. Figure 1(a) shows that the distribution on each ϵ-neighborhood of x_i is different from each other, even in the simple normal Gaussian distribution. Considering the sample points lying on the major axis and minor axis of the ellipse, we see that their directions of the eigenvectors are nearly parallel to the horizontal axis and vertical axis. Although the sample

$$\begin{aligned} J_{graph} &= E\left[v_{ij}\|h_i - h_j\|^2\right] \\ &= \int\int \|h_i - (h_i + J_i(x_i - x_j) + o(\sigma^2))\|^2 p(x_j|x_i) p(x_i) dx_i dx_j \\ &\approx \int\int \|J_i(x_i - x_j)\|^2 p(x_j|x_i) p(x_i) dx_i dx_j \\ &= \int\int (x_i - x_j)^T J_i^T J_i (x_i - x_j) p(x_j|x_i) p(x_i) dx_i dx_j \\ &= \int\int \left(x_j^T J_i^T J_i x_j - 2x_i^T J_i^T J_i x_j + x_i^T J_i^T J_i x_i\right) p(x_j|x_i) dx_j\, p(x_i) dx_i \\ &= \int \left(tr\left[J_i^T J_i \Sigma_{x_j|x_i}\right] + \mu_{x_j|x_i}^T J_i^T J_i \mu_{x_j|x_i} - 2x_i^T J_i^T J_i \mu_{x_j|x_i} + x_i^T J_i^T J_i x_i\right) p(x_i) dx_i \\ &= \int \left(tr\left[J_i^T J_i \Sigma_{x_j|x_i}\right] + \left(x_i - \mu_{x_j|x_i}\right)^T J_i^T J_i \left(x_i - \mu_{x_j|x_i}\right)\right) p(x_i) dx_i \\ &= \int \left(tr\left[J_i^T J_i \Sigma_{x_j|x_i}\right] + tr\left[J_i^T J_i \left(x_i - \mu_{x_j|x_i}\right)\left(x_i - \mu_{x_j|x_i}\right)^T\right]\right) p(x_i) dx_i \\ &= \int \left(tr\left[J_i^T J_i \left(\Sigma_{x_j|x_i} + \left(x_i - \mu_{x_j|x_i}\right)\left(x_i - \mu_{x_j|x_i}\right)^T\right)\right]\right) p(x_i) dx_i \\ &= \int tr[J_i L_i L_i^T J_i^T] p(x_i) dx_i \\ &= E\left[\|JL\|_F^2\right] \end{aligned} \tag{6}$$

points lying on the same ellipse have the same value of the probability density, their eigenvectors are different and rotated with respect to the sample location. On each sample point, the eigenvectors present the directions on which each row of Jacobian matrix, J_{pi}, should be projected.

To find out the impact of the eigenvalues, we preserve the eigenvector corresponding to larger eigenvalue in Figure 1(b). We use the red arrows to denote those eigenvectors with larger projection on horizontal axis, and the blue arrows to denote those eigenvectors with larger projection on vertical axis. One can see that the major components of the preserved eigenvectors are also different with respect to the sample location. It is interesting to note the preserved eigenvectors provide an estimation of the gradient on the input space. Therefore, R_i can tell the amount and the direction of the variation in the input space encoded by its eigenvectors in Q_i and eigenvalues in Λ_i.

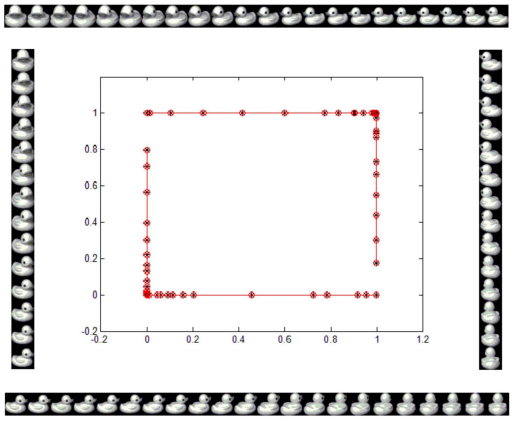


Fig. 2. Given 72 images where each represents a pose of a toy duck, they are projected to a two-dimensional hidden space based on GAE. Each point in the two-dimensional hidden space denotes an image sample, the images lie in the surroundings correspondingly.

B. Effect of Minimizing Graph Regularizer

To theoretically understand the variations in hidden space, which is propagated by the Jacobian matrix from R_i, we focus on the first component in the ninth line of (6), $tr[J_i^T J_i \Sigma_{x_j|x_i}]$. By circulating the terms in trace, we have

$$J_{cov} \triangleq tr[J_i \Sigma_{x_j|x_i} J_i^T]$$

According to the uncertainty propagation theory, the term within the trace is actually the covariance matrix of the hidden representation for the conditional distribution given the input sample x_i, which can be denoted as follow:

$$\Sigma_{h_j|x_i} = J_i \Sigma_{x_j|x_i} J_i^T$$

As can be seen, $\Sigma_{h_j|x_i}$ models the uncertainty in hidden space propagated from the input space by Jacobian matrix. The trace only care about the diagonal entries, leading to

$$\begin{aligned} J_{cov} &= \sum_p J_{pi} \Sigma_{x_j|x_i} J_{pi}^T \\ &= \sum_p \Sigma_{h_{pj}|x_i} \end{aligned}$$

where $\Sigma_{h_{pj}|x_i}$ is the uncertainty of hidden unit p. It means that J_{cov} measures the sum of the variance in each hidden unit, regardless of the covariance across hidden units. As a result, an underlying mechanism of the graph regularizer is to enforce most hidden units as constant as possible in the neighborhood, so that the variance can be small. To achieve this goal, the learned representation would be affected in two ways:

- **Effect 1:** For each hidden unit, its responses to most data samples in the neighborhood lie in the saturation region, as the variance in the saturation region can be small.
- **Effect 2:** The neighboring data samples are expected to be saturated in the similar subset of hidden units, which can lead to reduced variance.

To illustrate these two effects, we implement a toy example on 72 images with each one representing a pose of the toy duck. The images with similar poses are regarded as neighboring samples and connected to form a graph. After training a GAE with two hidden units on this dataset, the hidden representations are visualized in Figure 2. One can see that all the samples are located in the boundary of the unit square, indicating that all representations are with at least one hidden unit saturated. This support the first effect found above. The results also show the neighboring samples are near in the hidden space, which means they share the same saturated hidden unit. The differences between the neighboring samples are stored in the remaining unsaturated hidden units. This result supports the second effect found above. In addition, the toy example also indicates that, although only the first component in (6) is considered, the derived effect is definitely supported by the experiment. These two effects are further analyzed and explained in Section V.

C. Relationships to Other Auto-Encoders

The SAE regularized with sparse constraint is similar to our GAE in the first effect that both of them try to obtain more saturated units. Here a slight difference is that SAE keeps the hidden units saturated in one side (approach to 0 due to the sparse constraint), while GAE can reach saturation region in two sides (either 0 or 1). The major difference between these two auto-encoders occurs when considering the second effect. The SAE does not impose the learned representations to share the similar saturated units for neighboring samples. This suggests that the SAE can have the different group of saturated units for the neighboring samples comparing with GAE. As a result, the local neighborhood of the input space may not be preserved in the representation space.

CAE tries to minimize the following regularization on Jacobian matrix:

$$J = \frac{1}{m} \sum_{i=1}^{m} \|x_i - \hat{x}_i\|^2 + \frac{\lambda}{m} \sum_{i=1}^{m} \|J_f(x_i)\|_F^2$$

Note that the regularizer of CAE can be included as a special form in the derived regularizer in (6) when the weight $L_i L_i^T$ is an identity matrix, I. It is not surprising that CAE actually performs in the similar way that GAE does since both two methods regularize the Jacobian matrix, so that the hidden representation could be invariant to small perturbations in the

input space. Thus the CAE also encourages the hidden units to be saturated as the same as our first effect. However, CAE is equal to assume that the local property at each sample point is the same. As demonstrated in Figure 1 that even in the simple Gaussian distribution, the local variation varies with respect to the direction and the amount. By regularizing only $J_i J_i^T$, the local property in the input space is not accurately described. As a result, the neighborhood structure in the input space may be partially lost in the hidden space in CAE. However, compared with SAE which does not consider the local invariance, CAE should perform better. It is further analyzed experimentally in Section V.

In summary, with the help of local neighborhood description based on the data samples, it is possible for GAE to intrinsically impose more structures of the representation space from aspects of saturation and positions of saturated units in local regions. The relationships and differences among GAE, SAE and CAE will be further investigated in the following experiments.

V. Experimental Results

In this section, comparison experiments are carried out to demonstrate the performance of our proposed method in both clustering and classification tasks. In each case, our GAE and other comparable state-of-the-art methods are evaluated within the framework of embedded representation learning.

A. Clustering Experiment

Clustering is a key method for evaluating the quality of the unsupervised learned features. In this section, we validate the clustering performance of the proposed method in addition with several comparison methods on two dataset, MNIST and COIL20. MNIST is the well-known digit classification dataset,[3] which has 70,000 digit images with 10 classes. Generally, it is split into 50,000 training images, 10,000 validation images and 10,000 test images. Here we conduct clustering experiments on the 50,000 training images. COIL20 is composed of 1440 images collected from 20 objects with different views. Here all 1440 images with 20 classes are considered for clustering, and the images are pre-processed by PCA for all methods. In our experiments, we first implement various dimension reduction methods on these two datasets, then the k-means clustering is applied to the low-dimensional representations learned by those methods. We use two metrics, the normalized mutual information(MI) and accuracy(AC), to measure the clustering performances. For fair comparison, the dimension of learned representation through all algorithms are set to be the same.

1) Evaluation Metrics: The normalized mutual information (MI) is given in Cai et al. [8], which is a normalized measure to evaluate the similarity of two sets of clusters. The accuracy (AC) is used to measure the percentage of correct labels compared to the ground truth label provided by the dataset. Specifically, given a data sample x_i with clustered label c_i and ground truth label g_i, the accuracy is defined as

$$AC = \frac{\sum_i \delta(g_i, map(c_i))}{n}$$

where n is the total number of samples, $\delta(a, b)$ is a delta function, which outputs 1 when $a = b$ and outputs 0 otherwise, $map(c)$ is the permutation mapping function that maps each clustered label c_i to the best label provided by the dataset. For the normalized mutual information, denote C and C' as the set of clusters obtained from the ground truth and the k-means. We first compute the mutual information as follows,

$$\overline{MI}(C, C') = \sum_{c_i \in C, c'_j \in C'} p(c_i, c'_j) \log \frac{p(c_i, c'_j)}{p(c_i)p(c'_j)}$$

where $p(c_i)$ and $p(c'_j)$ are the probabilities that a sample selected from the dataset is belonging to cluster c_i or c'_j, $p(c_i, c'_j)$ is the probability that a sample selected from the dataset is belonging to both c_i and c'_j. Then, the normalized mutual information can be computed as,

$$MI(C, C') = \frac{\overline{MI}(C, C')}{max(H(C), H(C'))}$$

where $H(C)$ and $H(C')$ are the entropy of C and C'. When $MI = 1$, these two clusters are identical. when $MI = 0$, these two clusters are independent.

2) Comparison Methods: The methods we compared in our clustering experiments is composed of two parts, one is the locality preserving methods including LPP and GNMF, and the other is the auto-encoder variants including the original AE, SAE, CAE and the proposed GAE. A brief introduction to these comparing methods are given as follows:

- LPP: Locality Preserving Projection (LPP) [44] is a linear method for dimensional reduction. It also preserves the neighborhood information of the input data space from the adjacency graph.
- GNMF: Graph regularized Nonnegative Matrix Factorization [8]. It combines the nonnegative constraint and the locally invariant constraint, which can be formulated as follow,

  $$\hat{\theta} = \arg\min \|X - UV^T\|_F^2 + \lambda tr(V^T L V)$$

  where the elements in both basis matrix U and representation matrix V are nonnegative, L is also the Laplacian matrix in (4), and λ corresponds to the strength of regularization. In our experiments, the GNMF employs the unsupervised KNN-graph with Heat kernel weights. Those two coefficients, the number of connected neighbors k and the weight of regularization λ, are selected by the optimal grid search.
- AE: The original basic auto-encoder with tied weights. It is considered as a baseline of the auto-encoder variants.
- SAE: Sparse Auto-encoder (SAE) [29] is a frequently-used model for building deep architectures. The sparse constraint imposed to the auto-encoder is a very common constraint choice in the field of auto-encoder. The formula is given as follows,

  $$\hat{\theta} = \arg\min \|X - \hat{X}\|_F^2 + \eta \sum_j KL(\rho|\rho_j)$$

[3] http://yann.lecun.com/exdb/mnist/index.html

TABLE I

CLUSTERING RESULTS OF COMPARISON METHODS ON MNIST, EVALUATED WITH NORMALIZED MUTUAL INFORMATION (MI) AND ACCURACY (AC)

Class	5	6	7	8	9	10	Average
LPP MI	45.09 ± 11.34	47.29 ± 3.69	48.90 ± 6.28	43.77 ± 10.89	43.54 ± 4.39	46.42 ± 0.03	45.84
GNMF MI	62.30 ± 11.93	62.54 ± 7.58	60.68 ± 5.69	66.04 ± 8.68	68.20 ± 5.53	64.66 ± 4.43	64.07
AE MI	47.89 ± 9.97	41.86 ± 7.94	43.18 ± 5.49	41.51 ± 5.48	42.36 ± 1.83	40.35 ± 1.03	42.86
SAE MI	52.46 ± 9.44	51.33 ± 5.92	46.40 ± 5.29	45.08 ± 5.10	45.68 ± 5.21	42.97 ± 2.07	47.32
CAE MI	59.45 ± 7.22	58.40 ± 3.83	55.19 ± 10.60	54.80 ± 5.47	53.32 ± 1.60	49.44 ± 2.77	55.10
GAE MI	**70.41 ± 11.48**	**69.97 ± 7.93**	**71.52 ± 4.97**	**68.94 ± 5.40**	**69.49 ± 2.13**	**66.26 ± 2.97**	**69.43**
LPP AC	58.63 ± 11.45	57.10 ± 2.82	55.88 ± 4.86	50.37 ± 8.79	47.58 ± 3.74	50.06 ± 0.14	53.27
GNMF AC	69.72 ± 12.10	72.08 ± 9.00	67.66 ± 5.55	69.28 ± 12.09	69.65 ± 6.70	65.22 ± 6.34	68.93
AE AC	69.90 ± 8.28	56.88 ± 6.74	54.35 ± 5.79	53.52 ± 6.01	52.62 ± 4.80	49.64 ± 1.34	56.15
SAE AC	73.12 ± 7.72	65.83 ± 4.92	61.91 ± 9.57	57.68 ± 5.73	57.56 ± 6.40	52.86 ± 3.36	61.49
CAE AC	71.39 ± 8.26	70.43 ± 6.27	65.39 ± 11.17	62.61 ± 4.97	60.54 ± 2.99	54.58 ± 3.30	64.16
GAE AC	**81.77 ± 10.89**	**75.85 ± 9.12**	**77.57 ± 7.03**	**74.50 ± 2.79**	**72.28 ± 3.06**	**68.20 ± 1.25**	**75.03**

TABLE II

CLUSTERING RESULTS OF COMPARISON METHODS ON COIL20, EVALUATED WITH NORMALIZED MUTUAL INFORMATION (MI) AND ACCURACY (AC)

Class	6	8	10	12	14	16	20	Average
LPP MI	96.30 ± 5.68	90.68 ± 3.70	91.95 ± 3.59	**90.14 ± 3.06**	94.37 ± 2.91	89.88 ± 3.10	**91.04 ± 1.03**	92.05
GNMF MI	91.78 ± 8.24	89.64 ± 3.94	92.44 ± 3.96	89.04 ± 3.85	91.39 ± 5.19	86.24 ± 5.64	83.21 ± 2.01	89.11
AE MI	72.91 ± 7.97	74.01 ± 8.13	72.71 ± 9.16	73.73 ± 3.35	76.04 ± 3.26	76.76 ± 2.79	75.31 ± 1.04	74.50
SAE MI	76.84 ± 8.53	71.09 ± 9.20	77.00 ± 9.92	81.81 ± 3.94	77.12 ± 6.76	79.10 ± 2.24	76.15 ± 1.59	77.02
CAE MI	81.90 ± 5.46	81.06 ± 10.09	76.19 ± 8.58	77.72 ± 5.66	79.01 ± 4.99	75.30 ± 4.28	76.58 ± 0.71	78.25
GAE MI	**96.84 ± 4.13**	**92.52 ± 5.36**	**92.32 ± 6.57**	89.88 ± 4.13	**94.75 ± 1.08**	**89.94 ± 3.10**	90.14 ± 0.48	**92.34**
LPP AC	94.21 ± 10.87	**88.82 ± 6.64**	82.97 ± 5.14	82.45 ± 6.29	85.95 ± 8.37	79.24 ± 4.71	**82.65 ± 3.36**	85.18
GNMF AC	91.76 ± 8.59	87.01 ± 8.50	86.67 ± 7.02	81.97 ± 4.34	85.50 ± 8.98	79.53 ± 7.29	71.53 ± 3.55	83.42
AE AC	77.64 ± 4.96	72.78 ± 9.24	68.00 ± 9.43	69.58 ± 4.26	69.21 ± 2.93	68.66 ± 4.20	64.69 ± 3.09	70.08
SAE AC	78.89 ± 8.22	71.15 ± 9.56	73.75 ± 10.04	77.27 ± 6.11	71.73 ± 7.37	72.41 ± 3.49	65.69 ± 2.71	72.98
CAE AC	83.47 ± 3.22	81.18 ± 11.00	73.06 ± 10.79	72.66 ± 4.05	72.90 ± 4.62	68.56 ± 4.90	66.81 ± 3.45	74.09
GAE AC	**97.36 ± 3.75**	**88.82 ± 9.79**	**88.03 ± 10.59**	**85.07 ± 6.74**	**90.30 ± 2.52**	**81.23 ± 3.44**	81.58 ± 1.32	**87.48**

where η is the coefficient to regularize the strength of sparse constraint, ρ is user defined sparsity coefficient which is close to zero, ρ_j is the average response of the jth hidden unit for the whole dataset. The sparseness is implemented by the KL distance regularization that the average response ρ_j is forced to be similar to the small ρ. In our experiments, we set $\rho = 0.05$ and η is selected by the optimal grid search.

- CAE: Contractive Auto-Encoder. As introduced in Section II, CAE is a state-of-the-art method for unsupervised feature extraction, which tries to minimize the average Jacobian norm of the hidden activations with respect to the input. The formula is given as follow,

$$\hat{\theta} = \arg\min \|X - \hat{X}\|_F^2 + \eta \|J_f(x)\|_F^2$$

where η corresponds to the strength of regularization. In our experiments, η is also selected by the optimal grid search. Similar to GAE, CAE also tends to learn a invariant representation by penalizing the sensitivity of representation to the input.

- GAE: Graph Regularized Auto-Encoder. We use the KNN-graph with Heat kernel weights the same as GNMF. Our coefficients k and λ are also selected by the optimal grid search.

3) Clustering Results: In our experiments, we set the features learned from all different methods to the same dimension. More specifically, For LPP and GNMF, the dimension of the learned representation is set to be the number of classes n_c of the input dataset directly. As for all auto-encoders models, the network configurations for MNIST and COIL-20 are set according to the number of samples of each dataset. Specifically, we use 3 hidden layers for MNIST and the network architecture is 784-500-500-n_c. For COIL-20, we use 2 hidden layers and the network architecture is 1024-200-n_c. For all methods, the k-means is applied to the n_c dimensional representation for clustering, where the data samples are partitioned into n_c clusters.

To randomize the experiments, we randomly choose a series of subsets with different number of classes n_c from those two datasets to learn the representation. For each given n_c, we repeat the random selection 5 times and then average the clustering results. The average clustering results as well as standard deviation on different subsets under various n_c of those two datasets are shown in Table I, II. For both tables, the first row is the number of classes n_c of the subsets, and the column of the tables shows the average performances on the five random subsets with the n_c classes. Furthermore, the overall average clustering result of each dimension reduction method on all of the subsets is shown in rightmost column of the tables.

Table I and Table II suggest that the proposed GAE achieves superior clustering performances on both MNSIT and COIL20. Comparing LPP and GNMF with GAE, we can find that all these methods demonstrate competitive results on COIL20 with consideration of the local preserving. However, the performances of LPP and GNMF on MNIST are relatively lower due to the limitation of their shallow architectures. The superiority of GAE on MNIST comparing to LPP and GNMF demonstrates the expressive power of the deep architecture.

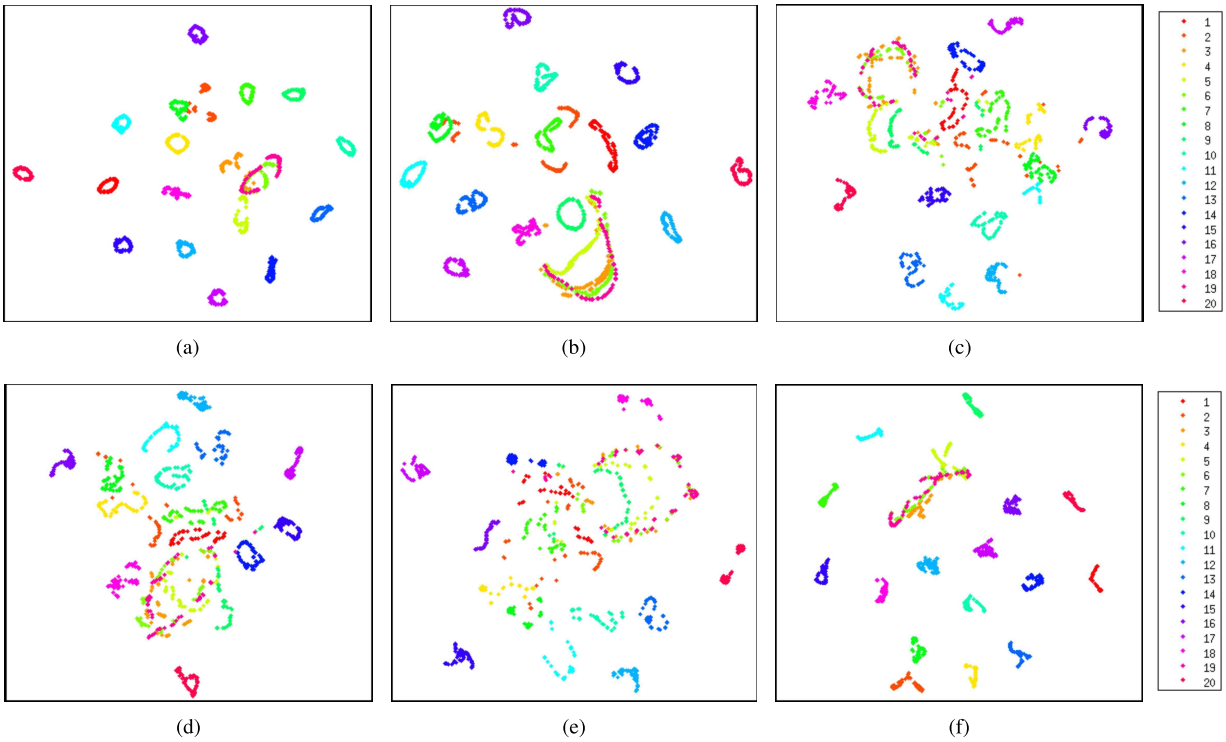


Fig. 3. Representations learned by different methods on COIL20, reduced to two dimension using t-SNE. Each figure contains the full COIL20 image dataset with 1440 samples, where different color denotes different class.

It is to be noted that the performance of LPP on MNIST degenerates more considerable with its linear structure, while GNMF is more stable on both MNIST and COIL20, demonstrating the effectiveness of simultaneously learning reconstruction and regularizing the local property. For comparison between auto-encoders, SAE, CAE and GAE all perform better than the original AE intuitively. Furthermore, GAE also gives better results than SAE and CAE, demonstrating the effectiveness of the proposed GAE.

TABLE III
SATURATION RATIO OF REPRESENTATIONS LEARNED FROM MNIST AND COIL-20

	MNIST(%)	COIL20(%)
AE	20.74	38.35
SAE	78.46	91.77
CAE	92.11	80.38
GAE	82.80	88.97

4) Saturation Analysis: To validate the first effect that the graph regularization encourages the hidden units to approach the saturation region, we analysis the saturated ratio of the learned representation of different methods. As LPP and GNMF have no saturation region, they are not taken into consideration. We define the saturation unit the same as Rifai et al. [13], which considers a unit is saturated when its activation is below 0.05 or above 0.95. Without loss of generality, we analysis the saturated ratio of the representations learned from the full dataset, where we have $n_c = 10$ for MNIST and $n_c = 20$ for COIL20. In Table III, we present the mean saturated ratio of all samples on all hidden units, including all hidden layers.

The results in Table III provide quantitative support to our analysis that the GAE encourages the hidden units to be saturated. In addition, SAE and CAE also learn saturated representations with their regularizations, while the original AE has a low saturated ratio. Correspondingly, all SAE, CAE and GAE outperform the original AE in Table I and Table II. It is suggested that the saturated ratio can partially reflect the representation learning performance. However, the clustering performance is not exactly proportional to the saturated ratio. We use the following visualization experiment to demonstrate that our GAE gains the advantage with its second effect.

5) 2D Visualization of Learned Representations: To further explain the advantage of our graph regularization, we visualize the representations used in the saturation analysis in 2D space. For both MNIST and COIL20, the learned representations are reduced to two dimensions based on t-SNE [45], which is a dimension reduction technique widely employed for high-dimensional data visualization. As shown in both Figure 3 and Figure 4, the qualitative results validate the second effect that GAE regularize the neighboring samples having similar hidden activations.

Specifically, Figure 3 illustrates representations learned on the full COIL20. Each image in COIL20 is represented as a 2D point with different color denoting different class, and all 1440 samples are presented in this figure. As can be seen, the qualitative results are consistent with the quantitative results in Table II. LPP, GNMF and the proposed GAE outperform the other methods, demonstrating the effectiveness

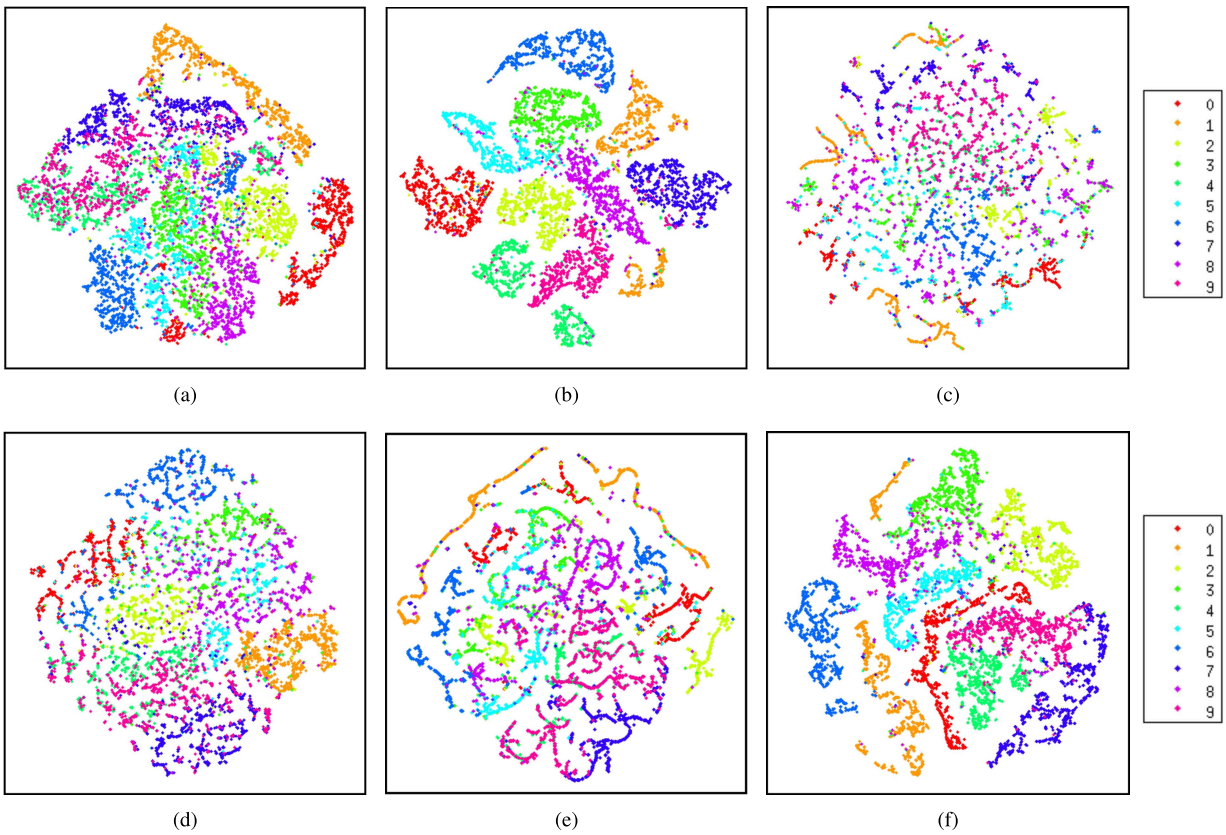


Fig. 4. Representations learned by different methods on MNIST, reduced to two dimension using t-SNE. Each figure contains 10,000 samples randomly chose from MNIST, where different color denotes different class.

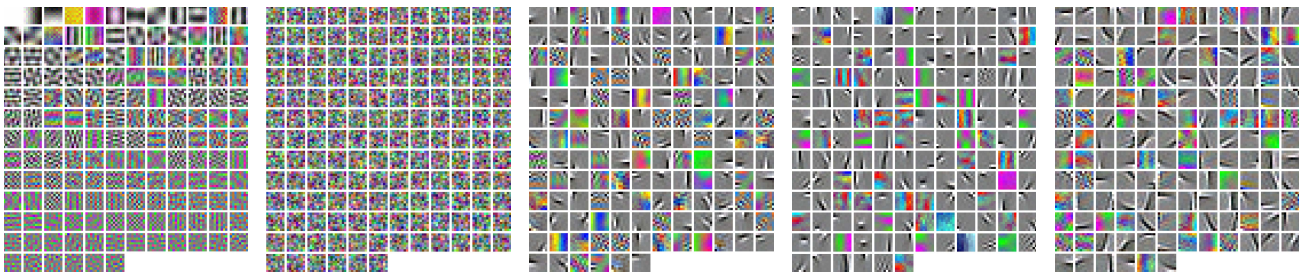
of the graph regularization. For SAE and CAE, it is suggested that samples with different classes are more likely to be mixed in their representation space. The rationale is that SAE only explicitly regularizes the saturated ratio with the sparsity regularization, but does not require the neighboring samples to be saturated in the similar subset of hidden units as the second effect of the GAE. As for CAE, it regularizes the Jacobian matrix on each sample point equally, while the GAE propagates the local property in the input space to regularize the variation in hidden space and leads to better representations.

In Figure 4, the representations learned on the full MNIST are also visualized based on t-SNE, where 10,000 samples are randomly chosen for visualization in this figure. It also supports the results in Table I that GNMF and GAE achieves better performances on MNIST clustering. The advantage of GAE over other auto-encoders is similar as in Figure 3, where the neighbor samples are projected to similar hidden representations according to the second effect. It is to be noted that CAE regularizes the representation to lie on the manifold. However, in some cases samples from different class are contracted to the same manifold. Thus they are mixed up and the clustering performance is decreased, which reflects our theoretical analysis that the structure of the neighborhood may be lost in CAE. For GAE, the contraction property is also revealed in the figure because of the similarity between CAE and GAE, while the local connectivity is preserved at the same time.

B. Classification Performance

In addition to the clustering experiment, we also conduct the classification experiments to validate the effectiveness of the proposed algorithm. The dataset we use for classification including MNIST and CIFAR-10. CIFAR-10 is a benchmark dataset containing 10 classes of small objects with 60,000 RGB images. Here COIL20 is not considered, as this dataset is relatively simple and its classification performances between different methods are very close. For comparison methods, all dimension reduction methods in the clustering experiment are considered except for GNMF, as GNMF only models the decoder and it is not clear to inference the features on test dataset given the learned basis on the training dataset. We also present the state-of-the-art performances on both MNIST and CIFAR-10 at the time of writing as a reference [46], [47]. Note that they are both obtained using convolutional neural networks, which is out of the scope of this paper.

1) Classification Results: The classification results on MNIST and CIFAR-10 are given in Table IV. Specifically, for classification of MNIST, we use the generally split configuration where there are 50,000 images for training, 10,000 images for validation and 10,000 images for test. To be consistent with the network architecture used in the clustering experiment, we use two hidden layers with 500 units for feature learning and a softmax classifier is built on top of the feature for classification, which means the model structure



(a) (b) (c) (d) (e)

Fig. 5. 200 filters learned by different methods on patches of CIFAR-10. (a) LPP. (b) AE. (c) SAE. (D) CAE. (e) GAE.

TABLE IV
CLASSIFICATION ERROR RATES ON MNIST AND CIFAR-10

	MNIST(%)	CIFAR-10(%)
LPP	29.87	58.50
AE	1.29	54.35
SAE	1.21	33.28
CAE	1.18	33.38
GAE	**1.07**	**32.75**
Ranzato et al. [47]	0.64	–
He et al. [48]	–	7.77

is 784-500-500-10. We employ the unsupervised layer-wise pre-training to initialize the two hidden layers of the stacked auto-encoders. Then the stacked auto-encoders and the softmax classifier are fine tuned together by the backpropagation with dropout. The parameters of the fine tuning are selected based on the best performance on the validation dataset.

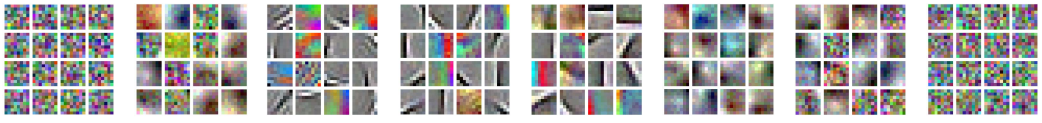
For classification on CIFAR-10, we adopt the same framework suggested in [48], where different feature learning methods are employed for unsupervised feature extraction and then a softmax classifier is trained to classify the unsupervised extracted features. CIFAR has 50,000 training images and 10,000 test images with size $32 \times 32 \times 3$. We split the 50,000 training images into 45,000 training images and 5,000 validation images. For unsupervised feature learning, we randomly extract 160,000 patches with size $8 \times 8 \times 3$ from the training images and learn representations on them. As suggested in [48], we perform local contrast normalization (substract the mean and divide by the standard deviation) to each patch, followed by the whitening process which aims to reduce the feature correlation. In the context of dimension reduction representation learning, all comparing methods learn a 150-dimension representation on the pre-processed patches with the dimension of 192. Note that we use the single-layer architecture for all auto-encoder variants in this experiment, then the network configuration is 192-150. Each 192-dimensional vector in the weight matrix can be regarded as a filter with size $8 \times 8 \times 3$. After unsupervised training of the single-layer architecture, all the 150 filters are applied to the full images using convolution. Thus each image is represented as a $25 \times 25 \times 150$ feature map. The sum pooling is then applied to the quadrants of the feature map, generating the pooled feature map with size $2 \times 2 \times 150$. Finally, it is stretched to a vector of 600 dimension to present each image.

As can be seen in Table IV, our GAE achieve competitive performance compared to the locality preserving methods and auto-encoder variants on both MNIST and CIFAR-10. In addition, it is to be noted that the performance of LPP drops significantly on MNIST for two reasons. One is that LPP has much smaller capacity due to its shallow architecture, the other is that features extracted using LPP are classified directly without the fine-tuning scheme in auto-encoders.

2) Visualization of Filters: We illustrate the filters learned by LPP and variants of auto-encoders as in Figure 5, where all 150 filters are visualized. The filters of LPP are displayed in the descending order of eigenvalues. The results demonstrated that AE can only learn filters with meaningless shapes since no additional regularizer is applied to constrain its hidden space. SAE, CAE and GAE can all lean filters with edges and colors which are believed to be useful for capturing locality features. This explains why these variants can give better classification results over the original AE. Besides, the edge-like filters are learned by CAE and SAE because of the encouragement of saturation in hidden space. The similar filters learned by GAE also reflect the first effect derived by our theoretical analysis. For the filters learned by LPP, the shape reflects more global features as same as PCA, which cannot lead to good generalization performance without further fine tuning. As can be seen, with the same graph regularizer, auto-encoder enables learning of edges while LPP cannot. This is also a complement that GAE bring to the conventional manifold learning methods.

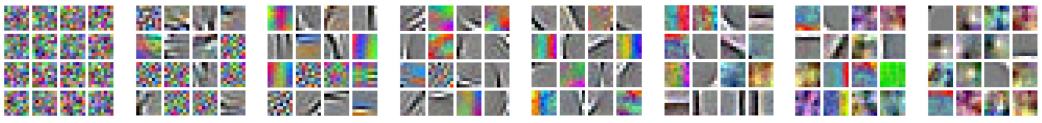
We also try to understand the impact of the hyper-parameters of GAE by visualizing the changing trend of the learned filters. Firstly, we fix the number of the neighbors $k = 3$ and vary the λ in $[0, 1, 3, 5, 10, 30, 100, 300]$ as shown in Figure 6. Secondly, Figure 7 shows the learned filters when k varying in $[0, 1, 2, 3, 5, 10, 15, 20]$ with $\lambda = 3$. For each different parameter setting, we randomly choose 16 filters from the 150 filters for visualization.

Seen from Figure 6, when $\lambda = 0$, the GAE degenerates to the original AE, and the learned filters are with meaningless shapes. As λ increases, the edges and color information gradually emerges from the learned filters, leading to the increasing performance of GAE. This is inspired by the effects of GAE theory. In this interval, the first effect enables the learning of edge-like filters, while the second effect keeps the local property of the data space in an appropriate size. When λ gets larger, i.e. when the graph regularization gets stronger, the neighboring samples are enforced to be over-similar in the representation space, and the representations



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 6. Filters learned by GAE with $k = 3$ and varying λ, 16 filters are randomly chosen for visualization. (a) $\lambda = 0$. (b) $\lambda = 1$. (c) $\lambda = 3$. (D) $\lambda = 5$. (e) $\lambda = 10$. (f) $\lambda = 30$. (g) $\lambda = 100$. (h) $\lambda = 300$.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 7. Filters learned by GAE with $\lambda = 4$ and varying k, 16 filters are randomly chosen for visualization. (a) $k = 0$. (b) $k = 1$. (c) $k = 2$. (D) $k = 3$. (e) $k = 5$. (f) $k = 10$. (g) $k = 15$. (h) $k = 20$.

would be over-saturated. As a result, only the average of the data can be captured, losing the high frequency information. Visually, the edge-like filters gradually become more global: the edge blurs and the region of edge stretches, which indicates the beginning of worsening of performance. When λ continuous growing, some filters becomes random filters. These random filters are different from ones when $\lambda = 0$. They have much larger magnitudes. As a result, all data will be mapped into the same saturated regions and give very small variances in each hidden units, regardless of the reconstruction cost. Obviously, this is an extreme situation of graph regularizer, losing all discriminative information. As can be seen, this trend is driven by the two effects derived from the theory. When λ is fixed and k becomes larger, the graph becomes more densely connected, leading to an over-sized neighborhood. Thus more data in this neighborhood are required to be with similar representations to keep the hidden space with small variances. As a result, the trend for the visualization of filter with respect to hyper-parameter k can be predicted to be similar with that of hyper-parameter λ based on our theory. The results in Figure 7 also validates our hypothesis that the filters become to show more global features when k grows.

In this experiment, we introduce the relationship between the theoretic underlying mechanism of the graph regularizer with the hyper-parameter and the potential performance. This gives us a very directive method for evaluating the performance of the given hyper-parameters, especially the guidance for tuning of the hyper-parameters to better results. This is an advantage of GAE over the LPP for manifold assumption induced representation learning. In relatively large intervals for λ and k, one can see that the filters are learned with edge-like shapes, guaranteeing a robust performance of the method.

VI. Conclusion

In this paper, we propose a novel graph regularized auto-encoder, which can learn a locally invariant representation of the images. A theoretical analysis is carried out to reveal the intrinsic impact of the graph regularization, which is a constraint on the weighted Jacobian matrix of the encoder mapping. We also reveal the underlying effects imposed to the representation space by minimizing the weighted Jacobian matrix. The relationship between the proposed graph regularized auto-encoder with other auto-encoder variants is also discussed based on the theoretical analysis.

Experimental results on image clustering and classification show our method provides a superior option for image representation. Additional visualization results are presented to qualitatively demonstrate the effectiveness of the graph regularization, and validate our theoretical analysis. The influence of the hyper-parameters is also investigated based on the visualization experiments. The future works may focus on investigating the deeper architectures with graph regularization, imposing the graph regularization to the convolutional networks is also possible future works.

References

[1] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, 1997.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[3] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

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

[5] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, vol. 2. Cambridge, MA, USA: MIT Press, 2006.

[6] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.

[7] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

[8] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

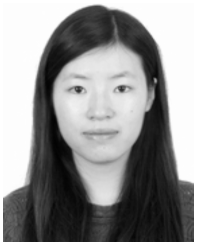
[9] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-Scale Kernel Mach.*, vol. 34, no. 5, pp. 1–41, 2007.

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

[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19. 2007, p. 153.

[12] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 873–880.

[13] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 833–840.
[14] Y. Liao, Y. Wang, and Y. Liu, "Image representation learning using graph regularized auto-encoders," [Online]. Available: http://arxiv.org/abs/1312.0786
[15] K. Jia, L. Sun, S. Gao, Z. Song, and B. E. Shi, "Laplacian auto-encoders: An explicit learning of nonlinear data manifold," *Neurocomputing*, vol. 160, pp. 250–260, Jul. 2015.
[16] P. C. Ng and S. Henikoff, "Sift: Predicting amino acid changes that affect protein function," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3812–3814, 2003.
[17] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 221–228.
[18] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[19] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 1585–1592.
[20] S. Rifai *et al.*, "Higher order contractive auto-encoder," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Germany: Springer-Verlag, 2011, pp. 645–660.
[21] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011.
[22] L. Wang, T. Liu, G. Wang, K. L. Chan, and Q. Yang, "Video tracking using learned hierarchical features," *IEEE Trans. Image Process.*, vol. 24, no. 4, pp. 1424–1435, Apr. 2015.
[23] J. Kuen, K. M. Lim, and C. P. Lee, "Self-taught learning of a deep invariant representation for visual tracking via temporal slowness principle," *Pattern Recognit.*, vol. 48, no. 10, pp. 2964–2982, 2015.
[24] C. Hong, J. Yu, J. Wan, D. Tao, and M. Wang, "Multimodal deep autoencoder for human pose recovery," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5659–5670, Dec. 2015.
[25] S. Ding, L. Lin, G. Wang, and H. Chao, "Deep feature learning with relative distance comparison for person re-identification," *Pattern Recognit.*, vol. 48, no. 10, pp. 2993–3003, Oct. 2015.
[26] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.
[27] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. NIPS*, 1993, pp. 3–10.
[28] 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.
[29] C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proc. NIPS*, 2006, pp. 1137–1144.
[30] I. Goodfellow, Q. Le, A. Saxe, H. Lee, and A. Y. Ng, "Measuring invariances in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 22. 2009, pp. 646–654.
[31] H. Larochelle and Y. Bengio, "Classification using discriminative restricted Boltzmann machines," in *Proc. ICML*, 2008, pp. 536–543.
[32] R. Goroshin and Y. LeCun. (2013). "Saturating auto-encoders." [Online]. Available: http://arxiv.org/abs/1301.3577
[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.
[34] 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.*, New York, NY, USA, 2008, pp. 1096–1103.
[35] G. Alain and Y. Bengio. (2012). "What regularized auto-encoders learn from the data generating distribution." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3563–3593, 2014
[36] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. NIPS*, 2001, pp. 585–591.
[37] A. Elgammal and C.-S. Lee, "Inferring 3D body pose from silhouettes using activity manifold learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jun./Jul. 2004, p. II-681.
[38] X. Cao, B. Ning, P. Yan, and X. Li, "Selecting key poses on manifold for pairwise action recognition," *IEEE Trans. Ind. Informat.*, vol. 8, no. 1, pp. 168–177, Feb. 2012.
[39] R. Hettiarachchi and J. F. Peters, "Multi-manifold LLE learning in pattern recognition," *Pattern Recognit.*, vol. 48, no. 9, pp. 2947–2960, 2015.
[40] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. Washington, DC, USA, 2006, pp. 1735–1742.
[41] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 2012, pp. 639–655.
[42] W. Yu, G. Zeng, P. Luo, F. Zhuang, Q. He, and Z. Shi, "Embedding with autoencoder regularization," in *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2013, pp. 208–223.
[43] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.
[44] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 16. 2003, pp. 234–241.
[45] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, nos. 2579–2605, p. 85, 2008.
[46] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[47] K. He, X. Zhang, S. Ren, and J. Sun. (2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385
[48] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.



Yiyi Liao (S'15) received the B.S. degree in automation from Xian Jiaotong University in 2013. She is currently pursuing the Ph.D. degree with the Institute of Cyber-Systems and Control, Department of Control Science and Engineering, Zhejiang University. Her research interests include machine learning, computer vision, and mobile robotics.

Yue Wang (S'16) received the B.Sc. degree in communication engineering from the Zhejiang University of Technology, Hangzhou, China, in 2011, and the Ph.D. degree in control science and engineering, Zhejiang University, Hangzhou, China, in 2016. He was a Joint Ph.D. Student with Stanford University, Stanford, CA, USA, funded by the China Scholarship Council. His research interests include mobile robot, machine learning, and big data analysis.

Yong Liu (M'11) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Zhejiang, China, in 2001 and 2007, respectively. He is currently an Associate Professor with the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University. His current research interests include machine learning, robotics vision, and information fusion. He has authored over 30 research papers in machine learning, computer vision, information fusion, and robotics.