# Stacked Similarity-Aware Autoencoders

**Wenqing Chu, Deng Cai***
State Key Lab of CAD&CG, College of Computer Science, Zhejiang University, China
wqchu16@gmail.com, dengcai@cad.zju.edu.cn

## Abstract

As one of the most popular unsupervised learning approaches, the autoencoder aims at transforming the inputs to the outputs with the least discrepancy. The conventional autoencoder and most of its variants only consider the one-to-one reconstruction, which ignores the intrinsic structure of the data and may lead to overfitting. In order to preserve the latent geometric information in the data, we propose the stacked similarity-aware autoencoders. To train each single autoencoder, we first obtain the pseudo class label of each sample by clustering the input features. Then the hidden codes of those samples sharing the same category label will be required to satisfy an additional similarity constraint. Specifically, the similarity constraint is implemented based on an extension of the recently proposed center loss. With this joint supervision of the autoencoder reconstruction error and the center loss, the learned feature representations not only can reconstruct the original data, but also preserve the geometric structure of the data. Furthermore, a stacked framework is introduced to boost the representation capacity. The experimental results on several benchmark datasets show the remarkable performance improvement of the proposed algorithm compared with other autoencoder based approaches.

## 1 Introduction

In the recent years, deep neural networks (DNN) have been developed and used successfully to learn rich and powerful visual representations [Krizhevsky *et al.*, 2012]. However, applying this approach usually brings about a huge burden as it demands millions of labeled examples. A natural way to address this problem would be to combine the powerful DNN model and the idea of unsupervised learning so that we can extract useful features without any annotations. Then with limited label information, the learned representations will be more suitable as input to a supervised machine than the raw input. Among massive unsupervised learning models, the autoencoder could be stacked to build a deep structure easily and then utilized to generate useful features naturally [Hinton and Salakhutdinov, 2006]. Not surprisingly, autoencoder based models have been shown to achieve state-of-the-art performance in a number of challenging problems ranging from computer vision [Wang *et al.*, 2016] and audio processing [Plchot *et al.*, 2016] to natural language processing [Wang *et al.*, 2015].

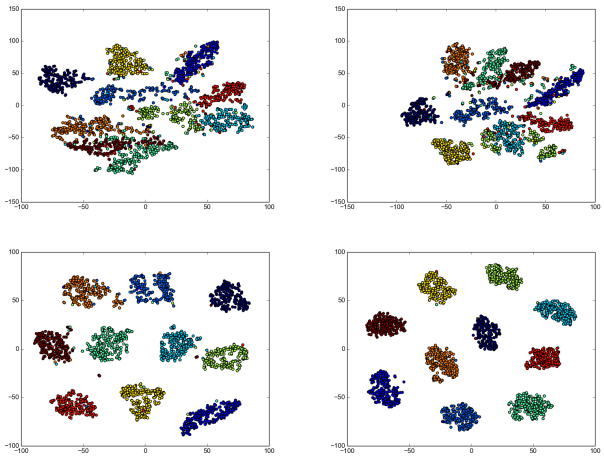

Figure 1: Visualization of different layer features of the MNIST dataset by t-SNE. From left top to right bottom, the corresponding figures are for original data, low level features, middle level features and high level features. Different colors represent different categories.

The conventional autoencoder is based on an encoder-decoder paradigm. Specifically, the encoder first transforms the input into a latent representation, and then the decoder is trained to reconstruct the initial input from this representation as far as possible. This architecture forces the latent representation to capture the most important information of the training data. Unfortunately, if the encoder and decoder are allowed too much capacity, the autoencoder may just learn to perform the identity function without extracting useful representations [Goodfellow *et al.*, 2016]. Thus, there are many studies presenting advanced regularization

*corresponding author

techniques which have led to a series of variants including denoising [Vincent *et al.*, 2010], contractive [Rifai *et al.*, 2011b], k-sparse [Makhzani and Frey, 2014] and winner-take-all [Makhzani and Frey, 2015] autoencoders.

However, the conventional autoencoder and most of its variants only consider the one-to-one reconstruction, which ignores the geometric structure of the data and may lead to overfitting. Generally speaking, the samples of the collected dataset can be grouped into several categories with the similar semantic. This intrinsic characteristic requires that their ideal discriminative representations should form several clusters in the high-dimensional feature space. For example, the MNIST database of handwritten digits has ten categories. Here we train a Convolutional Neural Networks (CNN) model [Lecun *et al.*, 1998] on the MNIST dataset and use t-SNE [Maaten and Hinton, 2008] to visualize different layer features of the samples. From Figure 1, it can be observed that for the original data space, the samples spread throughout the whole space. Interestingly, the samples mapped to the feature space from low-level to high-level, gather together and come into several groups gradually. It is obvious that these high-level representations are very useful for tasks like classification or retrieval. This observation inspires us to introduce a novel regularization technique to guide the unsupervised feature learning.

In this work, we develop a novel autoencoder based framework called Stacked Similarity-Aware Autoencoders (SSA-AE). In order to capture the geometric structure in the data, we exploit similarity constraint as a generic prior on the hidden codes of the autoencoder. Specifically, we first obtain the pseudo class label of each sample by clustering the input features. Then for those samples belonging to the same category, their hidden codes of the autoencoder will be constrained to be close to each other in the high-dimensional space. Here we employ an extension of the center loss [Wen *et al.*, 2016] to implement this similarity constraint. As a result, the learned feature representations not only can reconstruct the original data, but also preserve the geometric structure in the inputs. Even though the clustering results maybe not perfect. For example, two samples have a different real classes but same pseudo class or the same real class but different pseudo classes. However, we utilize the reliable hierarchical clustering method and only several outliers will not mislead the feature learning. Moreover, a stacked framework is utilized to improve the representation capacity. With respect to the deeper stacked autoencoder, the amount of the classes used for clustering will be set less to learn more compact high-level representations. To the best of our knowledge, such autoencoder based deep learning scheme has not been discussed before. We conduct extensive experiments on several benchmark datasets including MNIST and COIL100. The experimental results demonstrate the superior performance of the proposed algorithm compared with other autoencoder based approaches.

The rest of this paper is organized as follows. In Section 2, we present a brief review of the related works. Then we propose the Stacked Similarity-Aware Autoencoders in Section 3. The experimental results on two real-world datasets are presented in Section 4. Finally, we provide some concluding remarks in Section 5.

## 2 Related Work

In this section, we briefly review the recent work on the basic autoencoder and a series of its variants.

The basic autoencoder [Rumelhart *et al.*, 1988] was introduced as a technique for dimensionality reduction. It is a neural network that takes a vector input $x$, maps it into a hidden representation $z$ using an encoder which typically has this form:

$$z = \mathbf{f}(W_e x + b_e)$$

where $\mathbf{f}$ is a non-linear activation function, $W_e$ the encoding matrix and $b_e$ a vector of bias parameters. The hidden representation $z$, is then mapped back into the space of $x$, using a decoder of this form:

$$\tilde{x} = \mathbf{g}(W_d z + b_d)$$

where $\mathbf{g}$ is also a non-linear activation function, $W_d$ is the decoding matrix and $b_d$ a vector of bias parameters. The goal of the autoencoder is to minimize the reconstruction error, which is represented by a distance between $x$ and $\tilde{x}$. The most common type of distance is the mean squared error:

$$l(x, \tilde{x}) = \|x - \tilde{x}\|_2^2 = \|x - \mathbf{g}(W_d z + b_d)\|_2^2$$

The code $z$ typically has less dimensions than $x$, which forces the autoencoder to learn the most important information of the training data. If the code $z$ has as many components as $x$, then no compression is required, and the model could typically end up learning the identity function.

More recently, autoencoders have played a key role in the deep neural network based approaches [Hinton and Salakhutdinov, 2006] where autoencoders are stacked and trained bottom up in unsupervised fashion, followed by a supervised learning phase to train the top layer and fine-tune the entire architecture. Unfortunately, if the encoder and decoder are allowed too much capacity, the autoencoder can learn to perform the copying task without extracting useful feature representations [Goodfellow *et al.*, 2016].

Rather than limiting the model capacity by keeping the code size small, there are many studies presenting advanced regularization techniques for non-convolutional autoencoders. Denoising autoencoder (DAE) [Vincent *et al.*, 2010] is trained to have denoising ability by incorporating artificially random noise to the input data, and then encouraged the output to be as similar to the original undistorted input as possible. In addition, by imposing sparsity on the hidden units during training, an autoencoder can learn useful structures in the input data even if the model capacity is great enough to learn a trivial identity function. Sparsity may be achieved by additional terms in the loss function during training [Boureau *et al.*, 2008], or by manually zeroing all but the few strongest hidden unit activations (referred to as a k-sparse autoencoder [Makhzani and Frey, 2014]). Furthermore, the authors proposed Winner-Take-All (WTA) autoencoders [Makhzani and Frey, 2015] which use aggressive dropout, where all the elements but the strongest of a convolutional map are zeroed out. Moreover, the proposed method exploits a lifetime sparsity by keeping the k percent largest

activation of that hidden unit across the mini-batch samples and setting the rest of activations of that hidden unit to zero. This strategy forces the convolutional decoder to learn robust features.

Meanwhile, graph regularization was utilized to preserve the local structure of the original data space in [Yu *et al.*, 2013; Zhao *et al.*, 2016]. Furthermore, [Ding *et al.*, 2016] developed a novel algorithm named as Deep Robust Encoder (DRE) with locality preserving low-rank dictionary. The core idea is to jointly optimize deep autoencoder and a clean low-rank dictionary, which can rule out noises and extract robust deep features in a unified framework.

However, most of these autoencoder based approaches ignore the geometric structure within the dataset and may lead to overfitting. Motivated by the recent work [Yang *et al.*, 2016], in which the authors proposed to jointly perform unsupervised learning of deep representations and image clusters, we introduce a novel similarity constraint based regularization technique to guide the unsupervised feature learning.

## 3 The Proposed Approach

In this section, we first introduce the motivation of our proposed algorithm, followed by our detailed model named Similarity-Aware Autoencoder. Then we present the stacked architecture which is allowed more representation capacity.

### 3.1 Similarity-Aware Autoencoder

First, we provide the definition for the conventional autoencoder. For simplicity, we use $x_i$, $z_i$, $\tilde{x}_i$ to represent the $i$-th input vector, hidden vector and output vector respectively. Then the reconstruction error can be formulated as bellow:

$$\mathcal{L}_a = \sum_{i=1}^{n} \|x_i - \tilde{x_i}\|_2^2$$

From this formulation, we can see that the conventional autoencoder only consider the one-to-one reconstruction error, which ignores the geometric structure of the data and may lead to overfitting. In general, the collected dataset could have several latent concepts and some samples belong to the same concept. This intrinsic characteristic requires that their ideal representations should form several clusters, which can lead to satisfactory performance for tasks like classification or retrieval. In addition, according to our observation from Figure 1, the samples mapped to the high-level feature space, gather together and come into several groups. This observation inspires us to introduce a novel regularization technique to guide the unsupervised feature learning. Specifically, we hope that the ideal autoencoder will learn a mapping from the inputs to a compact Euclidean space, where the similarity relationships of the corresponding input vectors are preserved or even reinforced. With this similarity constraint, the learned feature will be more compact and discriminative. However, the first difficulty is that we don't have the similarity relationships or supervised signals here. Thus we want to use clustering algorithm to get pseudo class labels. For convenient, we employ the popular agglomerative clustering algorithm [Gowda and Krishna, 1978]. The core idea in agglomerative clustering is to merge two clusters at each step until some stopping conditions. More details can be found in [Jain *et al.*, 1999].

After that, one object class is comprised of the input vectors clustered into the same group. Even though the clustering results maybe not perfect, which means two examples have a different real class but same pseudo class or the same real class but different pseudo classes. These two situations might cause problems. However, the hierarchical clustering method is reliable and only several outliers will not mislead the feature learning. Then, how to develop an effective similarity constraint function to improve the discriminative power of the deeply learned feature representations? Intuitively, minimizing the intra-class variations while keeping the features of different classes separable is the key. Inspired by [Wen *et al.*, 2016], we utilize the center loss to supervise the learning. Suppose there are $N$ classes for the samples and we use $y_i$ to denote the pseudo class label for $x_i$. Here is the formulation for the center loss in [Wen *et al.*, 2016].

$$\mathcal{L}_c = \frac{1}{2} \sum_{i=1}^{n} \|z_i - c_{y_i}\|_2^2$$

The $c_{y_i} \in \mathbb{R}^d$ denotes the $y_i$th class center. And the centers are computed by averaging the hidden codes of the corresponding classes. The formulation effectively reduces the intra-class variations.

However, it didn't take the inter-class variations into account. We hope that the samples should be away from those cluster centers which have different pseudo class labels. To this end, we propose the improved center loss function, as formulated in below.

$$\mathcal{L}_s = \frac{1}{2} \sum_{i=1}^{n} \left\{ \|z_i - c_{y_i}\|_2^2 + \beta \sum_{y' \neq y_i} \left[ \gamma - \left\| z_i - c_{y'} \right\|_2^2 \right]_+ \right\}$$

Here the $\gamma$ is a margin to avoid the influence from those remote class centers who are far enough. And the $\beta$ is a balance parameter for minimizing the intra-class variations and maximizing the inter-class variations. The gradients of $\mathcal{L}_s$ with respect to $z_i$ is computed as:

$$\frac{\partial \mathcal{L}_s}{\partial z_i} = (z_i - c_{y_i}) - \beta \sum_{y' \neq y_i} \mathbf{1} \left[ \left\| z_i - c_{y'} \right\|_2^2 \leq \gamma \right] (z_i - c_{y'})$$

where $\mathbf{1} \left[ \left\| z_i - c_{y'} \right\|_2^2 \leq \gamma \right]$ is the indicator function which takes a value of one if its argument is true and zero otherwise.

Ideally, the class center $c_{y_i}$ should be updated as the hidden codes changed. In other words, we need to take the entire training set into account and average the hidden codes of every class in each iteration, which is inefficient even impractical. Therefore, we follow the solution in [Wen *et al.*, 2016] to address this problem. First, instead of updating the centers with respect to the entire training set, we perform the update based on mini-batch in each iteration. Second, to avoid large perturbations caused by few mislabeled samples, we use a scalar $\alpha$ to control the learning rate of the centers. Then the

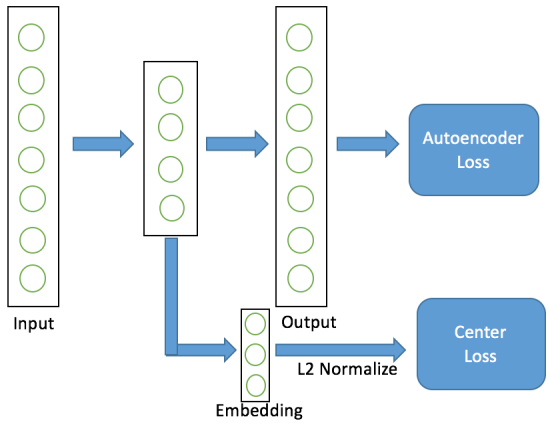


Figure 2: Visualization of the framework of the proposed similarity-aware autoencoder.

update equation of $c_{y_i}$ is computed as:

$$\Delta c_j = \frac{\sum_{i=1}^{n} \{\mathbf{1}[y_i = j](c_j - z_i) + \beta \mathbf{1}[y_i \neq j](z_i - c_j)\}}{1 + \sum_{i=1}^{n} \{\mathbf{1}[y_i = j] + \beta \mathbf{1}[y_i \neq j]\}}$$

Furthermore, the objective function of the similarity-aware autoencoder can be described as below:

$$\mathcal{L} = \mathcal{L}_a + \lambda \mathcal{L}_s \tag{1}$$

where $\lambda$ is a hyper parameter which maintains the balance between these two loss. The framework of the proposed similarity-aware autoencoder is illustrated in Figure 2. For convenience, we append an embedding layer on the top of the hidden codes. And impose the similarity constraint on the embedding layer. As a result, the learned feature representations not only can reconstruct the original data, but also are close to each other for the similar samples.

### 3.2 Stacked Architecture

In this section, we introduce a stacked framework which is utilized to improve the representation capacity. The traditional stacked autoencoders will use a single autoencoder as a building block to form a deep architecture. More precisely, the hidden units of the current autoencoder are worked as the input to feed in the next autoencoder to obtain the stacked representations. According to our observation from Figure 1, the samples mapped to the feature space , gather together and come into several groups gradually from low-level to high-level. This phenomenon gives us an insight that the feature representations from the deeper layer should be more discriminative in term of the semantic. Therefore, the deeper stacked autoencoder deserves stronger similarity constraint to make the hidden codes capture high-level semantic information. And this can be controlled easily by adjusting the amount of classes used for clustering. To this end, for the deeper layers of the proposed stacked autoencoders, the amount of classes used for clustering will be set less to learn more compact high-level representations.

**Algorithm 1** Stacked similarity-aware autoencoders

**Input:** cluster number $N_c > 0$ for agglomerative clustering, shrinking factor $0 < \alpha < 1$, iterative number $R$, stacked autoencoder number $T > 0$, and training examples $\{\mathbf{x}_n\}_{n=1}^N$,
**Output:** Parameters of stacked autoencoder $\{\mathbf{W}_t\}_{t=1}^T$,
Initialize training examples' labels $\{y_n\}_{n=1}^N = 0$
Compute the number of clusters $N_c = \lfloor N \times \alpha \rfloor$
Run agglomerative clustering on $\{\mathbf{x}_n\}_{n=1}^N$
Update training examples' labels $\{y_n\}_{n=1}^N$
**for** $t = 1$ to $T$ **do**
  **for** $r = 1$ to $R$ **do**
    Compute the stochastic gradient according to Eq. (1)
    Update $\{\mathbf{W}_t\}$
  **end for**
  Compute the number of clusters $N_c = \lfloor N_c \times \alpha \rfloor$
  Compute the hidden vector $\{\mathbf{z}_n\}_{n=1}^N$ with current learned autoencoder
  Update $\{\mathbf{x}_n\}_{n=1}^N = \{\mathbf{z}_n\}_{n=1}^N$
  Run agglomerative clustering on $\{\mathbf{x}_n\}_{n=1}^N$
  Update training examples' labels $\{y_n\}_{n=1}^N$
**end for**
**Return:** $\{\mathbf{W}_t\}_{t=1}^T$

The whole procedure of the proposed stacked similarity-aware autoencoders is summarized in Algorithm 1. Before the training of each similarity-aware autoencoder, we need to compute the similarity relationships between all samples. Here, we employ the agglomerative clustering algorithm to obtain the pseudo class labels. The cluster number varies from large to small when the hidden codes of the autoencoder need to represent higher level features. We will introduce a hyper parameter $0 < \alpha < 1$ to control the process. Suppose we have $N_c$ clusters in previous autoencoder, then the next cluster number is multiplied by $\alpha$. Through this way, the cluster number is shrinking for the deeper autoencoder, which is corresponding to our observation in Figure 1. The hyper parameter $\alpha$ is chosen so that the cluster number of the last autoencoder is approximate to the true class number in the original dataset. In addition, the agglomerative clustering continues to merge clusters from previous clustering results.

## 4 Experiments

We conduct experiments on two widely used datasets to evaluate our proposed SSA-AE for unsupervised feature learning and compare them with several state-of-the-art methods. The following describes the details of the experiments and results.

### 4.1 Results on COIL100

COIL100 contains 7,200 color images of 100 objects. The images of each objects were taken 5 degrees apart as the object is rotated on a turntable and each object has 72 images (Figure 3). The converted gray scale images with size 32 $\times$ 32 are used. We selected 10 images per object randomly to form the training set and the rest images are in testing set. The nearest neighbor classifier is applied and the inputs are

the learned feature representations. The random split is repeated 10 times, and the average results are reported with standard deviations. All of the images are used in the unsupervised feature learning and other compared methods to learn the projection function.

Our SSA-AE model has two essential parameters: the balance parameter of similarity constraint and the stacked autoencoders number. First, we focus on the parameter $\lambda$ in Eq. (1) which represents the importance of similarity constraint and fix other hyper-parameters. We conduct experiments on 20 class objects selected from COIL100 to evaluate the property of our method. Figure 4 shows how the average performance of SSA-AE varies with different $\lambda$. As we can see, when the parameter $\lambda$ becomes larger, our method achieves better recognition at first and then the performance decreases. SSA-AE achieves the best performance when the $\lambda$ is set to 5. It demonstrates that there exists a balance for the reconstruction error and the similarity constraint. Figure 5 shows how the average performance of SSA-AE varies with different stacked autoencoders number. From the results, we can observe that SSA-AE generally achieves better performance when the stacked autoencoders number goes up. However, we also notice that a much deeper structure would ruin the recognition performance. Therefore, in the experiments, we use a three-layer structure to generate the evaluation features.

To further study the performance of our approach, we compare SSA-AE with the following popular dimension reduction methods, including subspace learning: PCA [Turk and Pentland, 1991], NPE [He *et al.*, 2005], KPCA [Schlkopf *et al.*, 1998]; dictionary learning: SCC [Cai *et al.*, 2011]; nonnegative matrix factorization: SDNMF [Qian *et al.*, 2016]. In addition, three variants of the conventional autoencoder including Sparse-AE [Boureau *et al.*, 2008], Denoise-AE [Vincent *et al.*, 2010] and Graph-AE [Yu *et al.*, 2013] are added. The experiments with different numbers of used objects (20, 40, 60, 80 and 100) are conducted to evaluate the scalability of our method. Each compared method is tuned with parameters to achieve their best performance. For all four autoencoder based methods, we use the same network architecture, and report the results on the feature of the last stacked layer.

Figure 3: Samples of the dataset COIL-100.

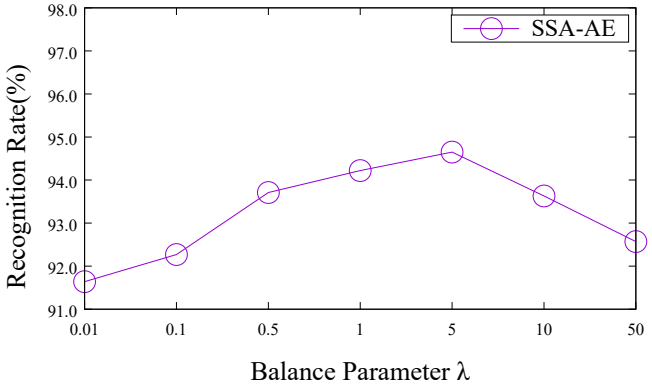


Figure 4: Average recognition rate(%) on 20 class objects selected from COIL100 with different values for $\lambda$.

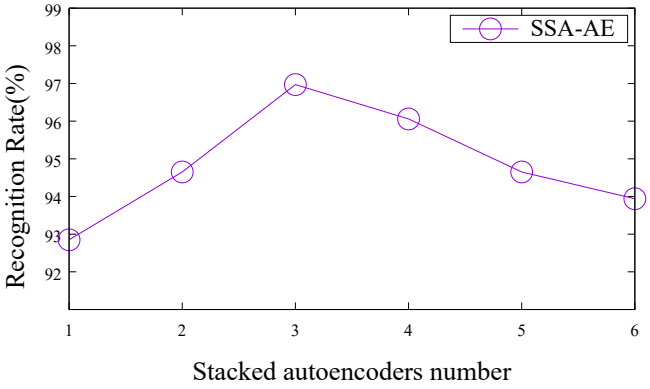


Figure 5: Average recognition rate(%) on 20 class objects selected from COIL100 with different amounts of stacked autoencoders.

Table 1 shows a large improvement on the recognition rates by our algorithm. From the results, we can observe that our method achieves consistently good performance when the number of used objects varies from 20 to 100. And autoencoder based methods are better than those traditional dimension reduction methods.

## 4.2 Results on MNIST

Furthermore, we evaluate the SSA-AE's performance on the MNIST dataset, a standard digit classification benchmark with a training set of 60,000 labeled images and a test set of 10,000 labeled images. An advantage of unsupervised learning algorithms is the ability to use them in semi-supervised scenarios where labeled data is limited. Here we use a randomly chosen subset of $N_L$ = 100, 600, 1000, and 3000 labeled images and $N_U$ = 60K unlabeled images from the training and validation set. We compare our method with many state-of-the-art semi-supervised learning methods consisting of MTC [Rifai *et al.*, 2011a], PL-DAE[Lee, 2013], WTA-AE [Makhzani and Frey, 2015], SWWAE dropout [Zhao *et al.*, 2015], M1+TSVM [Kingma *et al.*, 2014], RFM [Patel *et al.*, 2016], DRMM [Patel *et al.*, 2016] and LadderNetwork [Rasmus *et al.*, 2015].

Results are shown in Table 2 for our method with comparisons to related work. The experimental results of these compared methods are extracted from their paper. In this case, the unsupervised features are still trained on the whole dataset, but the SVM is trained only on the N labeled points where N varies from 100 to 3K. Also, we compare the performance of

Table 1: Average recognition rate(%) with standard deviations on COIL100 with different number of classes.

| Methods | 20 objects | 40 objects | 60 objects | 80 objects | 100 objects | Average |
|---|---|---|---|---|---|---|
| PCA | $90.41 \pm 0.89$ | $89.27 \pm 0.95$ | $87.20 \pm 0.80$ | $85.80 \pm 0.64$ | $84.17 \pm 0.63$ | 87.37 |
| NPE | $91.76 \pm 1.21$ | $89.59 \pm 0.77$ | $87.27 \pm 0.68$ | $85.94 \pm 0.83$ | $85.03 \pm 0.57$ | 87.91 |
| SCC | $91.49 \pm 1.02$ | $89.39 \pm 0.71$ | $85.91 \pm 0.46$ | $83.91 \pm 0.34$ | $82.91 \pm 0.42$ | 86.72 |
| KPCA | $90.12 \pm 0.80$ | $88.06 \pm 0.73$ | $85.75 \pm 0.57$ | $83.97 \pm 0.49$ | $82.52 \pm 0.53$ | 86.08 |
| SDNMF | $89.91 \pm 0.92$ | $87.88 \pm 0.62$ | $82.94 \pm 0.31$ | $81.44 \pm 0.41$ | $78.60 \pm 0.56$ | 84.15 |
| Sparse-AE | $92.26 \pm 0.68$ | $91.17 \pm 0.75$ | $88.59 \pm 0.44$ | $86.63 \pm 0.55$ | $85.54 \pm 0.49$ | 88.83 |
| Denoise-AE | $91.75 \pm 0.90$ | $90.52 \pm 0.75$ | $88.58 \pm 0.63$ | $86.35 \pm 0.66$ | $85.17 \pm 0.55$ | 88.47 |
| Graph-AE | $93.37 \pm 0.80$ | $91.33 \pm 0.58$ | $89.11 \pm 0.46$ | $86.67 \pm 0.57$ | $85.96 \pm 0.43$ | 89.28 |
| SSA-AE (ours) | $\mathbf{96.97 \pm 0.68}$ | $\mathbf{94.04 \pm 0.47}$ | $\mathbf{91.87 \pm 0.46}$ | $\mathbf{90.42 \pm 0.55}$ | $\mathbf{88.78 \pm 0.41}$ | **92.41** |

Table 2: Comparison of Test Error rates (%) between SSA-AE and other best published results on MNIST dataset for the semi-supervised setting with $N_U$ = 60K unlabeled images, of which $N_L$ = 100, 600, 1K, 3K are labeled.

| Methods | $N_L = 100$ | $N_L = 600$ | $N_L = 1K$ | $N_L = 3K$ |
|---|---|---|---|---|
| Convnet [Lecun *et al.*, 1998] | 22.98 | 7.86 | 6.45 | 3.35 |
| MTC [Rifai *et al.*, 2011a] | 12.03 | 5.13 | 3.64 | 2.57 |
| PL-DAE [Lee, 2013] | 10.49 | 5.03 | 3.46 | 2.69 |
| WTA-AE [Makhzani and Frey, 2015] | – | 2.37 | 1.92 | – |
| SWWAE dropout [Zhao *et al.*, 2015] | $8.71 \pm 0.34$ | $3.31 \pm 0.40$ | $2.83 \pm 0.10$ | $2.10 \pm 0.22$ |
| M1+TSVM [Kingma *et al.*, 2014] | 11.82 | 5.72 | 4.24 | 3.49 |
| M1+M2 [Kingma *et al.*, 2014] | $3.33 \pm 0.14$ | $2.59 \pm 0.05$ | $2.40 \pm 0.02$ | $2.18 \pm 0.04$ |
| LadderNetwork [Rasmus *et al.*, 2015] | $\mathbf{1.06 \pm 0.37}$ | – | $\mathbf{0.84 \pm 0.08}$ | – |
| RFM [Patel *et al.*, 2016] | 14.47 | 5.61 | 4.67 | 2.96 |
| DRMM unsup-pretr [Patel *et al.*, 2016] | 12.03 | 3.61 | 2.73 | 1.68 |
| DRMM semi-sup [Patel *et al.*, 2016] | 3.50 | **1.56** | 1.67 | **0.91** |
| SSA-AE (ours) | 2.78 | 2.20 | 2.29 | 1.90 |

a supervised deep Convnet [Lecun *et al.*, 1998] trained only on the N labeled training points. Despite the fact that we use no other regularization nor any hyperparameter search, the SSA-AE performs comparably to other approaches that are deep and employ various kinds of additional regularization. As we can see, SSA-AE achieves consistently good performance when N varies in a large range.

### 4.3 Implementation Details

We use Torch to implement our approach. For each single autoencoder, we stacked a combination of convolutional layer, batch normalization layer, ReLU layer and pooling layer to form the encoder. For all the convolutional layers, the number of channels is 50, and filter size is $5 \times 5$ with stride = 1 and padding = 0. Except for the experiments which evaluating the stacked autoencoder numbers, to preserve the convolutional feature maps, filter size is $3 \times 3$ with stride = 1 and padding = 1. For the decoder, we utilize the deconvolutional layer to obtain an upsampled feature map. The deconvolution operation is exactly the reverse of convolution. On the top of the hidden codes, we append an embedding layer whose dimension is 160. And we adopt stochastic gradient descent (SGD) for optimization.

## 5 Conclusions

In this paper, we proposed a novel autoencoder based approach called Stacked Similarity-Aware Autoencoders (SSA-AE) for unsupervised feature learning. To capture the geometric information in the data, we introduce a novel regularization technique. Specifically, we first obtain the pseudo class label of each sample by clustering the input features. Then for those samples sharing the same category label, their hidden codes of the autoencoder will be constrained to gather together in the high-dimensional space. This similarity constraint is implemented based on an extension of the recently proposed center loss. Through this combined cost function of the center loss and the autoencoder reconstruction error, the learned feature representations will be more discriminative and compact. As a general framework, different similarity constraint could be applied to the latent representations of the autoencoder as well. What's more, a stacked architecture is introduced to improve the representation capacity. The experimental results on several benchmark datasets indicated the effectiveness of our SSA-AE framework. Although our approach learns effective discriminative features, it has to pay high computational cost in clustering, which limits its applicability to large-scale problems. In the future, we plan to investigate more efficient way to obtain the pseudo class labels used for implementing the similarity constraint.

## Acknowledgments

This work was supported by the National Basic Research Program of China(973 Program) under Grant 2013CB336500.

## References


[Boureau *et al.*, 2008] Y-lan Boureau, Yann L Cun, et al. Sparse feature learning for deep belief networks. In *NIPS*, 2008.

[Cai *et al.*, 2011] Deng Cai, Hujun Bao, and Xiaofei He. Sparse concept coding for visual analysis. In *CVPR*, 2011.

[Ding *et al.*, 2016] Zhengming Ding, Ming Shao, and Yun Fu. Deep robust encoder through locality preserving low-rank dictionary. In *ECCV*, 2016.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[Gowda and Krishna, 1978] K. Chidananda Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern Recognition*, 10:105–112, 1978.

[He *et al.*, 2005] Xiaofei He, Deng Cai, Shuicheng Yan, and HongJiang Zhang. Neighborhood preserving embedding. In *ICCV*, 2005.

[Hinton and Salakhutdinov, 2006] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Jain *et al.*, 1999] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31:264–323, 1999.

[Kingma *et al.*, 2014] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[Lecun *et al.*, 1998] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lee, 2013] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, 2013.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[Makhzani and Frey, 2014] Alireza Makhzani and Brendan Frey. k-sparse autoencoders. In *ICLR*, 2014.

[Makhzani and Frey, 2015] Alireza Makhzani and Brendan J Frey. Winner-take-all autoencoders. In *NIPS*, 2015.

[Patel *et al.*, 2016] Ankit B Patel, Minh Tan Nguyen, and Richard Baraniuk. A probabilistic framework for deep learning. In *NIPS*, 2016.

[Plchot *et al.*, 2016] Oldrich Plchot, Lukas Burget, Hagai Aronowitz, Pavel Mat, et al. Audio enhancing with dnn autoencoder for speaker recognition. In *ICASSP*, 2016.

[Qian *et al.*, 2016] Wei Qian, Bin Hong, Deng Cai, Xiaofei He, and Xuelong Li. Non-negative matrix factorization with sinkhorn distance. In *IJCAI*, 2016.

[Rasmus *et al.*, 2015] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

[Rifai *et al.*, 2011a] Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *NIPS*, 2011.

[Rifai *et al.*, 2011b] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

[Rumelhart *et al.*, 1988] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[Schlkopf *et al.*, 1998] Bernhard Schlkopf, Alexander Smola, and Klaus Robert Mller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[Turk and Pentland, 1991] M Turk and A Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3 1:71–86, 1991.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[Wang *et al.*, 2015] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Relational stacked denoising autoencoder for tag recommendation. In *AAAI*, 2015.

[Wang *et al.*, 2016] Shuyang Wang, Zhengming Ding, and Yun Fu. Coupled marginalized auto-encoders for cross-domain multi-view learning. In *IJCAI*, 2016.

[Wen *et al.*, 2016] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, 2016.

[Yu *et al.*, 2013] Wenchao Yu, Guangxiang Zeng, Ping Luo, Fuzhen Zhuang, Qing He, and Zhongzhi Shi. Embedding with autoencoder regularization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

[Zhao *et al.*, 2015] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

[Zhao *et al.*, 2016] Zhou Zhao, Xiaofei He, Deng Cai, Lijun Zhang, Wilfred Ng, and Yueting Zhuang. Graph regularized feature selection with data reconstruction. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):689–700, 2016.