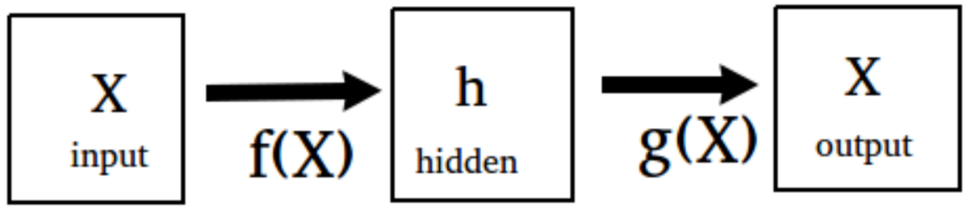
Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction

Wei Wang, Yang Huang, Yizhou Wang, Liang Wang

Autoencoder

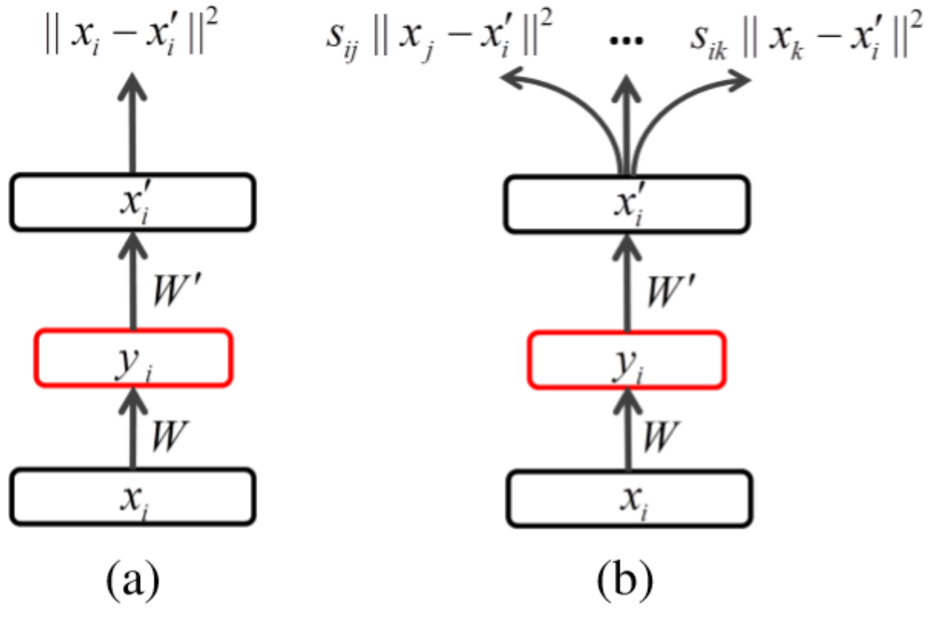
- The autoencoder algorithm belongs to a special family of dimensionality reduction methods implemented using artificial neural network.
- It tries to reconstruct the inputs at the outputs.
- It aims to learn a compressed representation for an input through minimizing its reconstruction error.



Autoencoder

In the **traditional autoencoder**, x_i is only used to reconstruct itself and the reconstruction error $\|x_i - x_i'\|^2$ just measures the distances between x_i and x_i'

In the **generalized autoencoder**, x_i involves in the reconstruction of a set of instances $\{x_j, x_k, \ldots\}$.
Each reconstruction error $s_{ij}\|x_j - x_i'\|^2$ measures a weighted distance between x_j and x_i'



Traditional autoencoder Generalized autoencoder

Generalized Autoencoder

The encoder maps an input $x_i \in R^{d_x}$ to a reduced hidden representation $y_i \in R^{d_y}$ by a function $g()$

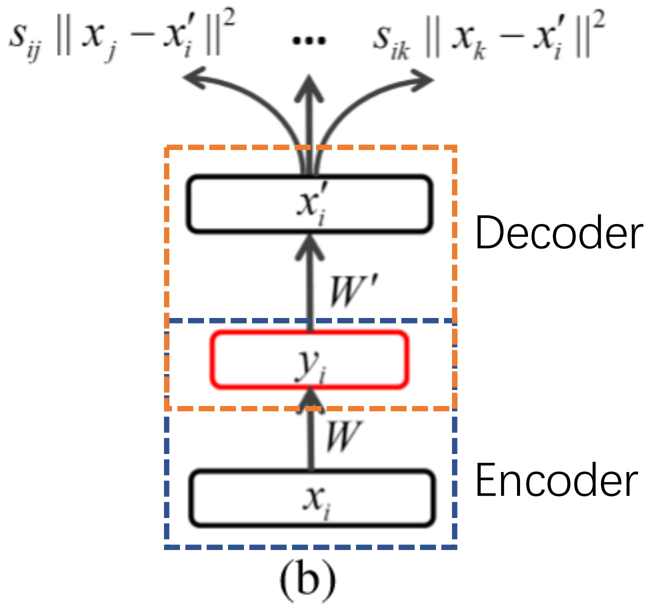
$$y_i = g(Wx_i)$$

where $g(\,)$ is either the identity function (恒等函数) for a linear projection or a sigmoid function for a nonlinear mapping. The parameter W is a $d_y \times d_x$ weight matrix.

The decoder reconstructs $x_i' \in R^{d_x}$ from the hidden representation y_i

$$x_i' = f(W' y_i)$$

the parameter W' is another $d_x \times d_y$ weight matrix, which can be W^T



To model the relation between data, the decoder reconstructs a set of instances indexed by $\Omega_i = \{j, k, \dots\}$ with specific weights $S_i = \{s_{ij}, s_{ik}, \dots\}$

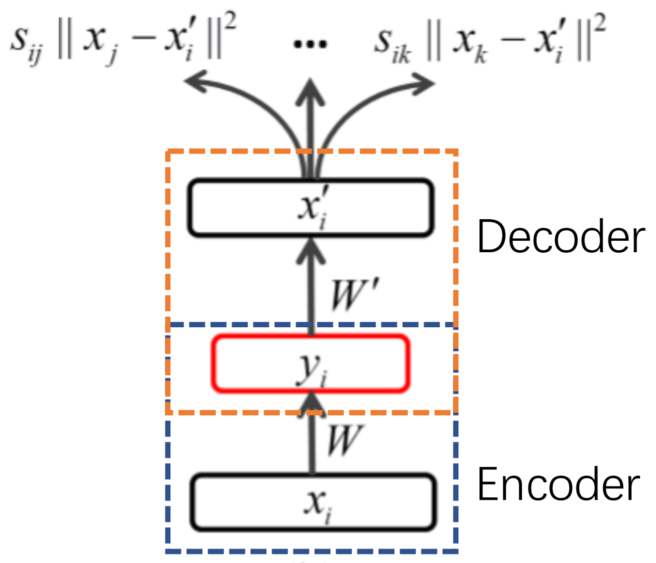
The weighted reconstruction error is

$$e_i(W, W') = \sum_{j \in \Omega_i} s_{ij} L(x_j, x_i')$$

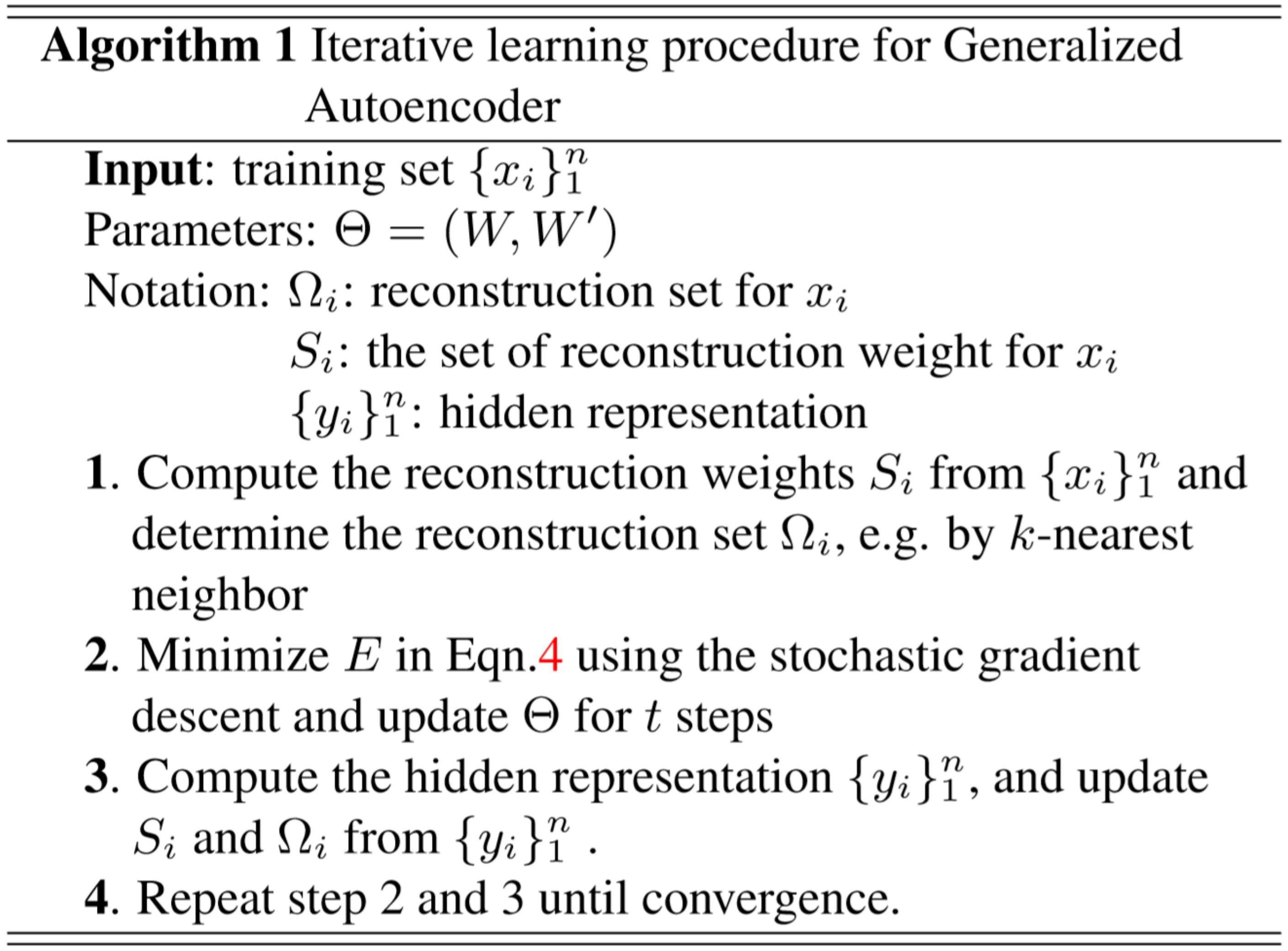
The total reconstruction error $E\ of\ n$ sample is

$$E(W, W') = \sum_{i=1}^{n} e_i(W, W') = \sum_{i=1}^{n} \sum_{j \in \Omega_i} s_{ij} L(x_j, x_i')$$

Generalized Autoencoder



(b)



```
Algorithm 1 Iterative learning procedure for Generalized Autoencoder
  Input: training set $\{x_i\}_1^n$
  Parameters: $\Theta = (W, W')$
  Notation: $\Omega_i$: reconstruction set for $x_i$
            $S_i$: the set of reconstruction weight for $x_i$
            $\{y_i\}_1^n$: hidden representation
  1. Compute the reconstruction weights $S_i$ from $\{x_i\}_1^n$ and
     determine the reconstruction set $\Omega_i$, e.g. by $k$-nearest
     neighbor
  2. Minimize $E$ in Eqn.4 using the stochastic gradient
     descent and update $\Theta$ for $t$ steps
  3. Compute the hidden representation $\{y_i\}_1^n$, and update
     $S_i$ and $\Omega_i$ from $\{y_i\}_1^n$ .
  4. Repeat step 2 and 3 until convergence.
```

Connection to Graph Embedding

Graph embedding is known to be a general framework for dimensionality reduction, of which each data is represented as graph mode in a low-dimensional vector, the edges preserve similarities between data pairs.

$$y^* = arg \min_{y^T B y = c} \sum_{i,j} s_{ij} \left\| y_i - y_j \right\|^2$$

where y is the low-dimensional representation. s_{ij} is the similarity between the vertex i and j. c is a constant and B is constraint matrix.

The linear graph embedding (LGE) assumes that low-dimensional representation can be obtained by a linear projection $y = X^T w$, where w is the projection vector and X=[x_1, $x_2 \cdots$, x_n]. The objection function of LGF is

$$w^* = arg \min_{\substack{w^T XBXw = c \\ or w^T w = c}} \sum_{i,j} s_{ij} \left\| w^T x_i - w^T x_j \right\|^2$$

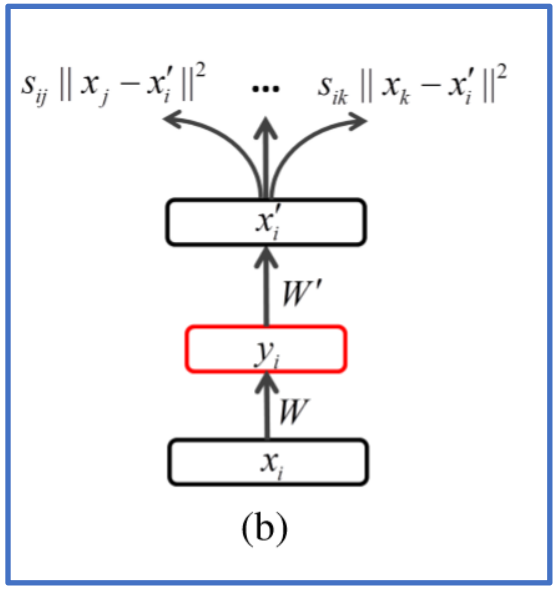
Connection to Graph Embedding

The total reconstruction error :

$$E(W,W') = \sum_{i=1}^{n} e_i(W,W') = \sum_{i=1}^{n} \sum_{j\in\Omega_i} s_{ij} L(x_j, x_i') \qquad (4)$$

Objective function in generalized autoencoder :

$$(W,W') = argmin \textstyle\sum_{i,j} s_{ij} \left\| x_j - f(W' g(W x_i)) \right\|^2) \qquad (7)$$



(b)

In the linear reconstruction case, if $W' = W^T$, and assuming only one hidden node in the network, W degenerates to a column vector w , the objective function (7) becomes

$$w^* = \arg\min \sum_{i,j} s_{ij} ||x_j - ww^T x_i||^2 \qquad (8)$$

Let $w^T w = c$, and $y_i = w^T x_i$, Eqn. 8 becomes

$$w^* = \arg \min_{w^T w = c} \sum_{i,j} s_{ij} (||w^T x_i - w^T x_j||^2 + (\frac{c}{2} - 1) y_i^2) \quad (9)$$

Connection to Graph Embedding

$$w^* = \arg\min \sum_{i,j} s_{ij} ||\boxed{x_j - ww^T x_i||^2} \qquad (8)$$

$$\|x_j - ww^T x_i\|^2$$

$$= x_j^T x_j + x_i^T ww^T ww^T x_i - 2x_j^T ww^T x_i$$

$$= x_j^T x_j + c x_i^T ww^T x_i - 2\cancel{x_j^T w}w^T x_i \qquad \text{(A)}$$

$$w^* = \arg \min_{w^T w = c} \sum_{i,j} s_{ij} \boxed{(||w^T x_i - w^T x_j||^2 + (\frac{c}{2} - 1) y_i^2)} \qquad (9)$$

$$\|w^T x_i - w^T x_j\|^2$$

$$= x_i^T ww^T x_i + x_j^T ww^T x_j - 2\cancel{x_i^T w}w^T x_j \qquad \text{(B)}$$

$$\text{(A)} - \text{(B)} \quad = x_j^T x_j - x_j^T ww^T x_j + (c-1) x_i^T ww^T x_i$$

$$= x_j^T x_j - x_j^T ww^T x_j + (c-1) x_i^T ww^T x_i$$

$$= x_j^T x_j - y_j^2 + (c-1) y_i^2$$

Implementation of the GAE Variants

As can be seen from the above analysis, the generalized autoencoder can also be a general framework for dimensionality reduction through defining different reconstruction sets and weights.

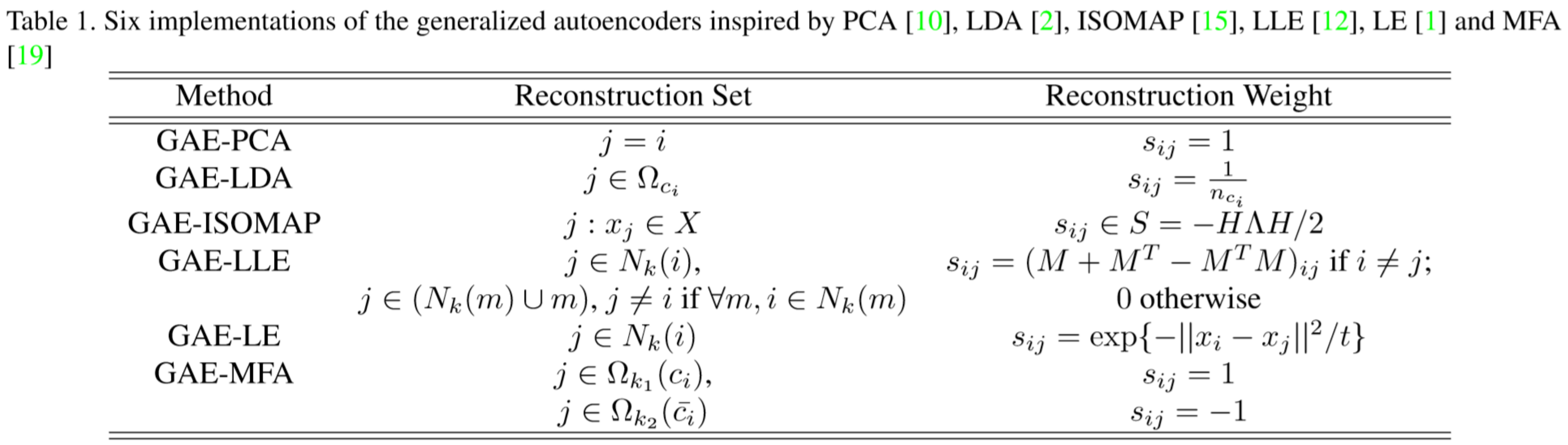


Table 1. Six implementations of the generalized autoencoders inspired by PCA [10], LDA [2], ISOMAP [15], LLE [12], LE [1] and MFA [19]

Method	Reconstruction Set	Reconstruction Weight
GAE-PCA	$j = i$	$s_{ij} = 1$
GAE-LDA	$j \in \Omega_{c_i}$	$s_{ij} = \frac{1}{n_{c_i}}$
GAE-ISOMAP	$j : x_j \in X$	$s_{ij} \in S = -H\Lambda H/2$
GAE-LLE	$j \in N_k(i)$,	$s_{ij} = (M + M^T - M^T M)_{ij}$ if $i \neq j$;
	$j \in (N_k(m) \cup m), j \neq i$ if $\forall m, i \in N_k(m)$	0 otherwise
GAE-LE	$j \in N_k(i)$	$s_{ij} = \exp\{-\|\|x_i - x_j\|\|^2/t\}$
GAE-MFA	$j \in \Omega_{k_1}(c_i)$,	$s_{ij} = 1$
	$j \in \Omega_{k_2}(\bar{c_i})$	$s_{ij} = -1$

Deep Generalized Autoencoder

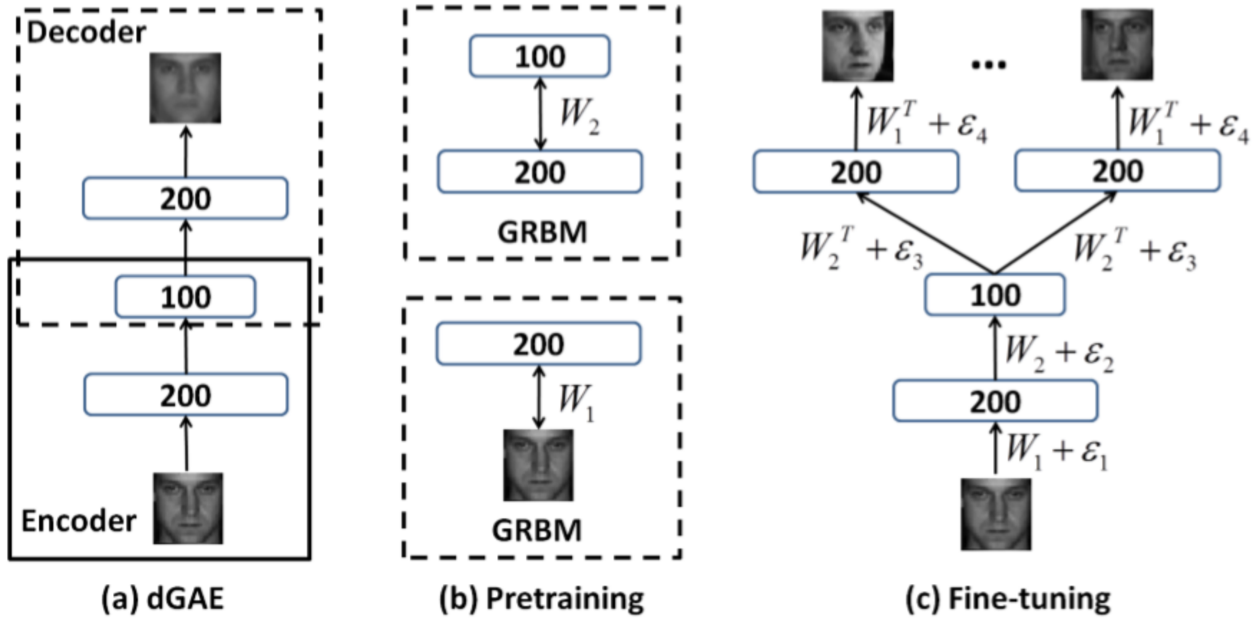


Figure 2. Training a deep generalized autoencoder (dGAE) on the face dataset used in Section 3.3. (a) The dGAE consists of a three-layer encoder network and a three-layer decoder network. A reconstructed face is in the output layer. (b) Pretraining the dGAE through learning a stack of Gaussian restricted Boltzmann machines (GRBM). (c) Fine-tuning a deep GAE-LDA.

“With large initial weights, autoencoders typically find poor local minima. ⋯ If the initial weights are close to a good solution, gradient descent works well. ⋯ We introduce this pretraining procedure for binary data, generalize it to real”

—— *G.E. Hinton “Reducing the Dimensionality of Data with Neural Networks ” 2006*

Experimental Results

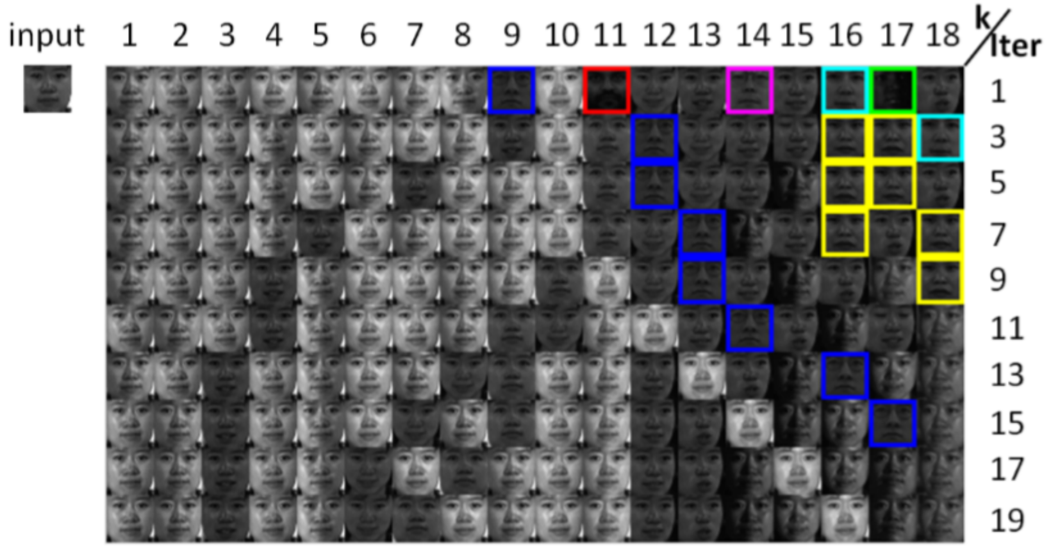


Figure 3. The changes of nearest neighbors during the iterative learning. Each row shows the 18 nearest neighbors of an input data during in one iteration of learning. The faces in the colored boxes belong to other persons. The color indicates the identity of the persons. We can see that along the iterative learning, the nearest neighbors of a data converges to its own class.

Fig 3 shows the changes of a face’ s 18 nearest neighbors during the first 20 iterations of our unsupervised dGAE-LE learning on the CMU PIE dataset

Experimental Results

CMU PIE dataset

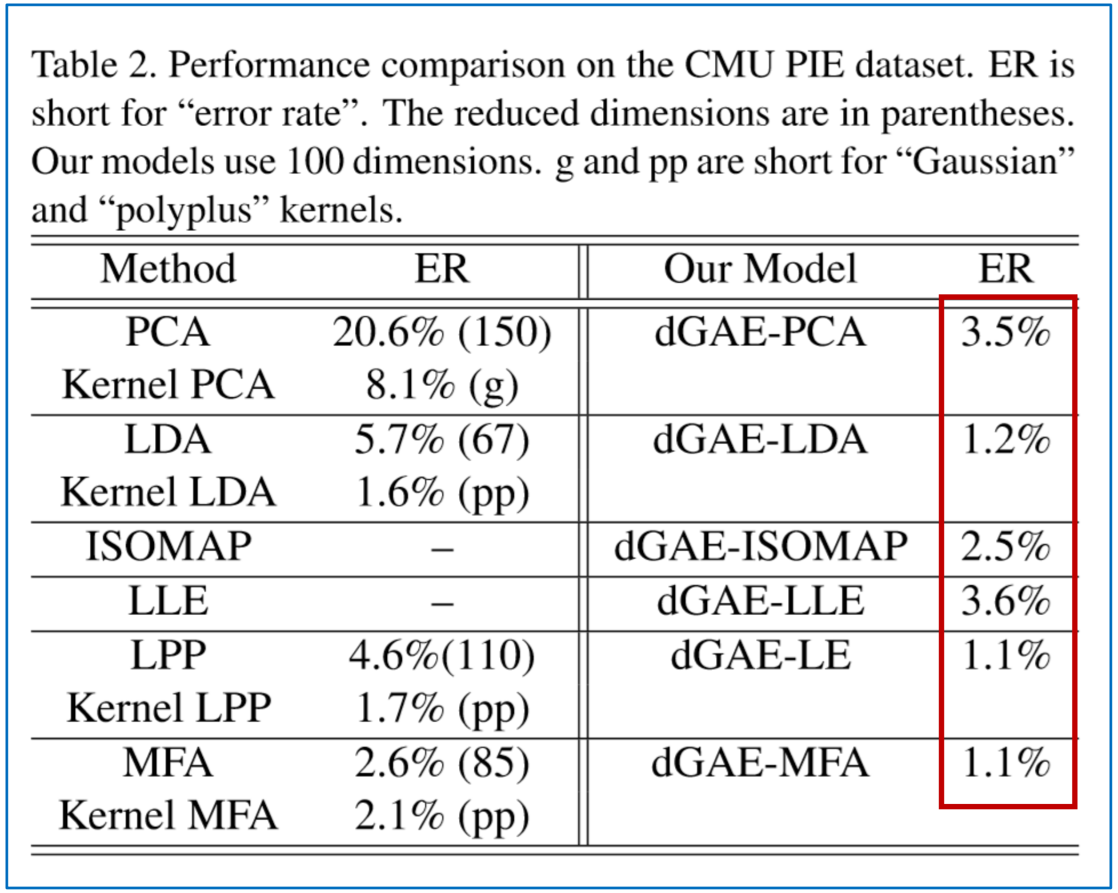


Table 2. Performance comparison on the CMU PIE dataset. ER is short for “error rate”. The reduced dimensions are in parentheses. Our models use 100 dimensions. g and pp are short for “Gaussian” and “polyplus” kernels.

Method	ER	Our Model	ER
PCA	20.6% (150)	dGAE-PCA	3.5%
Kernel PCA	8.1% (g)		
LDA	5.7% (67)	dGAE-LDA	1.2%
Kernel LDA	1.6% (pp)		
ISOMAP	–	dGAE-ISOMAP	2.5%
LLE	–	dGAE-LLE	3.6%
LPP	4.6%(110)	dGAE-LE	1.1%
Kernel LPP	1.7% (pp)		
MFA	2.6% (85)	dGAE-MFA	1.1%
Kernel MFA	2.1% (pp)		

MNIST dataset

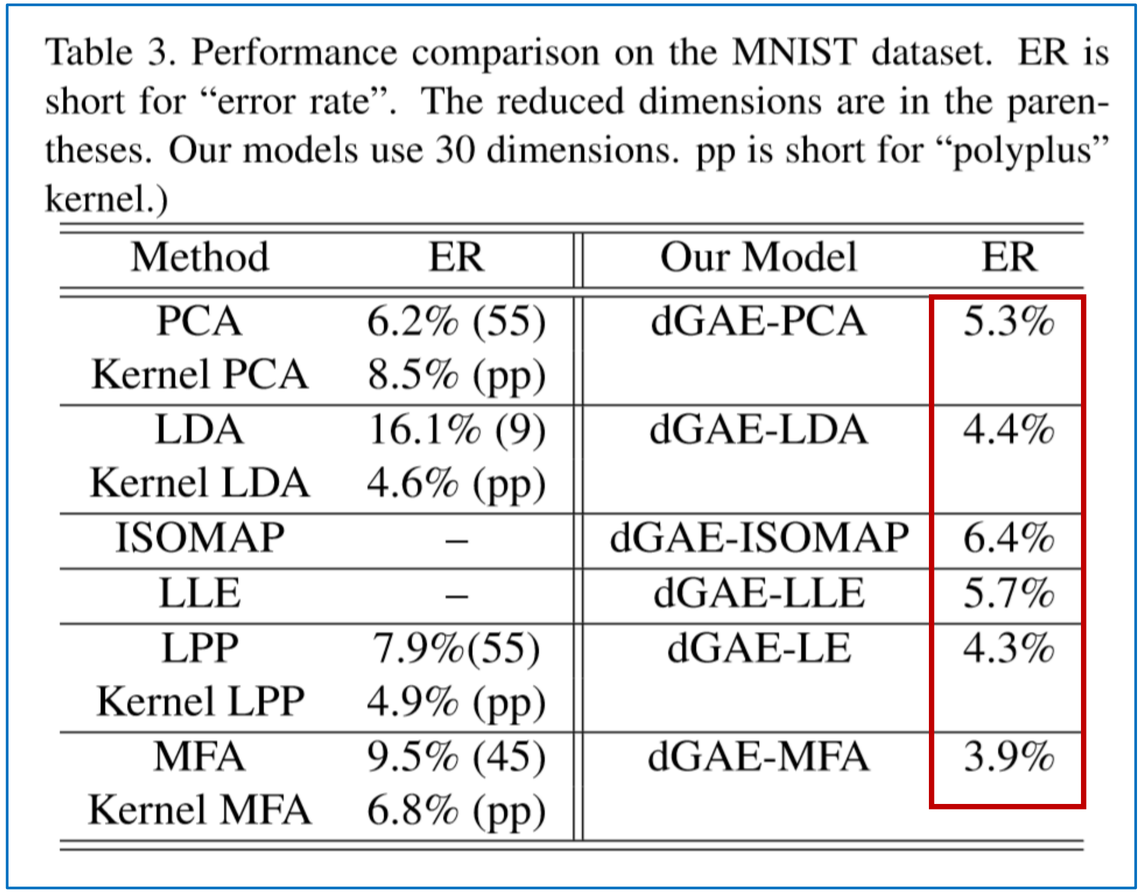
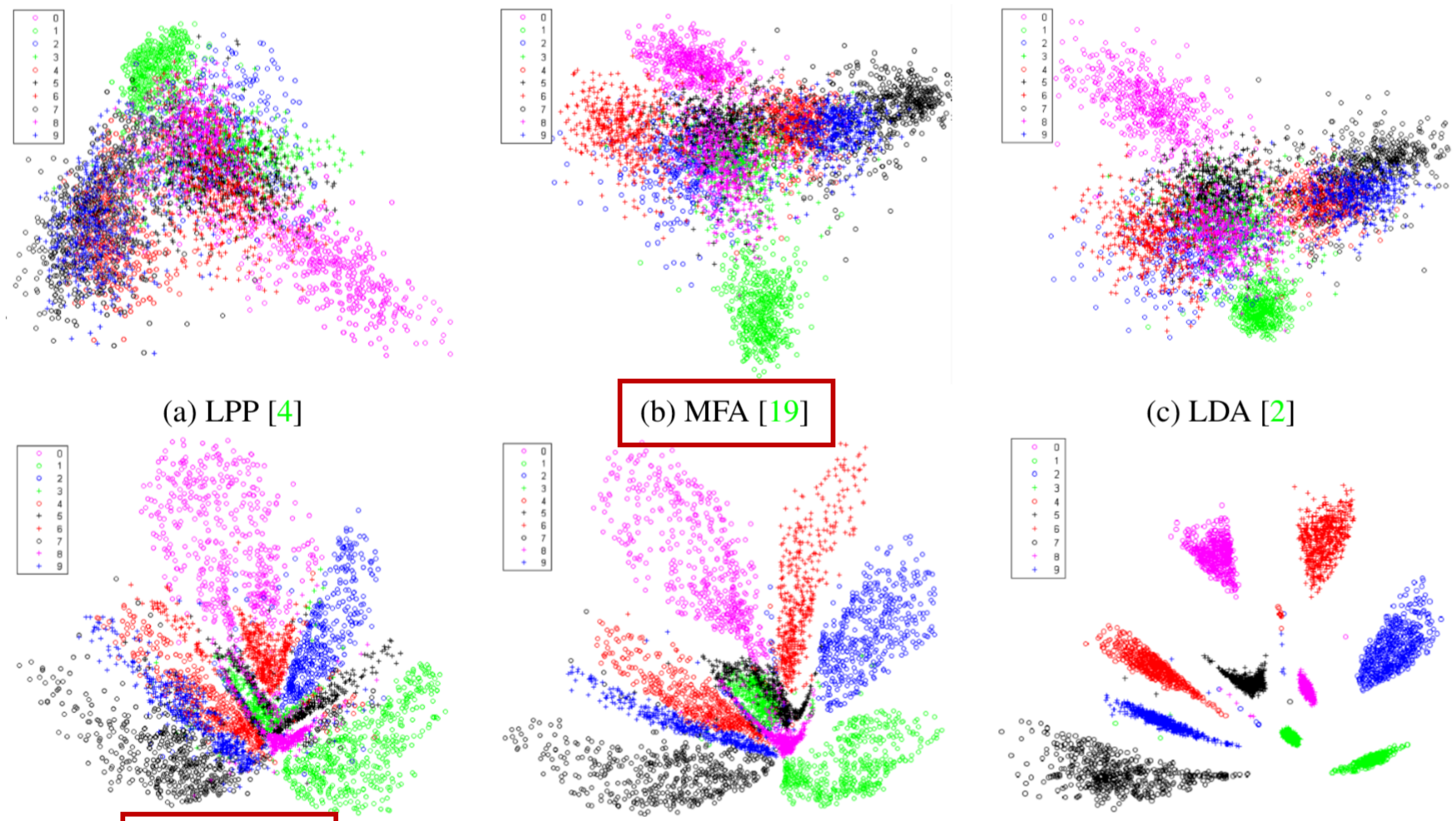


Table 3. Performance comparison on the MNIST dataset. ER is short for “error rate”. The reduced dimensions are in the parentheses. Our models use 30 dimensions. pp is short for “polyplus” kernel.)

Method	ER	Our Model	ER
PCA	6.2% (55)	dGAE-PCA	5.3%
Kernel PCA	8.5% (pp)		
LDA	16.1% (9)	dGAE-LDA	4.4%
Kernel LDA	4.6% (pp)		
ISOMAP	–	dGAE-ISOMAP	6.4%
LLE	–	dGAE-LLE	5.7%
LPP	7.9%(55)	dGAE-LE	4.3%
Kernel LPP	4.9% (pp)		
MFA	9.5% (45)	dGAE-MFA	3.9%
Kernel MFA	6.8% (pp)		

LPP is a popular linear approximation to the LE
“Locality preserving projections.”

Experimental Results



(a) LPP [4] (b) MFA [19] (c) LDA [2]

(d) dGAE-LE (e) dGAE-MFA (f) dGAE-LDA

Figure 6. 2D visualization of the learned digit image manifold.

Thank You!