Pattern Recognition 000 (2017) 1–16

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Deep unsupervised learning with consistent inference of latent representations

Jianlong Chang[a,b], Lingfeng Wang[a], Gaofeng Meng[a,*], Shiming Xiang[a,b], Chunhong Pan[a]

[a] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Article history:
Received 6 February 2017
Revised 1 October 2017
Accepted 16 October 2017
Available online xxx

Keywords:
Deep unsupervised learning
Consistent inference of latent representations

ABSTRACT

Utilizing unlabeled data to train deep neural networks (DNNs) is a crucial but challenging task. In this paper, we propose an end-to-end approach to tackle this problem with consistent inference of latent representations. Specifically, each unlabeled data point is considered as a seed to generate a set of latent labeled data points by adding various random disturbances or transformations. Under the expectation maximization framework, DNNs can be trained in an unsupervised way by minimizing the distances between the data points with the same latent representations. Furthermore, several variants of our approach can be derived by applying regularized and sparse constraints during optimization. Theoretically, the convergence of the proposed method and its variants are fully analyzed. Experimental results show that the proposed approach can significantly improve the performance on various tasks, including image classification and clustering. Such results also indicate that our method can guide DNNs to learn more invariant feature representations in comparison with traditional unsupervised methods.

1. Introduction

Thanks to its extraordinary capability of feature representation, deep neural networks (DNNs) trained via supervised methods have achieved significant successes in various pattern recognition problems, including image classification [1–4], image segmentation [5–8], object detection [9–12], face recognition [13–16], speech recognition [17–20]. These achievements verify that the supervised methods can provide explicit guidance to DNNs for feature learning.

A straightforward way to train excellent DNNs is increasing the number of labeled training data [21]. However, it is laborious and expensive to obtain a large amount of labeled data. Since training DNNs can be considered as an optimal problem with a group of parameters [22], it is reasonable to learn deep models by pre-training these models with a mass of unlabeled data and fine-tuning the models for the specific classification task with labeled data. Along this line, in the literature, several unsupervised methods have been presented, such as the deep belief networks [23,24], the autoencoders [25–27], the deep Boltzmann machine [28], the convolutional autoencoder [29], and so on. The results achieved by these methods have demonstrated that significant improvements can be obtained in various tasks, such as classification [25,27,28], recognition [30–32], retrieval [33–36], and so on.

Most unsupervised methods aim to pre-train DNNs by reconstructing inputs, such as the restricted Boltzmann machine [37], the autoencoder [25], and so on. Technically, what the entire network learns is a non-linear mapping that is able to transform data points identically to be themselves. DNNs trained by these methods consist of two different parts: encoding and decoding. The encoding part is designed to learn a function to map data points into a latent space and yields feature representations for inputs. Meanwhile, the decoding part can be treated as an inverse operator of encoding, with which the inputs could be reconstructed by using the learned feature representations. The encode-decode architecture is a typically unsupervised way to train the DNNs. Furthermore, these methods can be assisted to learn more robust representations by associating with some frequently-used regularization terms, including the Frobenius norm [38], sparsity [39], and so on. Different from adding regularization to loss function, a denoising function will be learned by reconstructing the original clean data from the corrupted input data [40].

Although significant performances have been achieved over the past few years, there are two crucial problems that still need to be tackled. First, it is difficult to pre-train DNNs with a large number of layers. An essential reason is that decoding layers may deepen the depth of the networks during pre-training phase [29,41,42].

* Corresponding author.
E-mail addresses: jianlong.chang@nlpr.ia.ac.cn (J. Chang), lfwang@nlpr.ia.ac.cn (L. Wang), gfmeng@nlpr.ia.ac.cn (G. Meng), smxiang@nlpr.ia.ac.cn (S. Xiang), chpan@nlpr.ia.ac.cn (C. Pan).

https://doi.org/10.1016/j.patcog.2017.10.022
0031-3203/© 2017 Published by Elsevier Ltd.

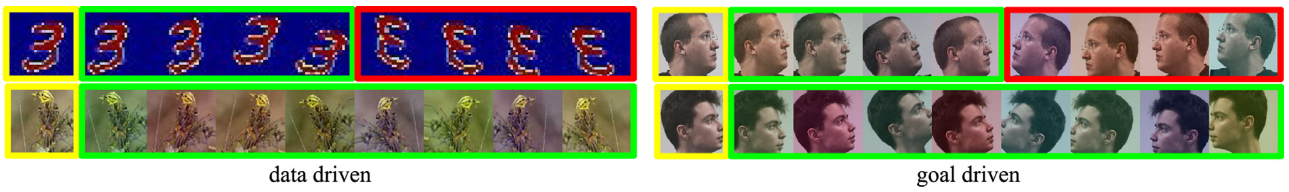

Fig. 1. Exemplary samples generated based on the different datasets and goals. **Left:** Identical task (classification) on the different datasets (Top: handwritten digits. Down: animal images.). **Right:** Different tasks (Top: attitude estimation. Down: identification recognition.) on the identical dataset (face dataset). The original seed data is on the left of each line and depicted in the yellow box. The images in the green and the red boxes represent the reasonable and the improper generated samples, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

However, a universal method for training very deep networks is under the way of exploring [4]. Second, the traditional pre-training methods are model-specified, which means that different strategies are required to pre-train various DNNs. Generally, more general method is desired for facilitating the usage of unsupervised deep feature learning.

To solve the aforementioned issues in pre-training DNNs, we develop a novel unsupervised leaning method, *i.e.*, **C**onsistent **I**nference of **L**atent **R**epresentations based learning (CILR). Specifically, inspired by the supervised learning, we introduce consistent inference of latent representations to generate latent labeled data points to pre-train deep models first. Then, CILR is derived to pre-train DNNs by minimizing the distance between latent labeled data points which possess the same representations in a latent space. According to the triangle inequality, an alternating iterative algorithm is developed to optimize our model. Furthermore, several variants of CILR are derived by applying regularization and sparsity constraint.

To sum up, the main contributions of this work are:

- A novel end-to-end unsupervised method is presented to pre-train DNNs. By minimizing the distances between the data points with the same latent representations in the learned high-level representation space, DNNs are pre-trained by alternately forward calculating latent representations of unlabeled data and backward optimizing the parameters.
- By adding some regularization terms to our model, a series of variants of the proposed method can be obtained. In theory, for such extensions, the universal form of the regularization term is analysed.
- Theoretically, the convergence of CILR and its variants is verified. Experimental results on numerous tasks with various DNNs indicate that our approach is effective for pre-training DNNs and learning representations.

The remainder of this paper is organized as follows. We first review existing techniques related to deep unsupervised learning in Section 2. Section 3 presents the details of the proposed method and its variants. The convergence of our approach is theoretically analysed in Section 4. The overfitting risk of our method is handled in Section 5. The performance of our algorithm is sufficiently discussed in Section 6. Conclusions are drawn in Section 7.

2. Related work

Various deep unsupervised learning methods with different structures have been presented in the literature. Traditional pre-training algorithms pre-train fully connected neural networks via layer-wise pre-training strategy, such as the deep belief network [23] and the autoencoder [25]. Specifically, the fully connected layers are pre-trained by reconstructing inputs in these methods. Furthermore, by imposing the sparse constraints to encoding layers, more robust feature representations can be learned in the sparse autoencoder [39]. Similar to the sparse autoencoder, the contractive autoencoder [38] attempts to learn the invariant feature representations of the inputs by introducing the Frobenius norm to encoding layers. The denoising autoencoder [40] successfully learns a denoising function by reconstructing the original clean data from the corrupted inputs. Although it is feasible to pre-train very deep fully connected networks via these methods, the pre-training procedures are cumbersome. Compared with these methods, in this paper, we attempt to develop an end-to-end pre-training strategy, which can simplify the pre-training procedure and achieve the promising performance.

As a natural framework to process images and videos, some previous works have been proposed to pre-train convolutional neural networks (CNNs). By applying the up-pooling operator, the convolutional autoencoder [29] has been developed to pre-train CNNs by reconstructing input images. Similar to the autoencoder [25], a variety of regularization terms like Frobenius norm and sparse constraint can be employed to learn more robust and meaningful feature representations. In addition, the deconvolutional network [41] pre-trains CNNs based on the decomposition under a sparse constraint with unlabeled data only. As a result, CNNs are pre-trained in an unsupervised learning way. Inspired by the sparse coding algorithm, Ranzato et al. [43] presented an unsupervised method to learn a group of sparse feature detectors that are invariant to small shifts and distortions. By initializing these detectors to convolutional kernels, more robust feature representations can be yielded. However, these methods mainly focus on shallow CNNs. To pre-train more deeper CNNs (*e.g.*, residual networks [4]), an end-to-end approach is desiderated.

Recently, most unsupervised or semi-supervised methods have been presented to improve the capability of DNNs when labeled data is limited. Inspired by principal component analysis, Valpola et al. [44] proposed a semi-supervised method to train deep neural networks by including the lateral shortcut connections from the encoder part to decoder part at each layer. Furthermore, Pezeshki et al. [45] adjusted connections between individual parts of deep models to enhance performance by learning their relationships. Experimentally, they have demonstrated the promising results on the MNIST classification task. Zhao et al. [42] presented an architecture which provides an uniform strategy to supervised, semi-supervised and unsupervised learning. Specifically, this model consists of convolutional and deconvolutional network parts to encode inputs and produce reconstruction, respectively. While the performance has been significantly improved, these methods have to learn superfluous auxiliary parameters that may degrade the pre-training efficiency and effectiveness.

3. Consistent Inference of Latent Representations Based Learning

In this section we detail the proposed learning method called CILR. First, latent representation consistent inference is introduced to generate latent labeled samples in Section 3.1. Then, the objec-

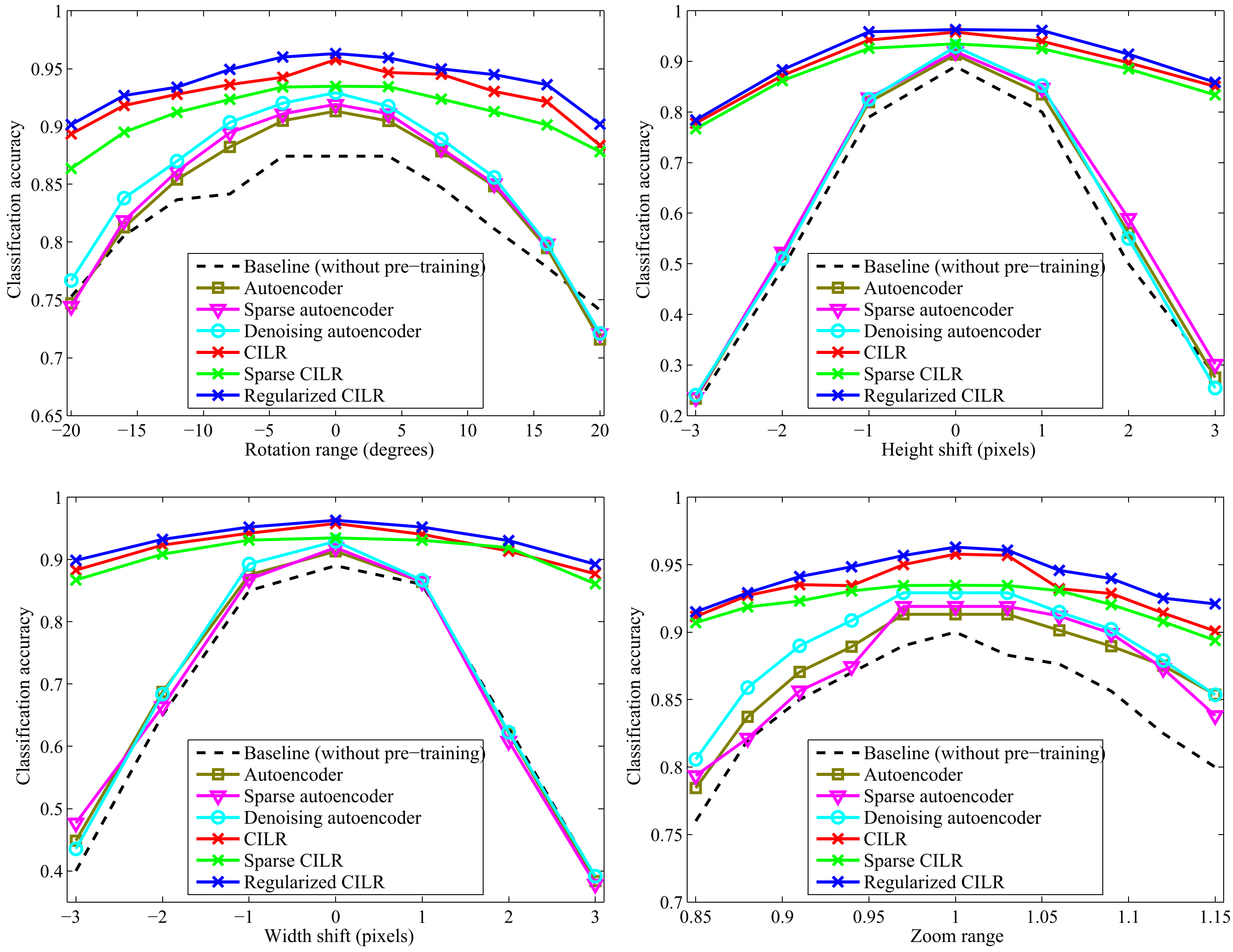

Fig. 2. The invariance of the learned feature representations by the fully connected neural network with 1000 labeled samples (without data augmentation).

tive of CILR is derived in Section 3.2. Finally, some extensions of CILR are developed in Section 3.3.

3.1. Creating latent labeled data

The inputs of the pre-training phase are a large amount of unlabeled data. For an unlabeled data point $\mathbf{x}$, inspired by the data augmentation in the supervised learning, we randomly generate n samples to form a latent labeled data set $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, where $\mathbf{x}_i$ is generated by adding random disturbance to $\mathbf{x}$. According to the way of generating $\mathbf{x}_i$, it is a natural inference termed as **Consistent Inference of Latent Representations** that $\mathbf{x}$ and each $\mathbf{x}_i$ have the consistent latent representations, since they belong to the same classes. After that, a latent labeled data set is yielded. Expediently, we define a family of functions $\{T_\varepsilon | \varepsilon \in \mathcal{A}\}$, where $\mathcal{A}$ is a set of transformations, such as rotation, shift, adding Gaussian noise, and so on. For each transformation $\varepsilon \in \mathcal{A}$, $\mathbf{x}$ and $T_\varepsilon(\mathbf{x})$ possess the consistent latent representations.

The definition of $\mathcal{A}$ is data and goal driven, *i.e.*, $\mathcal{A}$ varies with different datasets and objectives of tasks. For classification, it is of an identical class that flipping bird images horizontally. In contrast to the bird images, flipped handwritten digit images like “3” and “4” are severely different from the original images “3” and “4”, respectively. This indicates that $\mathcal{A}$ is data driven. Furthermore, for the identification recognition and the attitude estimation on an identical dataset, it is necessary that the latent labeled samples are generated in goal driven way. Because DNNs should learn different feature representations for diverse goals. Some exemplary samples generated from the data and goal driven are illustrated in Fig. 1.

3.2. CILR

Given a latent labeled data set $\{\mathbf{x}_i^k\}_{i=1}^n$, $\forall\ \mathbf{x}_i \in \{T_\varepsilon(\mathbf{x}) | \varepsilon \in \mathcal{A}\}$, where $\mathbf{x}_i^k$ represents the ith generated samples from the seed data point $\mathbf{x}^k$. If the label $\mathbf{l}^k$ of $\mathbf{x}^k$ is observable, the objective of supervised learning methods is to learn a function g, which maps data point $\mathbf{x}^k$ to label $\mathbf{l}^k$. It is logical that g satisfies the following equation:

$$g(\mathbf{x}_i^k) = \mathbf{l}^k = g(\mathbf{x}_j^k). \tag{1}$$

In contrast to the supervised learning, however, the label information $\mathbf{l}^k$ is unknown in the unsupervised learning. As the aforementioned consistent inference of latent representations, the feature representations of $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$ are naturally consistent. Based on this observation, we attempt to train DNNs that can map the data points with same latent representations to an identical point in the learned latent space. Namely, $f(\mathbf{x}_i^k) = f(\mathbf{x}_j^k)$ is satisfied for

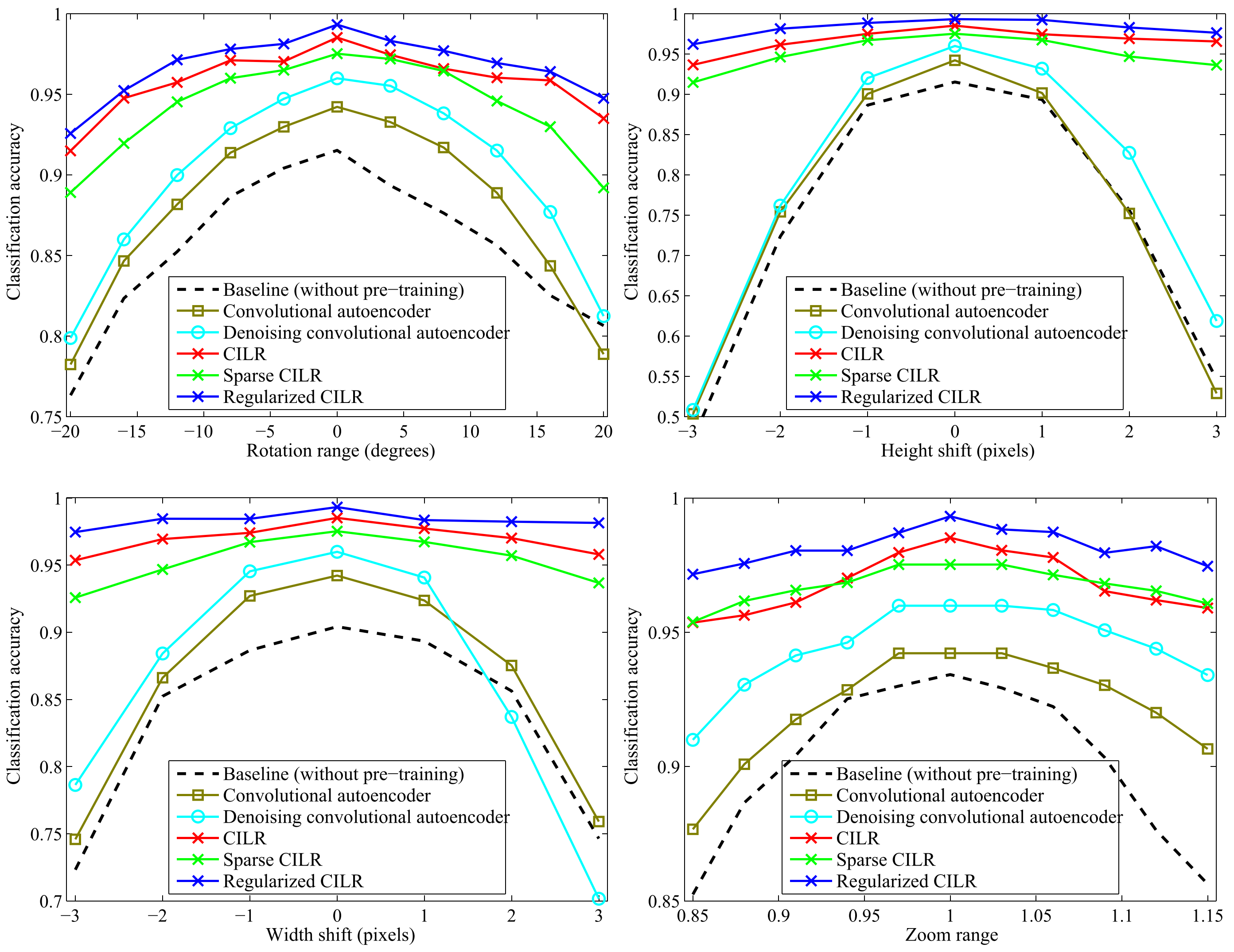

Fig. 3. The invariance of the learned feature representations by the convolutional neural network with 1000 labeled samples (without data augmentation).

the learned function f which is parameterized by DNNs. Formally, DNNs can be trained through minimizing the following loss function:

$$\ell(f) = \sum\nolimits_k d(f(\mathbf{x}^k), f(T_\varepsilon(\mathbf{x}^k))), \tag{2}$$

where $d(\cdot, \cdot)$ is a distance defined in a latent space and $T_\varepsilon(\mathbf{x}^k)$ represents a generated sample. A feasible way to optimizing Eq. (2) is finding an upper bound of Eq. (2) and optimizing the original loss function $\ell(f)$ by minimizing the upper bound. To this end, we introduce $\mathbf{y}^k$ that represents the latent feature representation of $\mathbf{x}^k$. According to the triangle inequality, we have:

$$\ell(f) \le \sum\nolimits_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k))). \tag{3}$$

Specifically, if $\mathbf{y}^k$ is available, Eq. (3) can be regarded as a general objective function in the supervised methods. That is, the upper bound of $\ell(f)$ is a generalized description for both supervised and unsupervised learning.

An alternating iterative optimization algorithm is developed to optimize the upper bound described in Eq. (3). That is, given f, $\mathbf{y}^k$ can be calculated through forward propagation algorithm. When $\mathbf{y}^k$ is observable, optimizing the function f is just a supervised problem and a variety of supervised learning methods can be employed to tackle it. The proposed algorithm is summarized as Algorithm 1.

Algorithm 1 Consistent Inference of Latent Representations Based Learning (CILR).

```
Input: Unlabeled data set $\{\mathbf{x}^k\}$, deep neural network $\mathfrak{F}$, a set of transformations $\mathcal{A}$.
Output: Pre-trained network $f$.
  Yield $f_0$ by initializing $\mathfrak{F}$, $t = 0$
  while $f_t \neq f_{t-1}$ or $t == 0$ do
    $\mathbf{y}^k := f_t(\mathbf{x}^k)$
    $f_{t+1} := \underset{f}{\mathrm{argmin}} \sum_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k)))$
    $t := t + 1$
  end while
  return $f_t$
```

3.3. Extensions

Similar to the autoencoder [25], some variants of CILR can be obtained by applying frequently-used constraint, such as regularization constraint and sparsity constraint.

3.3.1. Regularized CILR

Theoretically, in the supervised learning tasks, reducing the structural error can improve the generalization capability of mod-

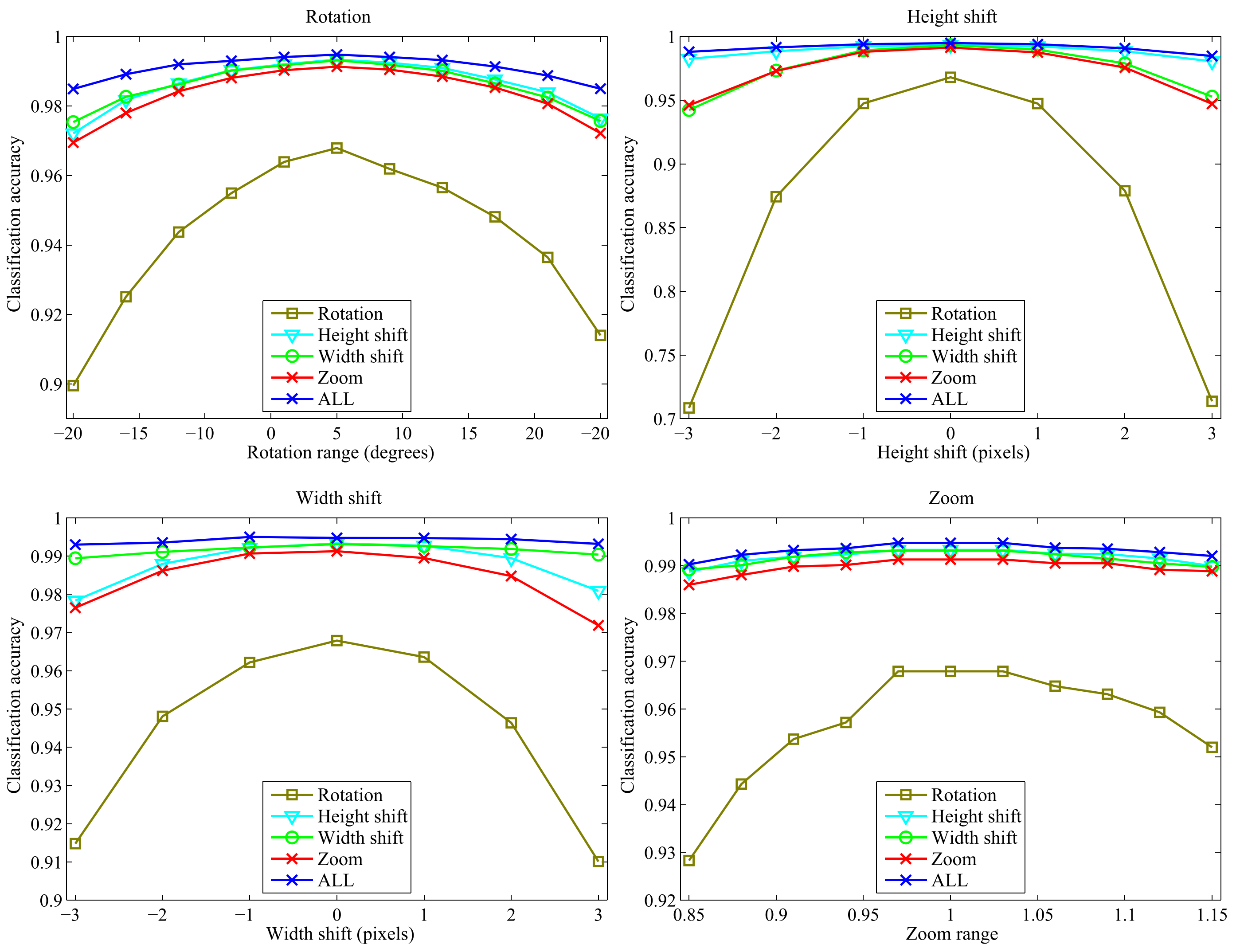

Fig. 4. The contributions of different transformations.

els [46]. Empirically, Krizhevsky et al. [1] have found that the weight decay utilized in the AlexNet is not merely a regularizer, it can reduce the training error of models. Inspired by such observations, for training better models, the weight decay is added in the objective function of CILR to assist pre-training. We name it regularized CILR that the loss function is formulated as:

$$\begin{aligned}\ell_R(f) &= \sum\nolimits_k d(f(\mathbf{x}^k), f(T_\varepsilon(\mathbf{x}^k))) + \lambda \parallel \mathbf{w} \parallel_2^2, \\ &\le \sum\nolimits_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k))) + \lambda \parallel \mathbf{w} \parallel_2^2,\end{aligned} \tag{4}$$

where $\mathbf{w}$ represents the model parameter vector inside the function f, and $\mathbf{w}_i$ stands for the i^{th} entity of $\mathbf{w}$[1]. $\parallel \cdot \parallel_2^2$ represents l_2-norm of vectors, and λ is a regularization parameter to control the weight of the weight decay.

Specifically, when $\mathbf{y}^k$ is observed, the gradient of $\ell_R(f)$ with respect to weight $\mathbf{w}_i$ can be calculated as follows:

$$\frac{\partial \ell_R}{\partial \mathbf{w}_i} = \frac{\partial \ell^{up}}{\partial \mathbf{w}_i} + 2\lambda \mathbf{w}_i, \tag{5}$$

where $\ell^{up} = \sum_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k)))$, and $\frac{\partial \ell^{up}}{\partial \mathbf{w}_i}$ is the gradient of ℓ^{up} with respect to $\mathbf{w}_i$. Compared with CILR, the regularized CILR not only attempts to map similar samples to a point in latent space, but limits the complexity of the model to promote the generalization.

[1] $\mathbf{w}_i$ corresponds to the weight of a pair of connected nodes in neural networks.

3.3.2. Sparse CILR

The sparsity can be utilized to discover interesting structure in data and to learn the more robust feature representations of data points [43]. Inspired by Ng [39], we develop an variant of CILR termed sparse CILR which attempts to learn sparse feature representations by adding a sparse penalty term in loss function. More formally, the objective function of the sparse CILR is formulated as:

$$\begin{aligned}\ell_S(f) &= \sum\nolimits_k d(f(\mathbf{x}^k), f(T_\varepsilon(\mathbf{x}^k))) + \beta \sum\nolimits_j KL(\rho \parallel \hat{\rho}_j), \\ &\le \sum\nolimits_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k))) \\ &\quad + \beta \sum\nolimits_j KL(\rho \parallel \hat{\rho}_j),\end{aligned} \tag{6}$$

where ρ is a sparsity parameter, and β is a regularization parameter to control the weight of the sparsity. The average activation of the output unit j can be calculated by:

$$\hat{\rho}_j = MEAN\Big(\sum\nolimits_k \mathbf{a}_j^k\Big), \tag{7}$$

where $\mathbf{a}^k = \frac{f(\mathbf{x}^k)+f(T_\varepsilon(\mathbf{x}^k))}{2}$, $\mathbf{a}_j^k$ is the jth component of $\mathbf{a}^k$, and $MEAN(\cdot)$ is a mean function. In addition, $KL(\rho \parallel \hat{\rho}_j)$ represents the

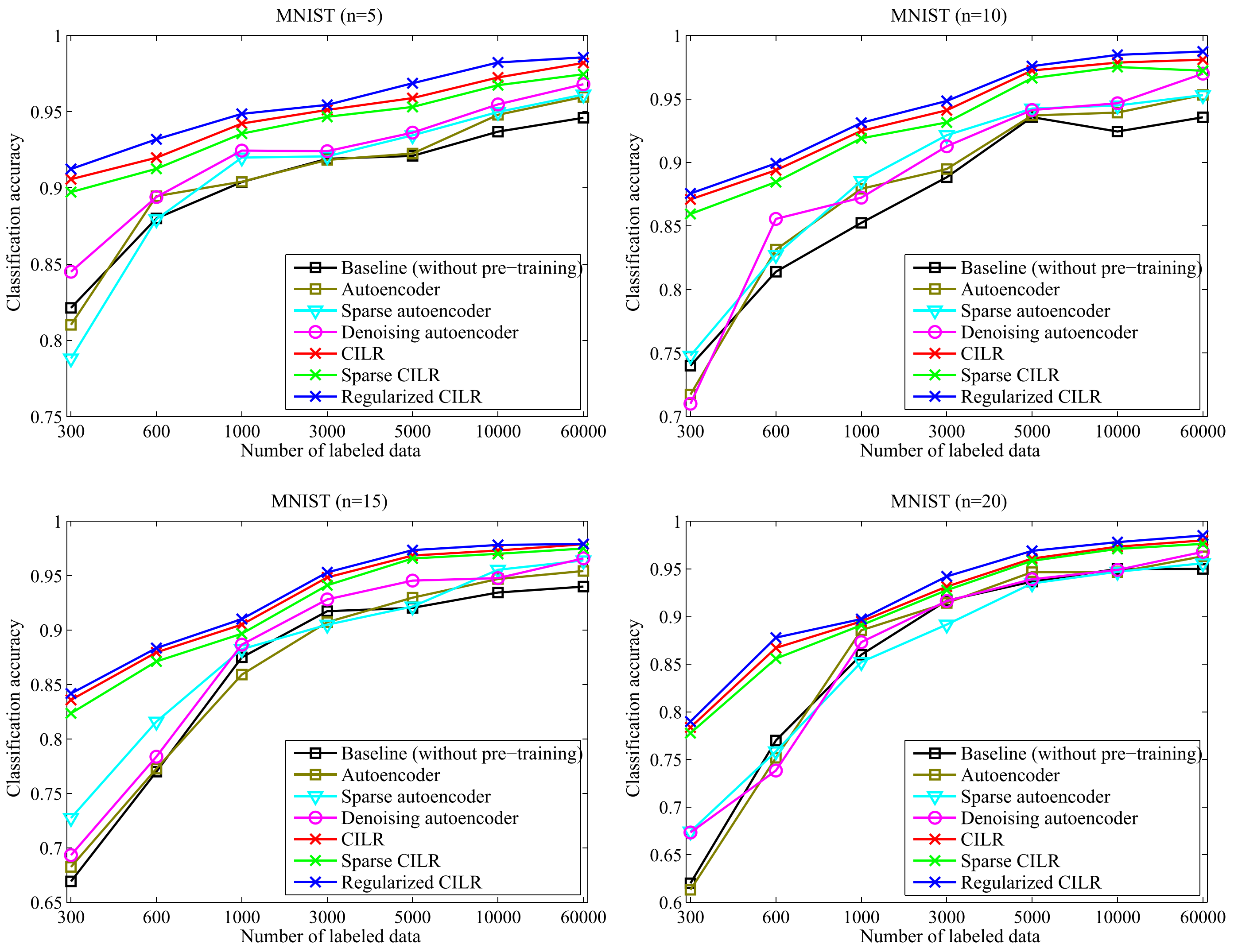

Fig. 5. The results of MNIST classification obtained from a variety of models with different depths. *n* represents the number of layers added in the networks (without data augmentation in fine-tuning).

Kullback–Leibler divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$, and can be formulated as:

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}. \tag{8}$$

Specifically, when $\mathbf{y}^k$ is observed, the gradient of $\ell_S(f)$ with respect to weight $\mathbf{w}_i$ can be calculated as follows:

$$\frac{\partial \ell_S}{\partial \mathbf{w}_i} = \frac{\partial \ell^{up}}{\partial \mathbf{w}_i} + \beta \sum_j \left(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j} \right) \frac{\partial \hat{\rho}_j}{\partial \mathbf{w}_i}, \tag{9}$$

where $\frac{\partial \ell^{up}}{\partial \mathbf{w}_i}$ is the gradient of the upper bound of Eq. (3) with respect to $\mathbf{w}_i$, and $\frac{\partial \hat{\rho}_j}{\partial \mathbf{w}_i}$ is the gradient of the $\hat{\rho}_j$ (Eq. (7)) with respect to $\mathbf{w}_i$. As described in Eq. (6), contrary to CILR and regularized CILR, sparse CILR not merely attempts to map similar samples to a point in latent space, it makes the obtained feature representations sparse which can enhance the robustness of the learned feature representations [39].

4. Convergence analyses

In this section, the convergence of CILR and its variants is theoretically analyzed. As previously described, f_t is a learned function from a function family $\mathfrak{F}$ which is only determined by the structure of DNNs. It is one-to-one correspondence between the network and the parameters of the network, when the structure is determined. That is, each function f_t can be considered as a point (the parameters of the networks) in high-dimensional function space. Therefore, the learning procedure described in Algorithm 1 is a process of finding the limit point from a sequence of points $\{P_{f_0}, P_{f_1}, \cdots, P_{f_i}, \cdots\}$, where P_{f_i} represents the parameters of the function f_i.

For a deep neural network, a function family $\mathfrak{F}$ is determined by its structure. We define that f_t represents the optimal function of tth iteration as described in Algorithm 1. A sequence of function $\{f_t \in \mathfrak{F}\}$ will be yielded during the pre-training procedure. Therefore, the convergence of CILR is the convergence of the sequence $\{f_t\}$. We have the following theorem:

Theorem 1. *The sequence $\{f_t\}$ is convergent for distance $d(\cdot, \cdot)$.*

Proof. It is permanently true that $\ell(f_t)$ is nonnegative, because $\ell(f_t)$ is a summation of distance $d(\cdot, \cdot)$ as the definition described in Eq. (2). In addition, according to the triangle inequality, $\ell(f_{t+1})$ has an upper bound:

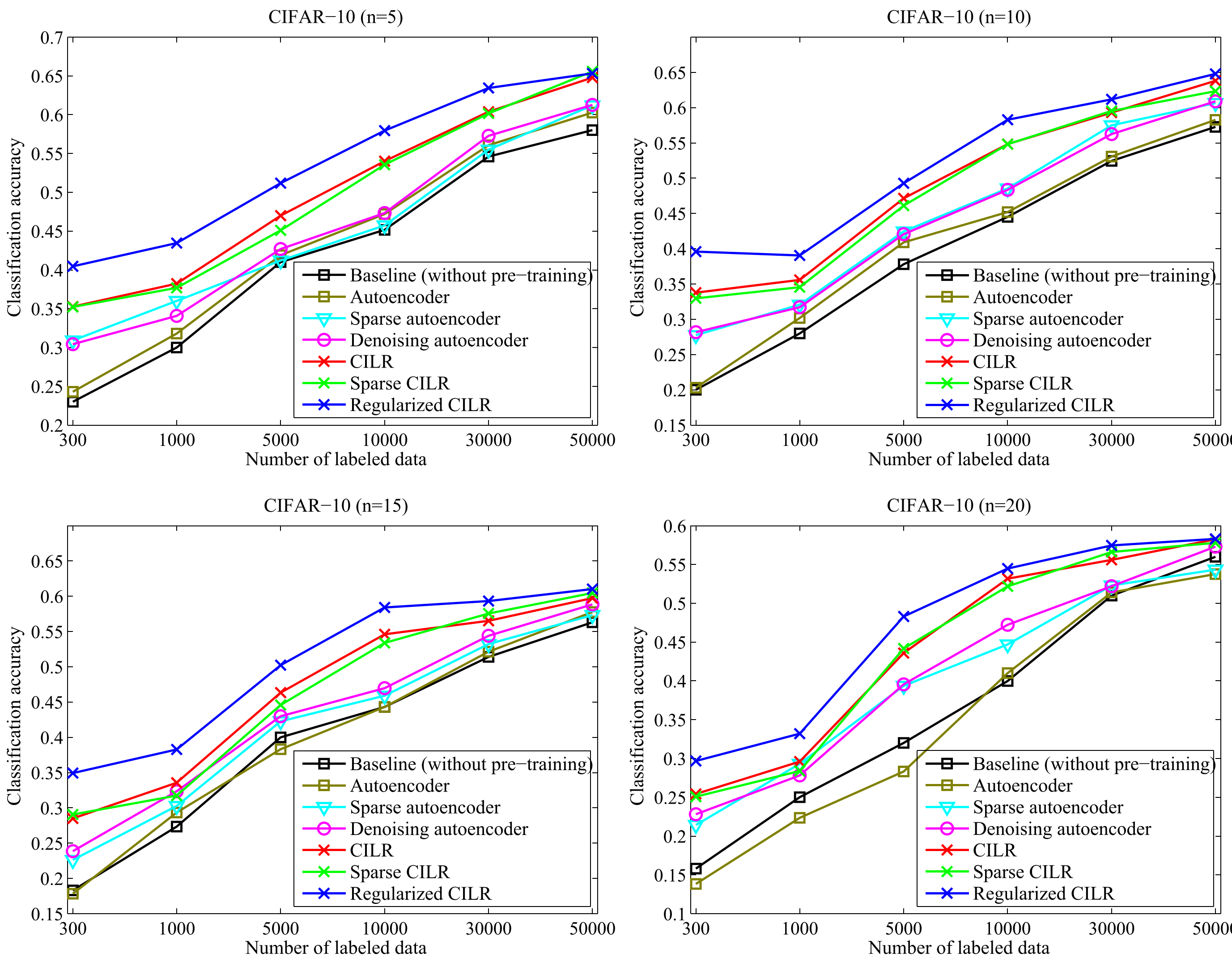

Fig. 6. The results of CIFAR-10 classification obtained from a variety of models with different depths. n represents the number of layers added in the networks (without data augmentation in fine-tuning).

$$\ell(f_{t+1}) = \sum_k d(f_{t+1}(\mathbf{x}^k), f_{t+1}(T_\varepsilon(\mathbf{x}^k))),$$
$$\leq \sum_k d(f_{t+1}(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f_{t+1}(T_\varepsilon(\mathbf{x}^k))), \tag{10}$$

where $\mathbf{y}^k$ is calculated from $\mathbf{y}^k := f_t(\mathbf{x}^k)$. Since f_{t+1} is the optimal function learned by minimizing the $\sum_k d(f_t(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f_t(T_\varepsilon(\mathbf{x}^k)))$, we have:

$$\ell(f_{t+1}) \leq \sum_k d(f_t(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f_t(T_\varepsilon(\mathbf{x}^k))), \tag{11}$$

where $d(f_t(\mathbf{x}^k), \mathbf{y}^k) = d(\mathbf{y}^k, \mathbf{y}^k)$ is satisfied. According to the basic property of distance, $d(\mathbf{y}^k, \mathbf{y}^k) = 0$ is permanently true. Then, we have the following inequation:

$$\ell(f_{t+1}) \leq \sum_k d\left(\mathbf{y}^k, f_t(T_\varepsilon(\mathbf{x}^k))\right), \tag{12}$$

where $\mathbf{y}^k = f_t(\mathbf{x}^k)$. By the definition of $\ell(f_t)$, we have:

$$\ell(f_{t+1}) \leq \ell(f_t). \tag{13}$$

There are two conclusions that can be inferred from the aforementioned proof: $\ell(f_t)$ is monotonically decreasing and has a lower bound 0. According to the mathematical theorem, the sequence of values $\ell(f_t) = \sum_k d(f_t(\mathbf{x}^k), f_t(T_\varepsilon(\mathbf{x}^k)))$ has a limit point, and f_t converges monotonically to a fixed function f. These indicate that $\ell(f_t)$ decreases gradually at each iteration, until $\ell(f_t) = \ell(f_{t+1})$ is satisfied and a fixed function $f = f_t = f_{t+1}$ is obtained. □

Now we consider the convergence of the regularized CILR and the sparse CILR. To analyze the convergence of the variants of CILR, more generally, we prove the following corollary of the Theorem 1 firstly:

Corollary 1. *For an arbitrary function $h(\cdot)$, if $h(\cdot)$ has a lower bound and the update formula in the Algorithm 1 is modified to $f_{t+1} := \text{argmin}_f \sum_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k))) + h(f)$, the sequence $\{f_t\}$ is convergent for distance $d(\cdot, \cdot)$ yet.*

Proof. It is still permanently true that $\ell(f_t)$ has a lower bound, because $h(\cdot)$ has a low bound and $\sum_k d(f(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f(T_\varepsilon(\mathbf{x}^k)))$ has a low bound 0.

Similar to the proof in Theorem 1, we have the following inequations:

$$\ell(f_{t+1}) = \sum_k d(f_{t+1}(\mathbf{x}^k), f_{t+1}(T_\varepsilon(\mathbf{x}^k))) + h(f_{t+1}),$$
$$\leq \sum_k d(f_{t+1}(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f_{t+1}(T_\varepsilon(\mathbf{x}^k)))$$
$$+ h(f_{t+1}), \quad // \; Eq.\ (10)$$
$$\leq \sum_k d(f_t(\mathbf{x}^k), \mathbf{y}^k)$$
$$+ d(\mathbf{y}^k, f_t(T_\varepsilon(\mathbf{x}^k))) + h(f_t), \quad // \; Eq.\ (11)$$
$$= \sum_k d(f_t(\mathbf{x}^k), f_t(\mathbf{x}^k) + d(f_t(\mathbf{x}^k), f_t(T_\varepsilon(\mathbf{x}^k))) + h(f_t),$$

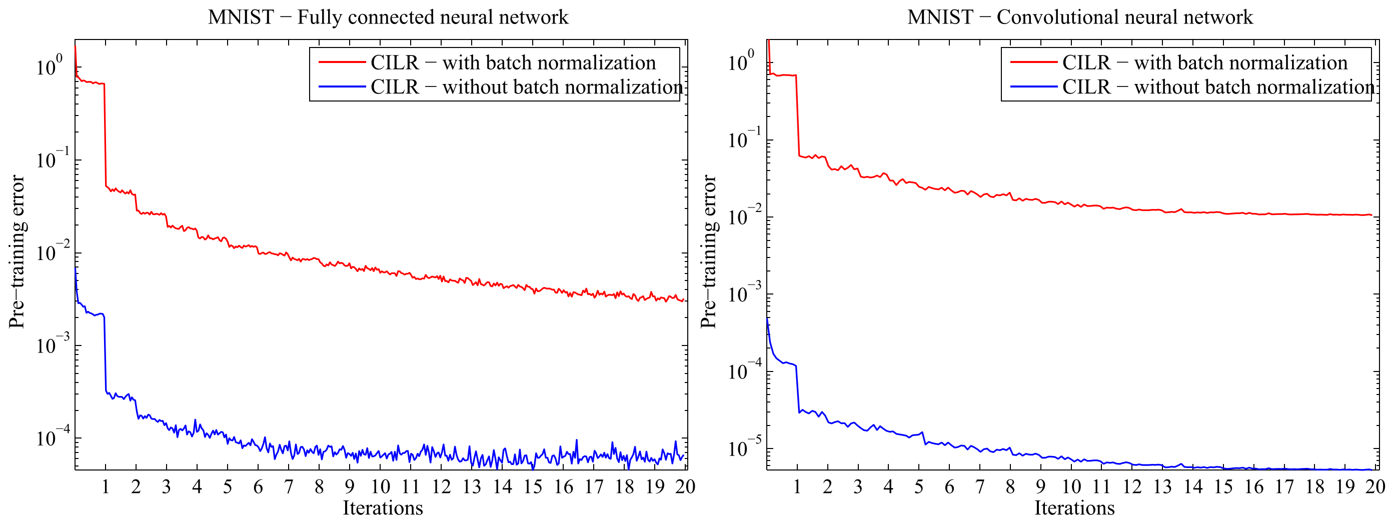

Fig. 7. Pre-training error of CILR on MNIST with/without batch normalization.

$$= \sum_k d(f_t(\mathbf{x}^k), f_t(T_\varepsilon(\mathbf{x}^k))) + h(f_t),$$
$$= \ell(f_t). \tag{14}$$

There are two conclusions which can be inferred from the aforementioned proof: $\ell(f_t)$ is monotonically decreasing and has a lower bound. According to the mathematical theorem, the sequence of values $\ell(f_t) = \sum_k d(f_t(\mathbf{x}^k), f_t(T_\varepsilon(\mathbf{x}^k))) + h(f_t)$ has a limit point, and f_t converges monotonically to a fixed function f. □

As the definitions in Eqs. (4) and (6), the regularized CILR and the sparse CILR can be treated as two particular cases in Corollary 1. Furthermore, we have the following lemma:

Lemma 1. *If $\lambda \geq 0$, $\beta \geq 0$, $0 \leq \rho \leq 1$ are met, the functions $h_R(f) = \lambda \parallel \mathbf{w} \parallel_2^2$ and $h_S(f) = \beta \sum_j KL(\rho \parallel \hat{\rho}_j)$ have lower bound.*

Proof. For $h_R(f)$:

$$h_R(f) = \lambda \parallel \mathbf{w} \parallel_2^2 = \lambda \sum_i \mathbf{w}_i^2 \geq 0. \tag{15}$$

For $h_S(f)$:

$$h_S(f) = \beta \sum_j \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j},$$
$$= \beta \sum_j (\rho \log \rho + (1-\rho)\log(1-\rho))$$
$$- (\rho \log \hat{\rho}_j + (1-\rho)\log(1-\hat{\rho}_j)). \tag{16}$$

Because the sigmoid activation function is utilized, $0 \leq \hat{\rho}_j \leq 1$ is met. According to the Gibbs' inequality, $h_S(f) \geq 0$ is satisfied. □

Follow the analyses in Corollary 1 and Lemma 1, the convergence of the regularized CILR and the sparse CILR can be demonstrated as follows:

Corollary 2. *If the condition of Lemma 1 is met, the regularized CILR and the sparse CILR is convergent for distance $d(\cdot, \cdot)$.*

Proof. According to the Lemma 1, $h_R(f) = \lambda \parallel \mathbf{w} \parallel_2^2$ and $h_S(f) = \beta \sum_j KL(\rho \parallel \hat{\rho}_j)$ have the lower bound 0. That is, the condition in Corollary 1 is satisfied. In theory, the regularized CILR and the sparse CILR are convergent. □

Similar to the expectation maximization algorithm [47], the Theorem 1 does not quite imply that the proposed method converges to a global optimal value, because it is indeterminate whether the result of optimizing $\sum_k d(f_t(\mathbf{x}^k), \mathbf{y}^k) + d(\mathbf{y}^k, f_t(T_\varepsilon(\mathbf{x}^k)))$ is the global minimum at each iteration.

Table 1
The structures of the modeled neural networks for MNIST classification.

Fully Connected Neural Network	Convolutional Neural Network
Input 28×28 monochrome image	Input $3 \times 32 \times 32$ RGB image
Full 1024 BN ReLU	5×5 conv. 128 BN ReLU
Full 512 BN ReLU	2×2 max-pooling strider 2
Full 256 BN ReLU	5×5 conv. 256 BN ReLU
Full 128 BN ReLU	2×2 max-pooling strider 2
	Full 512 BN ReLU
10-way softmax	

In the above analyses, the distance $d(\cdot, \cdot)$ and function family $\mathfrak{F}$ are alterable. This indicates that the proposed method is convergent for arbitrary $d(\cdot, \cdot)$ and $\mathfrak{F}$. Theoretically speaking, the loss functions of supervised learning defined by distance can be employed as the objective function $\ell(f)$ in the proposed unsupervised method directly, such as mean squared error, mean absolute error, cosine proximity, and so on. It is encouraging that this property builds a bridge between supervised and unsupervised learning. That is, a lot of achievements in the field of the supervised learning can be transplanted to unsupervised learning unhesitatingly. Additionally, for each function family $\mathfrak{F}$, it is only determined by the structure of the deep neural network. Therefore, the proposed method is suitable for training deep neural networks with different structures because of the universality of $\mathfrak{F}$.

5. Reducing overfitting

Similar to supervised learning methods, overfitting also exists in the proposed method. For an arbitrary constant vector **c**, there is a function $f(\mathbf{x}) = \mathbf{c}$ that can hit the minimum of Eq. (2). However, it is noneffective for pre-training DNNs or learning valuable feature representations. Inspired by Ioffe and Szegedy [48], we combat the overfitting by normalizing the inputs of each layer. Experiments turn out that the batch normalization operation can alleviate the overfitting and improve the performance in practice.

6. Experiments

To compare the performance of CILR and its variants with previous deep unsupervised learning methods, we report the classification results with various deep models on classification (pre-training deep models) and clustering tasks in Section 6.3 and

Table 2
Classification accuracies (%) on MNIST with different deep neural network networks and various number of labeled samples.

Methods	300	600	1000	3000	5000	10,000	60,000 (full)
Fully connected neural network							
Baseline (without pre-training)	82.55	90.33	93.64	95.23	96.03	97.17	98.03
DBN [23]	82.56	89.76	93.76	94.90	96.58	97.61	98.17
RBM [58]	85.22	90.75	94.87	96.01	97.14	97.91	98.32
AE [25]	84.52	90.45	94.48	95.55	97.15	97.81	98.23
SAE [39]	88.38	91.90	94.85	96.13	97.24	97.63	97.95
DAE [40]	88.06	92.34	94.90	96.03	97.41	97.79	98.12
SWWAE without dropout [42]	93.83	95.84	96.61	97.50	98.14	98.45	98.74
SWWAE with dropout [42]	94.29	96.69	97.17	97.90	98.35	98.73	99.01
Siamese [59]	85.35	91.64	95.64	95.74	97.02	97.31	98.30
CILR	**95.45**	96.15	96.95	97.67	98.05	98.46	98.93
Sparse CILR	92.41	94.52	95.39	96.71	97.45	98.03	98.54
Regularized CILR	95.21	**96.71**	**97.75**	**98.07**	**98.46**	**98.86**	**90.04**
Convolutional neural network							
Baseline (without pre-training)	95.49	96.74	97.16	97.95	98.52	98.87	99.09
CAE [25]	95.82	97.01	97.62	98.04	98.52	98.91	99.03
DCAE [40]	95.95	97.14	97.88	98.15	98.93	99.05	99.12
Scat1 [60]	95.30	96.69	97.70	98.23	98.97	99.12	99.25
Scat2 [60]	94.40	95.92	97.40	97.92	98.60	99.00	99.35
CKN_GM1 [61]	95.61	96.54	97.40	98.02	98.59	98.83	99.23
CKN_GM2 [61]	95.76	96.73	97.95	98.15	98.79	99.12	99.35
CKN_PM1 [61]	94.02	95.89	96.77	98.12	98.59	98.82	99.10
CKN_PM2 [61]	95.85	96.49	97.24	98.24	98.44	98.90	99.04
Kmeans+CNN [62]	97.03	97.20	97.50	98.60	99.01	99.13	99.39
Siamese [59]	95.63	97.34	97.64	97.93	98.32	98.56	98.95
CILR	97.30	97.94	98.24	98.86	99.06	99.24	99.43
Sparse CILR	96.04	96.94	97.44	97.86	98.23	98.94	99.09
Regularized CILR	**97.43**	**97.89**	**98.48**	**98.97**	**99.11**	**99.37**	**99.56**

Section 6.4, respectively. Furthermore, some additional experiments are performed in Section 6.5 to assess the invariance of the learned representations, the convergence of CILR, and so on. Specifically, our core code is released at https://github.com/vector-1127/CILR.

6.1. Datasets

We evaluate the proposed approaches on six popular image datasets:

MNIST [49]: A dataset of 28×28 gray handwritten digit images in 10 classes, has a training set of 60000 examples, and a test set of 10000 examples.

SVHN [50]: SVHN is a real-world digit image dataset, which consists of 32×32 color images in 10 classes. It contains 73257 digit samples for training, 26032 digit samples for testing.

CIFAR-10 [51]: A dataset of 32×32 colour images in 10 classes, with 5000 examples each. There are 50000 training images and 10000 test images.

CIFAR-100 [51]: This dataset is just like the CIFAR-10. Except it has 100 classes, and only 500 training images, 100 testing images per class.

STL-10 [52]: The STL-10 dataset consists of 96×96 color images and relatively less labeled data. There are 5000 training samples, 8000 test samples and 100000 unlabeled samples.

ILSVRC2012 1K [53]: ILSVRC2012 1K image dataset consists of 1000 categories. The dataset has 1.28 million training and 50k validation images, respectively.

6.2. Experimental settings

For each experiment, the Euclidean distance is employed to measure the similarity between different feature representations in the proposed method during pre-training procedure. We follow

Table 3
The structures of convolutional Neural Networks for SVHN/CIFAR-10/CIFAR-100 classification.

Block n	Input $3 \times 32 \times 32$ RGB image
	Block 96
3×3 conv. n BN ReLU	Block 192
3×3 conv. n BN ReLU	3×3 conv. 192 BN ReLU
3×3 conv. n BN ReLU	1×1 conv. 192 BN ReLU
2×2 max-pooling strider 2	1×1 conv. 10 BN ReLU
	global averaging BN
–	10 or 100-way softmax

the previous works [1,39] for setting hyper-parameters. Specifically, $\lambda = 0.0005$, $\beta = 0.1$ and $\rho = 0.01$ are utilized in the regularized CILR and the sparse CILR. Then, the binary cross entropy loss function with weight decay is utilized to fine-tune the pre-trained deep neural networks with an identical labeled data set. And the weight decay parameter is 0.0005. Furthermore, $\mathcal{A}$ is a composition of elementary transformations the following transformations:

- noise: apply to the input a zero-centered Gaussian noise with standard deviation;
- rotation: randomly rotate the image in $[-5, 5]$;
- scaling: multiply the image by a value in [0.9, 1.1];
- channel shift: randomly shift the channel in [0.9, 1.1];
- zoom range: randomly zoom the image in [0.85, 1.15];
- horizontal flip: randomly flip the image horizontally;
- width shift: randomly horizontally shift the image by a pixel-wise distance within 0.18 of image size;
- height shift: randomly vertically shift the image by a pixel-wise distance within 0.18 of image size.

Since the $\mathcal{A}$ is data driven, the horizontal flip is omitted in the MNIST and SVHN classification tasks. Furthermore, to get a typical performance indicator of different methods, we repeat the fine-tuning process for 10 times with different samples which are ran-

Table 4
Classification accuracies (%) on CIFAR-10 with different convolutional neural networks and various number of labeled samples.

Methods	300	1000	4000	5000	10,000	50,000 (full)
Baseline (without pre-training)	38.23	47.70	57.43	61.82	65.65	78.20
S3C [65]	42.12	54.82	69.10	71.07	74.69	78.30
SCAE [29]	39.32	52.14	68.94	74.45	78.35	90.42
NOMP-20 [66]	45.64	56.19	72.20	73.14	77.93	82.90
Learned RF [67]	40.35	53.24	70.70	72.23	78.03	82.00
Zero-bias CAE [68]	39.86	50.56	61.70	68.42	74.06	83.64
ALL-CNN [64]	40.56	54.67	70.87	72.35	79.89	90.69
SWWAE [42]	45.34	59.67	73.24	77.98	81.56	92.23
Siamese [59]	35.87	46.48	59.98	67.32	71.74	79.12
CILR	49.73	60.89	74.29	75.68	81.99	90.92
Sparse CILR	47.73	59.89	74.23	75.33	81.47	90.23
Regularized CILR	**50.45**	**61.75**	**76.74**	**78.37**	**83.59**	**92.31**

domly sampled from the original labeled dataset. The average of 10 results can be regarded as a reasonable evaluation of methods.

The deep learning library Keras [54] is employed to actualize deep models. The RMSProp optimizer [55] is utilized to optimize objective functions in this paper. The learning rate is 0.001 for the initial phase of learning. Batch size is 256 in the pre-training procedure. In the fine-tuning phase, batch size is 100. For each deep neural network, the Gaussian initialization strategy described in [56] is used to initialize the parameters. The ReLU activation function [57] is employed to improve the efficiency of training, as analysed in [1]. And the batch normalization described in [48] is utilized to normalize the inputs of each layer in the modeled deep neural networks.

6.3. *Classification results*

In this section, we compare our approach with existing deep unsupervised methods on the pre-training task.

6.3.1. *MNIST*

We optionally devise a fully connected neural network and a convolutional neural network to compare the performance of the proposed approaches with some traditional pre-training methods on MNIST. The structure of the molded network is illustrated in Table 1. The convolutional neural network is an extension of the LeNet-5 [49], but it has more filters in convolutional layers. In the pre-training stage, the whole images in MNIST are utilized to pre-training the deep networks. In the fine-tuning stage, we randomly select labeled samples to fine-tune the networks. The sizes of labeled subset are respectively 300, 600, 1000, 3000, 5000, 10,000 and 60,000 (full). Additionally, in this paper, the autoencoder and its variants are employed to compare with the proposed method. For more comprehensive compare, we also collect the previously reports to demonstrate the effectiveness of the proposed method and its variants.

The results of different pre-training methods are illustrated in Table 2 and verify that our approach outperforms the traditional pre-training methods. We also observe that the autoencoder can weakly improve the performance of the neural networks and a proper regularization term is necessary for a better result. In summary, these results empirically imply that our approach is valuable for pre-training both fully connected neural networks and convolutional neural networks.

6.3.2. *CIFAR-10 & CIFAR-100 & SVHN*

Since the samples in CIFAR-10, CIFAR-100 and SVHN have the same size, the similar deep neural network illustrated in Table 3 are devised in this subsection. The structure is inspired by Conv-Large network described in [63,64]. Contrary to the Conv-Large, the ReLU activation function is utilized and the Gaussian noise is added in the input of each layer. For data preparation, we follow the previous work as described in [42]. For CIFAR-10 and CIFAR-100, we resize the unlabeled images in STL-10 from 96×96 pixels to 32×32 pixels and utilize they to pre-train the networks. Specifically, the sizes of labeled subset used in CIFAR-10 are respectively 300, 1000, 4000, 5000, 10000 and 50000 (full). And the whole training data on CIFAR-100 are utilized in fine-tuning phase as the limited samples obtained in per class. When testing on SVHN, the whole images are employed to pre-training the deep networks first. Then, we utilize the whole labeled samples to fine-tune the networks. Because the traditional pre-training methods cannot pre-train the devised convolutional neural network, we compare the validity of CILR and its variants with recently proposed methods only.

The results of CIFAR-10, CIFAR-100 and SVHN are illustrated in Tables 4–6, respectively. The results indicate that the proposed method can improve the capability of the networks, especially when labeled data is very limited. In addition, these results also demonstrate that our method is practicable on various deep models. Contrary to the traditional pre-training methods, this property can enhance the universality of our approach.

Table 5
Classification accuracy (%) on CIFAR-100 with the whole labeled data.

Methods	accuracy
Baseline (without pre-training)	65.12
All-Convnet [64]	66.29
Highway Network [69]	67.76
Deeply-supervised nets [70]	65.43
Fractional Max-pooling [71]	68.55
SWWAE [42]	69.12
Siamese [59]	65.97
CILR	68.82
Sparse CILR	67.34
Regularized CILR	**69.39**

Table 6
Classification accuracy (%) on SVHN with the whole labeled data.

Methods	accuracy
Baseline (without pre-training)	94.30
M1+TSVM [72]	95.67
M1+M2 [72]	95.98
SWWAE without dropout [42]	95.17
SWWAE with dropout [42]	96.44
Siamese [59]	93.35
CILR	**97.12**
Sparse CILR	95.34
Regularized CILR	97.09

Table 7
The structures of convolutional Neural Networks for STL-10 classification.

Block n	Input $3 \times 96 \times 96$ RGB image
	Block 64
	3×3 max-pooling strider 3
5×5 conv. n BN ReLU	Block 128
5×5 conv. n BN ReLU	2×2 max-pooling strider 2
5×5 conv. n BN ReLU	$\{3 \times 3$ conv. 256 BN ReLU$\} \times 2$
	Full 2048 BN ReLU
	Full 1024 BN ReLU
-	10-way softmax

Table 8
Classification accuracy (%) on STL-10 with the whole labeled data.

Methods	accuracy
Baseline (without pre-training)	68.74
Siamese [59]	65.47
Convolutional K-means Networks [52]	60.10
Convolutional Kernel Networks [61]	62.32
Hierarchical Matching Pursuit (HMP) [74]	64.50
Exemplar CNN [75]	72.80
Multi-task Bayesian Optimization [76]	70.10
Zero-bias Convnets + ADCU [68]	70.20
SWWAE [42]	74.33
CILR	**74.63**
Sparse CILR	72.56
Regularized CILR	74.24

Table 9
Classification accuracy (%) on ILSVRC2012 1K [53] with VGG-16 and VGG-19 networks [73].

Networks	VGG-16	VGG-19
Baseline (without pre-training) [73]	74.40	74.50
CILR	**74.54**	74.59
Sparse CILR	73.43	73.72
Regularized CILR	74.49	**74.75**

6.3.3. *STL-10*

STL-10 is particularly primly suited for deep unsupervised learning as it contains a large set of 100,000 unlabeled images. We carry out CILR and its variants with a VGG-style network [73] as illustrated in Table 7. And the unlabeled samples are used to pre-train the deep model. Similar to CIFAR-100, the pre-trained network is fine-tuned with whole labeled samples from STL-10. The classification results of STL-10 are shown in Table 8. It also demonstrates that the proposed method is available for pre-training deep convolutional neural networks.

6.3.4. *ILSVRC2012 1K*

We conduct experiments on the ILSVRC2012 1K dataset [53] to evaluate the performance of the proposed method. Specifically, the VGG-16 and VGG-19 networks proposed in [73] are employed as the based models. From Table 9, we observe that our method can improve the baselines marginally. This indicates that the proposed approaches are useful in practical applications.

Furthermore, in terms of the pre-training task, several tendencies can be observed from the results on the aforementioned six datasets. First, the regularized CILR can effectively improve the performance of CILR. This indicates that more generalization capability can be obtained by applying the weight decay during pre-training phase. Secondly, we found that the sparse CILR may degrade the performance of CILR. One possible reason is that the sparsity constraint weakens the original objective which attempts to map similar data points to one point. This observation in turn indicates that the original objective is important to pre-training deep neural networks. Thirdly, although the siamese networks [59] consider pairwise samples as CILR to pre-train deep models, the contribution of the siamese networks is limited. Further analysis, the siamese networks treat the whole seed data points as dissimilar samples. This may introduce a mass of noisy samples during the pre-training stage and degrade the performance of the pre-training.

6.4. *Clustering results*

In this section, we evaluate the availability of the learned latent representations by performing the clustering task on MNIST, CIFAR-10 and STL-10. For each dataset, specifically, the networks modeled in Section 6.3 are utilized in the clustering task directly. To a roundly comparison, three popular measures in the literature are employed to evaluate the performance of the aforementioned clustering methods, including Accuracy (ACC) [77], Normalized Mutual Information (NMI)[78] and Adjusted Rand Index (ARI) [79]. Specifically, these measures range in [0, 1], and higher scores imply more accurate clustering results. In experiments, the K-means [80] method is employed as the postprocessing to cluster images.

In Table 10, we report the quantitative clustering results of these clustering methods. Note that CILR and its varieties dramatically outperform the others methods with significant margins on all the three clustering quality measures. The objective of CILR, intrinsically, is mapping similar data samples to the same points in the learned latent space. This scenario can be treated to achieve the same as that of the clustering task. Compared with CILR, the feature representations learned by these traditional methods may be ineffective for clustering, since they are trained by reconstructing inputs only.

6.5. *Detailed analyses*

In this section, we systematically investigate the invariance of the features learned by these unsupervised methods and the reliability of our approach when networks go deeper. The contribution of batch normalization and the convergence of our approach are also empirically demonstrated in the following.

6.5.1. *The invariance of the learned representation*

We quantificationally analyse the invariance of learned feature representations by some unsupervised methods, and the MNIST dataset is used to demonstrate it. The different unsupervised learning methods are utilized to pre-train the deep neural networks first. Then, we randomly sample 1000 images from the original labeled dataset to fine-tune the pre-trained models. In testing phase, we apply the original testing data and transformed testing data to evaluate the invariance of the learned feature representations. The transformations consist of rotation, height shift, width shift, and zoom range. Specifically, the classification accuracy is utilized to measure the invariance of the different feature representations.

We choose the somewhat arbitrary networks to compare the invariance of the learned representations. As shown in Table 11, the traditional fully connected network and convolutional neural network are modeled. Generally, the features learned by autoencoder and its variants are employed to compare with the features learned by our approach. The results of fully connected network and convolutional neural network are intuitively illustrated in Figs. 2 and 3, respectively. They show that the improvement yielded by our approach is significant when images occur severe transformations. That is, more invariant feature representations are learned by the proposed method.

The main reason is the information utilized in these traditional methods only comes from a single sample, but CILR mines the valuable information from a group of samples. Further analyse, the feature representations obtained by these traditional methods can

Table 10
The clustering results based on the learned label features by different methods.

Dataset	MNIST [49]			CIFAR-10 [51]			STL-10 [52]		
Metric	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means [80]	0.5129	0.4908	0.3607	0.2061	0.0789	0.0417	0.2205	0.1245	0.0608
AE [25]	0.5691	0.5088	0.3953	0.1989	0.0720	0.0347	0.2295	0.0960	0.0487
SAE [39]	0.6045	0.4964	0.3995	0.2160	0.0815	0.0436	0.2244	0.1071	0.0604
DAE [40]	0.5807	0.5359	0.4152	0.2042	0.0765	0.0380	0.2276	0.1128	0.0628
CILR	**0.8855**	**0.7986**	**0.7756**	**0.2583**	**0.1105**	**0.0612**	**0.2452**	**0.1398**	**0.0652**
Sparse CILR	0.8245	0.7307	0.6660	0.2252	0.1014	0.0561	0.2402	0.1103	0.0524
Regularized CILR	0.8844	0.7953	0.7696	0.2404	0.0966	0.0523	0.2318	0.1121	0.0573

Table 11
Models for testing invariance of learned features.

Input 28×28 monochrome image	
Full 1024 BN ReLU	3×3 conv. 32 BN ReLU
Full 512 BN ReLU	2×2 max-pooling strider 2 BN
Full 512 BN ReLU	3×3 conv. 64 BN ReLU
Full 256 BN ReLU	2×2 max-pooling strider 2 BN
Full 256 BN ReLU	3×3 conv. 128 BN ReLU
Full 128 BN ReLU	2×2 max-pooling strider 2 BN
Full 128 BN ReLU	Full 512 BN ReLU
10-way softmax	

Table 12
Models for going deeper.

Input 28×28 monochrome image	Input $3 \times 32 \times 32$ RGB image
{Full 784 BN ReLU} $\times n$	{Full 3072 BN ReLU} $\times n$
10-way softmax	

perfectly reconstruct the input data. However, it is dubious that the learned features benefit to supervised tasks (such as classification). For example, principle component analysis can minimize the mean squared error of training data, but the performance for image classification is worse than the autoencoder with larger mean squared error [37]. Compared with these traditional methods, CILR and its variants utilize much latent labeled information and stronger constraint which the feature representations of similarly samples are consistent. The results verify that this regulation is valid and practicable.

Additionally, we find that these transformations have a symmetrical influence on the performance of deep neural networks. Specially, it can be seen that the translation (width shift and height shift) can significantly degenerate the performance of deep neural networks pre-trained by the traditional unsupervised pre-training methods. One possible reason is that the labeled training data is insufficient to train an excellent deep neural network. However, as shown in Figs. 2 and 3, robust feature representations can be learned if a more appropriate guidance is given.

We perform additional experiments to study the contributions of different transformations, including rotation, height shift, width shift and zoom. In the experiments, each transformation is utilized separately to pre-train the CNN described in Table 11 based on CILR. Similar as the settings described in this subsection, we randomly sample 1000 images from MNIST to fine-tune the pre-trained CNN and evaluate the invariance on the MNIST testing images. From Fig. 4, it can be seen that the invariance of the learned feature representations origin from these transformations. For example, the learned feature representations will acquire the translation invariance if the translation transformations (*e.g.*, height and width shift) are utilized in the pre-training stage. Furthermore, as illustrated in the top left corner of Fig. 4, the rotation invariance may be hard to learn. The estimated reason is that the rotating transformed data points possess different distribution between the original data points. These generated data points drive the proposed method to learn an invalid model and degenerate the performance.

6.5.2. *Going deeper*

In this subsection, we employ two deep models, *i.e.*, highway [69] and residual networks [4], to evaluate the performance of our method on very deep architectures.

Highway network

The fully connected network ("plain network" described in [69]) is employed to model deeper networks to investigate the capacities of different pre-training methods. The details of network are illustrated in Table 12, where n represents the number of fully connected layers added in the networks and these two models are used for MNIST and CIFAR-10 classification, respectively. For data preparation, we follow the aforementioned settings in Section 6.3. The sizes of labeled subset are respectively 300, 600, 1000, 3000, 5000, 10,000, 60,000 (full) on MNIST. For CIFAR-10, the number of fine-tuned images are respectively 300, 1000, 5000, 10,000, 30,000, 50,000 (full).

The classification results of MNIST and CIFAR-10 are illustrated in Figs. 5 and 6, respectively. The overall experiments empirically confirm that the proposed method is stabilized with the network depth. Namely, with the increase of network depth, CILR and its variants still show superior performance than other unsupervised methods. Furthermore, the pre-trained networks also keep a high-accuracy with expanding the number of labeled data. In total, all these observations verify that our approach is practicable to pre-train deeper neural networks. Further analyse, the degradation is occurred with the networks depth increasing as analysed in [4]. Compared with those traditional pre-training methods, the end-to-end pre-training way can avoid learning the superfluous auxiliary parameters and layers. Therefore, the better pre-training capability is acquired. Additionally, we observe that the capabilities of deeper neural networks will be severely degenerated when only limited labeled data is utilized. It is reasonable that the more labeled data is indispensable to train an excellent deep neural network with more parameters.

Residual network

We also perform experiment to study the performance of CILR on the residual networks [4]. Following the protocols in [4], the resnet-32 network is used in the experiments. For the CIFAR-10 and CIFAR-100 datasets, the whole images are utilized to pre-train and fine-tune resnet-32 orderly. As for STL-10, the unlabeled and the labeled images are employed to pre-train and fine-tune resnet-32, respectively.

Several tendencies can be founded from Table 13. First, our approach marginally elevates the performance of resnet-32. This indicates that CILR is a general and useful approach for pre-training various deep models. Second, compared with the CIFAR-10 and

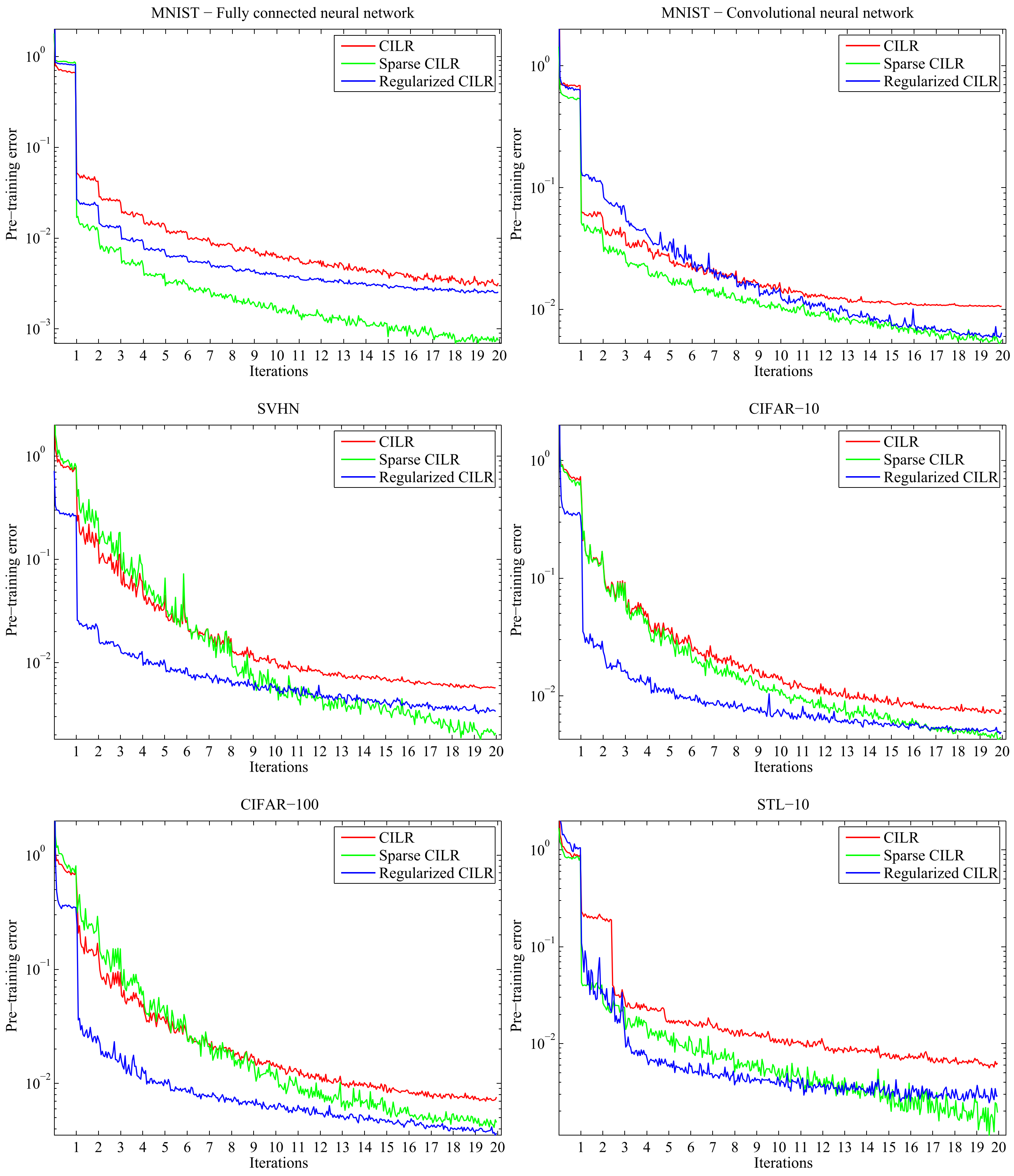

Fig. 8. Pre-training error of CILR and its variants in the first 20 iterations on different networks and datasets. The supervised learning procedure is executed between two continuous iterations.

CIFAR-100 datasets, a larger margin is yielded on STL-10. The main reason is that less labeled images are included in STL-10 than those in CIFAR-10 and CIFAR-100. As the results illustrated in the aforementioned sections, such scenario can highlight the advantages of our method.

6.5.3. *Contribution of batch normalization*

In order to evaluate the contribution of batch normalization in our approach, comparative experiments is performed on MNIST with the pre-defined networks in Section 6.3.1. More details are illustrated in Table 1.

Table 13
Classification accuracy (%) on several datasets with resnet-32 [4].

Datasets	CIFAR-10 [51]	CIFAR-100 [51]	STL-10 [52]
Baseline (without pre-training) [73]	87.50	60.44	75.42
CILR	87.84	61.76	**76.72**
Sparse CILR	85.26	61.12	74.56
Regularized CILR	**88.14**	**61.93**	76.53

Table 14
Classification accuracies (%) on MNIST with/without batch normalization.

Methods	300	600	1000	3000	5000	10,000	60,000 (full)
Fully connected neural network							
Baseline (without pre-training)	82.55	90.33	93.64	95.23	96.03	97.17	98.03
CILR without BN	85.47	93.33	95.64	96.14	96.36	97.98	98.74
CILR with BN	**95.45**	**96.15**	**96.95**	**97.67**	**98.05**	**98.46**	**98.93**
Convolutional neural network							
Baseline (without pre-training)	95.49	96.74	97.16	97.95	98.52	98.87	99.09
CILR without BN	95.74	96.24	96.23	96.39	97.74	98.37	98.78
CILR with BN	**97.30**	**97.94**	**98.24**	**98.86**	**99.06**	**99.24**	**99.43**

The results are shown in Fig. 7 and Table 14. From Fig. 7, it can be seen that batch normalization can restrain CILR to minimize the objective of our model. Although the large pre-training error is yielded, the results listed in Table 14 indicate that batch normalization can improve the performance of the pre-training. This is because batch normalization can normalize the activation of each layer to avoid generating constant representations in the our model.

6.5.4. *Convergence*

In this section, we quantitatively analyse the convergence of CILR and its variants. The networks illustrated in Section 6.3 are utilized to basic networks. For the MNIST and SVHN datasets, the whole labeled samples are used to pre-train the networks. The unlabeled images in STL-10 are employed to pre-train the networks modeled for the CIFAR-10, CIFAR-100, and STL-10 datasets. More intuitively, the pre-training error of CILR and its variants on the different networks and the datasets are illustrated in Fig. 8. It is convictive that our approach and its variants are convergent.We also observe that the weight decay and the sparse penalty term applied in CILR can reduce the pre-training error. Compared with the regularized CILR, furthermore, the small pre-training error is obtained by the sparse CILR, but high performance is yielded by the regularized CILR. One explanation could be that the sparse penalty term weakens the learning of the original objective function, but the weight decay can assist the learning.

7. Conclusion

We have proposed a novel end-to-end unsupervised method termed CILR to pre-train deep neural networks. The core idea is learning a function which maps a set of similar data points to an identical point. This treatment can be regarded as a natural extension of supervised learning to unsupervised learning. By creating a group of latent labeled data sets, the deep neural networks can be trained excellently by the proposed method in the unsupervised learning way. We confirm our approach both theoretically and experimentally, indicating that it is superior to the traditional pre-training methods and has enough capability to train a variety of deep neural networks. Furthermore, we improve CILR by applying some regularization terms to objective function, including l_2-norm and sparse penalties. It is an interesting direction that adding more reasonable penalties on layer parameters or activities during optimization to improve the method we proposed. We will see this for future work.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (NSFC Nos. 91646207, 61403376, 61370039 and 91338202), the Beijing Nature Science Foundation under Grant No. 4162064, and the Youth Innovation Promotion Association CAS.

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1106–1114.
[2] Y. Li, W. Xie, H. Li, Hyperspectral image reconstruction by deep convolutional neural network for classification, Pattern Recognit. 63 (2017) 371–383.
[3] C. Barat, C. Ducottet, String representations and distances in deep convolutional neural networks for image classification, Pattern Recognit. 54 (C) (2016) 104–115.
[4] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015.
[5] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Semantic image segmentation with deep convolutional nets and fully connected CRFs, Comput. Sci. (4) (2014) 357–361.
[6] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer, 2015, pp. 234–241.
[7] Z. Liu, X. Li, P. Luo, C.-C. Loy, X. Tang, Semantic image segmentation via deep parsing network, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1377–1385.
[8] B. Su, S. Lu, Accurate recognition of words in scenes without character segmentation using recurrent neural network, Pattern Recognit. 63 (2017) 397–405.
[9] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[10] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[11] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems, 2015, pp. 91–99.
[12] E. Ohn-Bar, M.M. Trivedi, Multi-scale volumes for deep object detection and localization, Pattern Recognit. 61 (2016) 557–572.
[13] Y. Sun, D. Liang, X. Wang, X. Tang, Deepid3: face recognition with very deep neural networks, arXiv preprintarXiv:1502.00873 (2015).
[14] O.M. Parkhi, A. Vedaldi, A. Zisserman, Deep face recognition, in: British Machine Vision Conference, 1, 2015, p. 6.
[15] S. Elaiwat, M. Bennamoun, F. Boussaid, A spatio-temporal RBM-based model for facial expression recognition, Pattern Recognit. 49 (C) (2016) 152–161.
[16] A.T. Lopes, E.D. Aguiar, A.F.D. Souza, T. Oliveira-Santos, Facial expression recognition with convolutional neural networks: coping with few data and the training sample order, Pattern Recognit. 61 (2016) 610–628.
[17] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.

[18] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEEInternational Conference on, IEEE, 2013, pp. 6645–6649.
[19] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al., Deep speech 2: end-to-end speech recognition in English and Mandarin, in: International Conference on Machine Learning, 2016, pp. 173–182.
[20] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al., Deep speech: scaling up end-to-end speech recognition, arXiv preprintarXiv:1412.5567 (2014).
[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[22] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al., Deepid-net: deformable deep convolutional neural networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2403–2412.
[23] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[24] N.N. Ji, J.S. Zhang, C.X. Zhang, A sparse-response deep belief network based on rate distortion theory, Pattern Recognit. 47 (9) (2014) 3179–3191.
[25] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al., Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153.
[26] W.W.Y. Ng, G. Zeng, J. Zhang, D.S. Yeung, W. Pedrycz, Dual autoencoders features for imbalance classification problem, Pattern Recognit. 60 (2016) 875–889.
[27] K.G. Lore, A. Akintayo, S. Sarkar, Llnet: a deep autoencoder approach to natural low-light image enhancement, Pattern Recognit. 61 (2015) 650–662.
[28] R. Salakhutdinov, G. Hinton, Deep Boltzmann machines, J. Mach. Learn. Res. 5 (2) (2009) 448–455.
[29] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: International Conference on Artificial Neural Networks, Springer, 2011, pp. 52–59.
[30] Y. Wang, X. Wang, W. Liu, Unsupervised local deep feature for image recognition, Inf. Sci. 351 (2016) 67–75.
[31] X. Feng, Y. Zhang, J. Glass, Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition, in: Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, IEEE, 2014, pp. 1759–1763.
[32] M. Kan, S. Shan, H. Chang, X. Chen, Stacked progressive auto-encoders (SPAE) for face recognition across poses, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1883–1890.
[33] Z. Zhu, X. Wang, S. Bai, C. Yao, X. Bai, Deep learning representation using autoencoder for 3d shape retrieval, Neurocomputing 204 (2016) 41–50.
[34] W. Wang, B.C. Ooi, X. Yang, D. Zhang, Y. Zhuang, Effective multi-modal retrieval based on stacked auto-encoders, Proc. VLDB Endow. 7 (8) (2014) 649–660.
[35] F. Feng, X. Wang, R. Li, Cross-modal retrieval with correspondence autoencoder, in: Proceedings of the 22nd ACM International Conference on Multimedia, ACM, 2014, pp. 7–16.
[36] S. Bai, X. Bai, Sparse contextual activation for efficient visual re-ranking, IEEE Trans. Image Process. 25 (3) (2016) 1056–1069.
[37] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[38] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 833–840.
[39] A. Ng, Sparse autoencoder, CS294A Lecture notes 72(2011) 1–19.
[40] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (6) (2010) 3371–3408.
[41] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: Computer Vision and Pattern Recognition, 2010 IEEE Conference on, IEEE, 2010, pp. 2528–2535.
[42] J. Zhao, M. Mathieu, R. Goroshin, Y. Lecun, Stacked what-where auto-encoders, 2015.
[43] M. Ranzato, F.J. Huang, Y. Boureau, Y. Lecun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: Computer Vision and Pattern Recognition, 2007, pp. 1–8.
[44] H. Valpola, From neural PCA to deep unsupervised learning, Adv. Indep. Compon. Anal. Learn. Mach. (2015) 143–171.
[45] M. Pezeshki, L. Fan, P. Brakel, A. Courville, Y. Bengio, Deconstructing the ladder network architecture, 2015.
[46] N. Lange, Neural Networks for Pattern Recognition, in: C.M. Bishop (Ed.), MIT Press, 1993. Pattern Recognition and Neural Networks. by B. D. Ripley.
[47] T.K. Moon, The expectation-maximization algorithm, IEEE Signal Process. Mag. 13 (6) (1996) 47–60.
[48] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, 2015.
[49] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[50] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, 2010.
[51] A. Krizhevsky, Learning multiple layers of features from tiny images, 2012.
[52] A. Coates, A.Y. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, J. Mach. Learn. Res. 15 (2011) 215–223.
[53] J. Deng, W. Dong, R. Socher, L. Li, K. Li, F. Li, Imagenet: a large-scale hierarchical image database, in: CVPR, 2009, pp. 248–255.
[54] F. Chollet, Keras, 2015, (https://github.com/fchollet/keras).
[55] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: divide the gradient by a running average of its recent magnitude, 2012, (COURSERA: Neural Networks for Machine Learning).
[56] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, J. Mach. Learn. Res. 9 (2010) 249–256.
[57] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines Vinod Nair, in: International Conference on Machine Learning, 2010, pp. 807–814.
[58] H. Larochelle, Y. Bengio, Classification using discriminative restricted Boltzmann machines, in: International Conference on Machine Learning ACM, 2008, pp. 536–543.
[59] R. Hadsell, S. Chopra, Y. LeCun, Dimensionality reduction by learning an invariant mapping, in: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, 2, IEEE, 2006, pp. 1735–1742.
[60] J. Bruna, S. Mallat, Invariant scattering convolution networks, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1872–1886.
[61] J. Mairal, P. Koniusz, Z. Harchaoui, C. Schmid, Convolutional kernel networks, Adv. Neural Inf. Process. Syst. (2014) 2627–2635.
[62] M. Simon, E. Rodner, Convolutional clustering for unsupervised learning, Comput. Sci. (2015) 1143–1151.
[63] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, T. Raiko, Semi-supervised learning with ladder networks, 2015.
[64] J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net, 2014.
[65] I. Goodfellow, A. Courville, Y. Bengio, Large-scale feature learning with spike-and-slab sparse coding, in: International Conference on Machine Learning, 2012, pp. 1439–1446.
[66] T.H. Lin, H.T. Kung, Stable and efficient representation learning with nonnegativity constraints, 2014, pp. 1323–1331.
[67] A. Coates, A.Y. Ng, Selecting receptive fields in deep networks, Adv. Neural Inf. Process. Syst. (2012) 2528–2536.
[68] T.L. Paine, P. Khorrami, W. Han, T.S. Huang, An analysis of unsupervised pre–training in light of recent advances, 2014.
[69] R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, 2015.
[70] ScienceOpen, D.S. Nets, Deeply-supervised nets, 2014.
[71] B. Graham, Fractional max-pooling, 2014.
[72] D.P. Kingma, D.J. Rezende, S. Mohamed, M. Welling, Semi-supervised learning with deep generative models, Adv. Neural Inf. Process. Syst. 4 (2014) 3581–3589.
[73] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014.
[74] L. Bo, X. Ren, D. Fox, Unsupervised Feature Learning for RGB-D Based Object Recognition, Springer International Publishing, 2013.
[75] A. Dosovitskiy, P. Fischer, J.T. Springenberg, M. Riedmiller, T. Brox, Discriminative unsupervised feature learning with exemplar convolutional neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 1 (10) (2015). 1–1.
[76] K. Swersky, J. Snoek, R.P. Adams, Multi-taskBayesian optimization, Adv. Neural Inf. Process. Syst. (2013) 2004–2012.
[77] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (Dec) (2002) 583–617.
[78] T. Li, C. Ding, The relationships among various nonnegative matrix factorization methods for clustering, in: Data Mining, 2006. ICDM'06. Sixth International Conference on, IEEE, 2006, pp. 362–371.
[79] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[80] J. Wang, J. Wang, J. Song, X. Xu, H. Shen, S. Li, Optimized cartesian k-means, IEEE Trans. Knowl. Data Eng. 27 (1) (2015) 180–192.

Jianlong Chang received the B.S. degree in School of Mathematical Sciences from University of Electronic Science and Technology of China, Chengdu, China, in 2015. He is currently pursuing the Ph.D. degree at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include deep feature learning, pattern recognition and machine learning.

Lingfeng Wang received his B.S. degree in computer science from Wuhan University, Wuhan, China, in 2007. He received his Ph.D. degree from Institute of Automation, Chinese Academy of Sciences in 2013. He is currently an associate professor with the National Laboratory of Pattern Recognition of Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision and image processing.

Gaofeng Meng received the B.S. degree in Applied Mathematics from Northwestern Polytechnical University in 2002, and the M.S. degree in Applied Mathematics from Tianjing University in 2005, and the Ph.D. degree in Control Science and Engineering from Xi'an Jiaotong University in 2009. He is currently an associate professor of the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. He also serves as an Associate Editor of Neurocomputing. His current research interests include document image processing, computer vision and pattern recognition.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1993, the M.S. degree from Chongqing University, Chongqing, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2004. From 1996 to 2001, he was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctorate Candidate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently a Professor with the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include pattern recognition, machine learning, and computer vision.

Chunhong Pan received his B.S. Degree in automatic control from Tsinghua University, Beijing, China, in 1987, his M.S. Degree from Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, China, in 1990, and his Ph.D. degree in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences, Beijing, in 2000. He is currently a professor with the National Laboratory of Pattern Recognition of Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, image processing, computer graphics, and remote sensing.