Neural Processing Letters (2020) 51:2649–2667
https://doi.org/10.1007/s11063-020-10223-w

Stacked Fusion Supervised Auto-encoder with an Additional Classification Layer

Rui Li[1] · Xiaodan Wang[1] · Wen Quan[2] · Lei Lei[1]

Published online: 4 March 2020

Abstract
Auto-encoders are unsupervised deep learning models, which try to learn hidden representations to reconstruct the inputs. While the learned representations are suitable for applications related to unsupervised reconstruction, they may not be optimal for classification. In this paper, we propose a supervised auto-encoder (SupAE) with an addition classification layer on the representation layer to jointly predict targets and reconstruct inputs, so it can learn discriminative features specifically for classification tasks. We stack several SupAE and apply a greedy layer-by-layer training approach to learn the stacked supervised auto-encoder (SSupAE). Then an adaptive weighted majority voting algorithm is proposed to fuse the prediction results of SupAE and the SSupAE, because each individual SupAE and the final SSupAE can both get the posterior probability information of samples belong to each class, we introduce Shannon entropy to measure the classification ability for different samples based on the posterior probability information, and assign high weight to sample with low entropy, thus more reasonable weights are assigned to different samples adaptively. Finally, we fuse the different results of classification layer with the proposed adaptive weighted majority voting algorithm to get the final recognition results. Experimental results on several classification datasets show that our model can learn discriminative features and improve the classification performance significantly.

Keywords Deep learning · Supervised learning · Auto-encoder · Adaptive weighted voting fusion

1 Introduction

Deep learning algorithms [1, 2] have been proposed in recent years to move machine learning systems towards the discovery of multiple levels of representation, and have attracted much attentions both from the academic and industrial communities. The main concept of deep leaning is automatically extracting complex data representation by developing a hier-

✉ Wen Quan
wen_quanafeu@163.com

[1] College of Air and Missile Defense, Air Force Engineering University, Xi'an 710051, China

[2] College of Air Traffic Control and Navigation, Air Force Engineering University, Xi'an 710051, China

archical architecture, which composes many non-linear transformations, each transform the representations at a low level into representations at a higher, slightly more abstract level. Deep learning algorithms often yield better results in different machine learning applications, including computer vision [3–8], speech recognition [9], natural language processing [10].

Auto-encoder [11–14] is one of the most common deep learning methods for unsupervised representation learning, it consists of two modules, an encoder which encode the inputs to hidden representations and a decoder which attempts to reconstruct the inputs from the hidden representations. The hidden representations can retain as much as information of the inputs and capture the posterior distribution of the underlying explanatory factors for the observed inputs. Generally, auto-encoder can be used for pre-training greedily and initialization for a stacked auto-encoder [11], where the encoded intermediate representations of each auto-encode are fed to next one layer-wise. In recent years, auto-encoder has attracted great attentions due to its simple implementation, various modifications of auto-encoder have been proposed. Sparse auto-encoders (SpAE) [12] encouraged the activations of neurons to be inactive most of the time with the KL divergence constraint, in order to exploit the hidden structures of the data and learn sparse representations. Denoising auto-encoder (DAE) [13] reconstructed the clean input with the corrupted data to capture the stable structures of the input distribution and increases the robustness of the model. Contractive auto-encoder (CAE) [14] learned robust representations to infinitesimal input variations, by adding a contractive penalty with Frobenius norm Jacobian matrix of hidden activations. Higher version of contractive auto-encoder [15] regularized the norm of the Hessian matrix of the hidden representation to favor smooth manifold, Hessian matrix can properly exploit the intrinsic local geometry of the data manifold [16]. Auto-encoder and its variants show excellent performance on a variety of tasks, including natural data analysis [17], language processing [18], image processing [19], and object detection [20].

We notice that the SpAE, DAE and CAE are trained in an unsupervised manner, they are non-discriminative as they do not utilize class information. Supervised learning method learns similar features within class and dissimilar features otherwise, which can improve classification performance. Use of supervision for auto-encoders has been explored recently, Du et al. [21] trained a supervised auto-encoder over the noisy concatenated data and label, so it taken the label information into account during feature detection for auto-encoder straightforwardly. Singh et al. [22] presented a class representative auto-encoder which aimed at learning discriminative features in nature by incorporating inter-class and intra-class variations at the time of feature learning process. Gao et al. [23] proposed a supervised auto-encoder which imposed similarity preservation term of same class samples to capture the discriminative structures of face images. Group sparse auto-encoder (GSAE) [24] introduced supervision using group sparse regularization on representations of same class to learn class-specific features. Large margin auto-encoders (LMAE) [25] added large margin penalty in hidden feature space to encourage the hidden representations to be large marginally distributed. Most of these algorithms incorporated class information at the time of feature extraction with the aim of reducing the intra-class variations and increasing the inter-class variations. Some other methods incorporated graph Laplacian regularization into auto-encoders from the manifold learning perspective, which encouraged the learned representation to preserve the local connectivity for data points on the manifold, such as Laplacian auto-encoder (LAE) [26], graph regularized auto-encoder (GAE) [27], and discriminative auto-encoders [28].

For unsupervised learning, a good representation is often one that captures the posterior distribution of the underlying explanatory factors for the observed input. For supervised learning, a good representation is often one that is discriminative for supervised predictors. In this paper, in order to extract discriminative representation with auto-encoder, we designed a

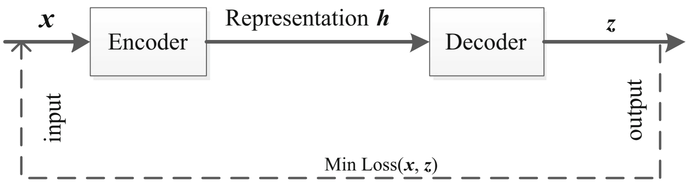

Fig. 1 The generic flowchart of an auto-encoder

supervised auto-encoder (SupAE) with an additional classification layer on the representation layer to jointly predict targets and reconstruct inputs, then we build a stacked supervised auto-encoder (SSupAE) for classification tasks. In both pre-training and fine-tuning stage, SupAE and SSupAE can output the classification results, we consider to fuse them from the classifiers ensemble perspective, so we further proposed a new adaptive majority weighted voting algorithm to fuse the results of the each individual SupAE and the SSupAE. Thus both taking the label information into account during pre-training and the classifiers ensemble can contribute the improvement of classification performance. The contributions of the proposed work are as follows:

(1) We propose a SupAE, it consists of three parts: an encoder to encode the input, a decoder to reconstruct the input, and a classifier to predict the target. In this way, the combination of the reconstruction loss and the classification loss has the promise to both balance extracting underlying structure in data, as well as learning discriminative representation, thus the unsupervised learning can be converted to supervised learning by designing a joint label-related cost function and reconstruction loss.
(2) With our SupAE as building block to initialize a multi-layer neural network, we can build a SSupAE for classification tasks specifically, SSupAE incorporate the label information in both pre-training and fine-tuning stage, so it can provide more accurate prediction performance.
(3) We further propose an adaptive weighted majority voting algorithm to fuse the prediction results of SupAE and the SSupAE. Because each individual SupAE and the final SSupAE can both get the posterior probability information of samples belong to each class, we introduce Shannon entropy based on the posterior probability information to measure the classification ability for different samples, and assigns high weight to sample with low entropy, thus more reasonable weights are assigned to different samples adaptively. Finally, we fuse the different results of classification layer to improve the classification performance with the proposed adaptive weighted majority voting algorithm.

The rest of the paper is organized as follows. Section 2 overviews related work about auto-encoders. Section 3 present the proposed framework. Experimental results and analysis are provided in Sect. 4. Section 5 includes conclusion.

2 Brief Review of Auto-encoder and Its Variants

Auto-encoder [11] is a typical unsupervised neural network, which aims at learning hidden representation of data based on an encoder-decoder paradigm, as shown in Fig. 1.

The encoder map the input $\boldsymbol{x}$ to the latent representation space $\boldsymbol{h}$ with a non-linear function f, such as the sigmoid function, the hyperbolic tangent function, and ReLu function.

$$\boldsymbol{h} = f(\boldsymbol{x}) = f(\mathbf{W}\boldsymbol{x} + \boldsymbol{b}) \tag{1}$$

where $\mathbf{W}$ is the weight matrix used for encoding the input $\boldsymbol{x}$, and $\boldsymbol{b}$ is bias vector.

The decoder attempts to transform the latent representation $\boldsymbol{h}$ to output z for the reconstruction of input.

$$z = \mathrm{g}(\boldsymbol{h}) = \mathrm{g}(\mathbf{W}'\boldsymbol{h} + \boldsymbol{b}') \tag{2}$$

where g is activation function, $\mathbf{W}'$ is weight matrix, $\boldsymbol{b}'$ is bias vector. The tied weights strategy $\mathbf{W}' = \mathbf{W}^T$ has been usually employed to simplify the network architecture.

Auto-encoder attempts to reconstruct the input with output as much as possible, so the objective function to train an auto-encoder can be defined as:

$$J_{AE}(\theta) = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathbf{X}} \|\boldsymbol{x} - g(f(\boldsymbol{x}))\|_2^2 + \lambda R(\mathbf{W}) \tag{3}$$

where $\theta = (\mathbf{W}, \boldsymbol{b}, \boldsymbol{b}')$ is the set of parameters, N is the samples number of training dataset $\mathbf{X}$, $R(\mathbf{W})$ is the regularization term to prevent overfitting, such as L2-norm regularization, λ is the regularization parameter. Minimizing reconstruction error is usually carried out by gradient descent based algorithms [29].

To exploit the hidden structure of the data and learn sparse representation, the sparse auto-encoder(SpAE) [12] imposes the sparsity constraint on the hidden units. Generally, a neuron whose output is close to one is active, while a neuron whose output is close to zero is inactive. Suppose $\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{n} h(\boldsymbol{x}_i)$ is the average activation value of the jth hidden unit, a sparse auto-encoder limits the activations of neurons $\hat{\rho}_j = \rho$ (ρ is typically a small number close to zero) so that they are inactive most of the time. The training objective of the SpAE can be denoted as:

$$J_{SpAE}(\theta) = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathbf{X}} l(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \beta\sum_{j=1}^{N} KL(\rho||\hat{\rho}_j) \tag{4}$$

where $KL(\rho||\hat{\rho}_j) = \rho\log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}$ is KL divergence, β is the sparsity penalty parameter.

In the case of denoising auto-encoder (DAE) [13], it reconstructs the clean input with the corrupted data, the training criterion for DAE is expressed as:

$$J_{DAE}(\theta) = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathbf{X}} \|\boldsymbol{x} - g(f(\tilde{\boldsymbol{x}}))\|_2^2 + \lambda R(\mathbf{W}) \tag{5}$$

where $\tilde{\boldsymbol{x}}$ is the stochastically corrupted input, while $\boldsymbol{x}$ is the clean input. DAE can capture the stable structure of the input distribution and increase the robustness of the model.

Contractive auto-encoder (CAE) [14] aims at minimizing the sensitivity of the hidden representation to slight changes in input for the purpose of leaning robust representations. The objective function of CAE is given by:

$$J_{CAE}(\theta) = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathbf{X}} l(\boldsymbol{x}, g(f(\boldsymbol{x})))) + \|J(\mathbf{X})\|_2^2 \tag{6}$$

where $J(\mathbf{X})$ denotes the Jacobian matrix of X. CAE penalizes the sensitivity of the features, and the penalty is analytic. So CAE is notable different with DAE, although they share a similar motivation of learning robust representations.

Laplacian auto-encoder [26] encourages the learned representation to preserve the local connectivity for data points on the manifold:

$$J_{LAE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|_2^2 + \frac{\lambda}{2}\sum_{i \neq j} w_{ij} \|f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j)\|_2^2 \tag{7}$$

where the second term is the local graph embedding regularization, w_{ij} is the similarity between sample $\boldsymbol{x}_i$ and $\mathbf{x}_j$, $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$ are the corresponding representations. LAE added the local graph embedding regularization to enforce mapping the neighboring samples close together in the embedding space, so it can capture discriminative information and preserve some geometric similarity of the data.

Auto-encoder and its variants can learn underlying explanatory representation of data. Once the auto-encoder is trained, the decoder module is abandoned, and the encoder module is stacked greedy layer-by-layer to construct the stacked auto-encoder, the final representations provide useful information which can be used as features for building classifiers. On the top of last layer, a softmax classifier is added for supervised training. Stacking up auto-encoder can learn more abstract and complicated representations of data by composing representations acquired in a hierarchical architecture.

3 Proposed Method

We intentionally want to learn discriminative feature with auto-encoder, so from the point view of supervised learning, we investigate a SupAE to jointly predict targets and reconstruct inputs, and then build a SSupAE specifically for the classification tasks. From the point view of classifiers ensemble, we proposed an adaptive weighted voting algorithm to fuse the results of each individual SupAE and the final SSupAE to improve the performance. So our proposed stacked fusion supervised auto-encoder(SFSupAE) framework consists of two phases: (1) Supervised pre-training and fine-tuning for SSupAE. (2) Adaptive weighted voting fusion recognition. Next, we describe the formulation of the proposed SFSupAE framework.

3.1 Supervised Pre-training and Fine-Tuning for Stacked Supervised Auto-encoder

For a classification problem with N training samples and m classes, suppose that the dataset is $\{\mathbf{X}, \mathbf{Y}\} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | \boldsymbol{x}_i \in R^d, \boldsymbol{y}_i \in R^m\}_{i=1}^N$, where $\boldsymbol{x}_i$ is the ith training data, d is the dimension of data, $\boldsymbol{y}_i$ is the corresponding one-hot label vector of ith sample. When using the stacked auto-encoder for prediction, we usually train several auto-encoders greedily in an unsupervised manner and stack them to construct stacked auto-encoder. In the pre-training stage, auto-encoders aim at capturing the underlying explanatory factors for the observed input, but does the learned representations are discriminative for data separation? We think the underlying representative structures of the data not always be optimal for classification. So we investigate a SupAE to jointly predict targets and reconstruct inputs. Our SupAE consist three part: an encoder to encode the input $\boldsymbol{x}$, a decoder to reconstruct $\boldsymbol{x}$, a classifier to predict $\boldsymbol{y}$, as shown in Fig. 2.

The decoder attempts to map the latent representation $\boldsymbol{h}$ to output $\boldsymbol{z}$ for reconstruction of the input:

$$\boldsymbol{z} = \mathrm{g}(\boldsymbol{h}) = \mathrm{g}(\mathbf{W}'\boldsymbol{h} + \boldsymbol{b}') \tag{8}$$

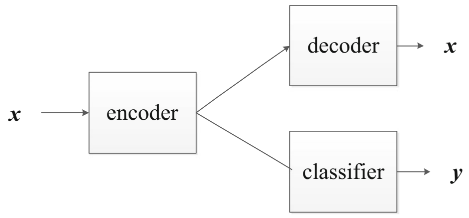

Fig. 2 Supervised auto-encoder with an addition classification layer

The classifier predicts the $\boldsymbol{y}$ with the latent representation $\boldsymbol{h}$:

$$\boldsymbol{y}^{'} = \mathrm{s}(\boldsymbol{h}) = \mathrm{s}(\mathbf{W}_c\boldsymbol{h} + \boldsymbol{b}_c) \tag{9}$$

The classifier is parameterized by weight matrix $\mathbf{W}_c$, and bias vector $\boldsymbol{b}_c$. s is the activation function of classifier, we use softmax function, in this case, the classification assignments are formulated as probabilities, continuous values between 0 and 1.

The aim of SupAE is learning not only representative but also discriminative features. The objective can be denoted as:

$$J_{SupAE}(\theta) = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathbf{X}}(L(\boldsymbol{x}, \boldsymbol{z}) + \lambda L(\boldsymbol{y}', \boldsymbol{y})) \tag{10}$$

where $L(\boldsymbol{x}, \boldsymbol{z})$ is the reconstruction loss, such as mean squared error loss, $L(\boldsymbol{y}^{'}, \boldsymbol{y})$ is the supervised loss, such as cross-entropy loss, mean squared error loss. λ is the parameter balance reconstruction loss and the supervised loss. The encoder and the decoder make up the standard auto-encoder, but along with the flexible classifier, we can combine the reconstruction loss and the classification loss to both balance extracting underlying structure, as well as providing discriminative features. The additional supervised loss can better direct representation learning towards representations those are effective for the classification tasks. Thus the unsupervised learning can be converted to supervised learning by designing a joint label-related cost function and reconstruction loss.

After the SupAE is pre-trained, the classifier and decoder are abandoned, we can take the intermediate representations learned from the previous SupAE to the next one layer-wise to extract complex discriminative representations at high levels of abstractions. At the end, another softmax classification layer is put on top of the last representation layer, and the whole network is fine-tuned in a supervised manner. The proposed stacked supervised auto-encoder framework is shown in Fig. 3.

The SSupAE encode the label information into hidden layer of SupAE, the learned parameters will extract not only representative but also discriminative features. Then, the fine-tuning process will further speed up the network convergence and the performance will be improved.

3.2 Adaptive Weighted Majority Voting Fusion

Training several classifiers at the same time to solve same problem, and then combining their outputs to improve classification accuracy, is known as ensemble method [30, 31]. Ensemble method seeks to combine the predictions of multiple classifiers to obtain better predictive performance, because different models can complement each other by appropriate compensation of weaknesses and strengths of the individual models. When working with classifiers ensemble, one important issue to be taken into consideration is related to the selection of an efficient combination method. Many combination methods have been proposed

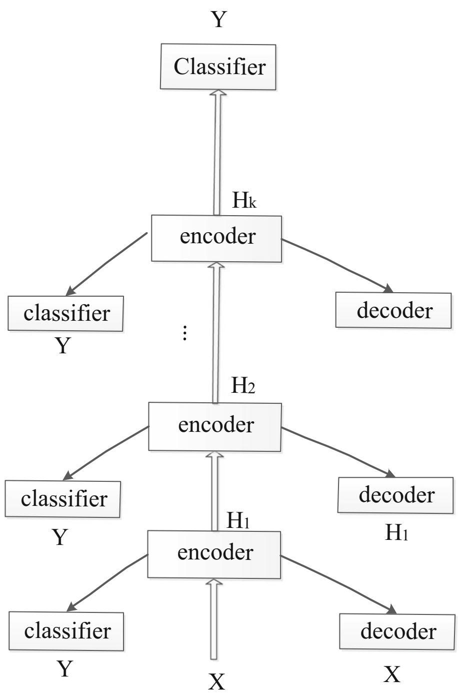

Fig. 3 Stacked supervised auto-encoder framework, X is the input dataset, Y is label, $\mathbf{H}_i, \ i = 1, 2, \ldots k$ is the hidden representation

for classifier ensembles, such as bagging [30], boosting [30, 31], evidence theory [32, 33], error correcting output coding(ECOC) [34, 35], and weighted majority vote [30, 31, 36–38]. Among them, weighted majority vote is a simple but effective ensemble learning algorithm, which allows one to classification via voting on the outputs of several classifiers. The key issues for weighted majority voting algorithm is to assign appropriate voting weights for classifiers. Many methods have been proposed to assign specific voting weights, such as classification error determines the importance of individual classifier [37], confusion matrix [38] is used to measure different classification ability for different classes.

For our proposed SSupAE, we notice that each individual SupAE and the final SSupAE can both get the posterior probabilities information of samples belong to each class. We should take advantage of those results, so we consider to design an adaptive weighted voting algorithm to fuse those recognition results.

For a sample $\boldsymbol{x}$, if it's posterior probability belongs to each class is even, then it's recognition result is uncertainty, so the sample $\boldsymbol{x}$ is more easy to get wrong when classification, and the classification ability of classifier for sample $\boldsymbol{x}$ is poor; In contrast, if the posterior probability is concentrated on a certain class, the classification result is credible, and the classification ability of classifier for sample $\boldsymbol{x}$ is strong. So posterior probabilities imply the classification certainty and classification ability for samples. Therefore we introduce information entropy [39, 40] to measure classification certainty, and assigns high weight to sample with low entropy, thus more reasonable weights are assigned to different samples adaptively.

For m classes classification problem, suppose that k classifiers are used for ensemble. For sample $\boldsymbol{x}$, the posterior probability matrix of all classifiers denotes as:

$$\boldsymbol{P}(\boldsymbol{x}) = \left\{ \begin{array}{cccc} p_{11}(\boldsymbol{x}) & p_{12}(\boldsymbol{x}) & \cdots & p_{1m}(\boldsymbol{x}) \\ p_{21}(\boldsymbol{x}) & p_{22}(\boldsymbol{x}) & \cdots & p_{km}(\boldsymbol{x}) \\ \cdots & \cdots & & \cdots \\ p_{k1}(\boldsymbol{x}) & p_{k2}(\boldsymbol{x}) & \cdots & p_{km}(\boldsymbol{x}) \end{array} \right\}_{k\times m} \tag{11}$$

Each row is the posterior probabilities of a classifier for sample $\boldsymbol{x}$. The entropy of each classifier for sample $\boldsymbol{x}$ is:

$$H_i(\boldsymbol{x}) = -\sum_{j=1}^{m} p_{ij} \log_2 p_{ij}, \quad i = 1, 2, \ldots, k \tag{12}$$

The $H_i(\boldsymbol{x})$ measures the classification certainty of sample $\boldsymbol{x}$ by ith classifier. If the entropy is lower, the classification result is more certainty. Thus the classification capability of ith classifier for sample $\boldsymbol{x}$ is stronger, the fusion weight of the ith classifier for the sample $\boldsymbol{x}$ should be larger. So in this paper we calculate fusion weight by the formula (13):

$$w_i = \frac{\exp(-H_i(\boldsymbol{x}))}{\sum_{j=1}^{k} \exp(-H_j(\boldsymbol{x}))} \tag{13}$$

For each classifier, after determining the fusion weight of the sample $\boldsymbol{x}$, the ith row of the probability matrix $\boldsymbol{P}(\boldsymbol{x})$ is multiplied by the corresponding weight w_i to get a new probability matrix $\boldsymbol{P}'(\boldsymbol{x})$:

$$\boldsymbol{P}'(\boldsymbol{x}) = \left\{ \begin{array}{cccc} w_1 p_{11}(\boldsymbol{x}) & w_1 p_{12}(\boldsymbol{x}) & \cdots & w_1 p_{1m}(\boldsymbol{x}) \\ w_2 p_{21}(\boldsymbol{x}) & w_2 p_{22}(\boldsymbol{x}) & \cdots & w_2 p_{2m}(\boldsymbol{x}) \\ \cdots & \cdots & & \cdots \\ w_k p_{k1}(\boldsymbol{x}) & w_k p_{k2}(\boldsymbol{x}) & \cdots & w_k p_{km}(\boldsymbol{x}) \end{array} \right\}_{k\times m} \tag{14}$$

Then the posterior probability of sample $\boldsymbol{x}$ belongs to each class is fused by weighted majority vote strategy, namely calculate $\boldsymbol{p}_{vote}$ by getting the column sum of the $\boldsymbol{P}'(\boldsymbol{x})$:

$$\boldsymbol{p}_{vote}(\boldsymbol{x}) = \left[\sum_{i=1}^{k} w_i p_{i1}, \sum_{i=1}^{k} w_i p_{i2}, \ldots \sum_{i=1}^{k} w_i p_{im} \right] \tag{15}$$

The predicted class is the column of $\boldsymbol{p}_{vote}$ with largest probability, namely:

$$Class(\boldsymbol{x}) = \underset{j=1,2,\ldots m}{\arg\max} \sum_{i=1}^{k} w_i p_{ij} \tag{16}$$

With the proposed adaptive weighted majority voting algorithm, we can fuse the classification results of the individual SupAE and the final SSupAE, and the SFSupAE framework is shown in Fig. 4.

The learning procedure of SFSupAE is summarized in Algorithm 1.

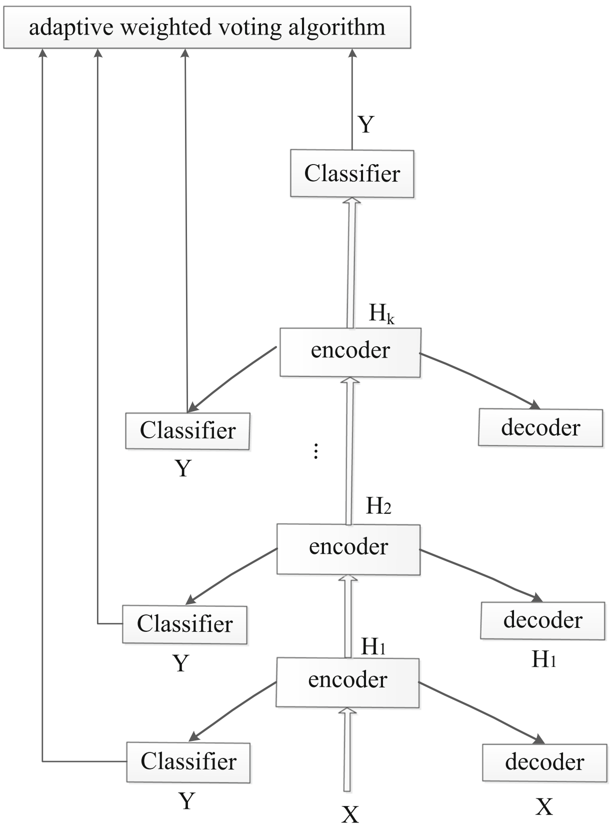

Fig. 4 Stacked fusion supervised auto-encoder framework, X is the input data, Y is label, $\mathbf{H}_i,\ i = 1, 2, \ldots k$ is the hidden representation

Algorithm 1 The training process of the SFSupAE

```
Input:
    Training data set $\{\mathbf{X}, \mathbf{Y}\}$
Output:
    Recognition results.
1. Supervised pre-training stage
For $i$ from 1 to $k$
    1) Train the i-th SupAE with gradient descent based algorithm;
    2) Let the learned hidden representation be the input of next SupAE;
End
2. Supervised fine-tuning stage
    Fine-tune the SSupAE with gradient descent based algorithm;
3. Adaptive weighted majority voting fusion stage
    1) Get the predicted posterior probabilities information of each individual SupAE and the final SSupAE.
    2) Fuse the result with proposed adaptive weighted voting algorithm to get the final recognition results.
```

Table 1 Specification of selected datasets for classification problems

Datasets	#Feature	#Train	#Test	#Categories
Spambase	57	3681	920	2
CarEvaluation	6	1100	628	4
Vowel	13	790	200	11
Vehicle	18	500	346	4
Shuttle	9	29,834	26,936	7
Letter	16	10,500	9500	26
Sat_image	36	4435	2000	7
Segmentation	19	1733	577	7
Page-blocks	10	4385	1100	5
Connect4	42	50,000	17,577	3
DNA	180	1400	1186	3
Musk2	166	3000	3598	2
Protein	357	14,895	6621	3
Isolete	617	6238	1559	26
YaleB	1024	1680	734	38

4 Experiments and Results

4.1 Experimental Setup

To analyze the performance of our proposed method, we perform extensive experiments on 15 UCI [41] benchmark datasets. Details of recognition datasets are listed in Table 1.

We compare our method with several state of-the-art methods, including stacked auto-encoder (SAE), stacked sparse auto-encoder (SSpAE), stacked denoising auto-encoder(SDAE), stacked contractive auto-encoder (SCAE), and stacked Laplacian auto-encoder (SLAE). For SSpAE, the KL-divergence sparsity constant sparsity parameter ρ is set as 0.05, and sparsity penalty coefficient $\beta = 0.1$. SDAE is added with Gaussian noise. For SLAE, it needs to calculate the similarity between pairs of samples.

In [26], the authors used KNN to determinate the neighbors, and Gaussian kernels to calculate the similarity weights, but when the data representation dimensions are high, it is very time-consuming. So in this paper, we treat the same class samples in a mini-batch data as the neighboring samples, and neighboring samples have same similarity weights:

$$w_{ij} = \begin{cases} 1/|N(\boldsymbol{x}_i)| & if\ \boldsymbol{x}_j \in N(\boldsymbol{x}_i) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where denote $\boldsymbol{x}_j \in N(\boldsymbol{x}_i)$, if $\boldsymbol{x}_j$ is neighbor of $\boldsymbol{x}_i$. For SLAE, the regularization parameter λ is searched in range [0.0001, 0.001, 0.01, 0.1, 1]. For our method, parameter λ are also chosen from [0.0001, 0.001, 0.01, 0.1, 1]. We optimize models with Adam [42] algorithm for 100 epochs. The learning rate is set 0.001 for pre-training stage, and 0.0005 for fine-tuning stage, mini-batch size is 256, all internal layers are activated by ReLU nonlinearity function. All these methods have the same network architecture for each dataset. All models are implement with TensorFlow and trained on a single Nvidia GeForce GTX 1080Ti GPU.

Table 2 Test accuracy of SSupAE on Spambase with different parameters

1st layer λ	2nd layer λ				
	0.0001 (%)	0.001 (%)	0.01 (%)	0.1 (%)	1 (%)
0.0001	91.73	91.84	91.35	91.84	91.57
0.001	92.17	**92.35**	92.17	91.73	91.59
0.01	92.25	91.52	92.08	91.98	91.83
0.1	91.32	91.63	91.79	91.19	91.08
1	91.27	91.52	91.49	91.08	91.30

Table 3 Test accuracy of SSupAE on Sat_image with different parameters

1st layer λ	2nd layer λ				
	0.0001 (%)	0.001 (%)	0.01 (%)	0.1 (%)	1 (%)
0.0001	88.95	88.65	89.21	89.15	88.8
0.001	88.45	89.23	**89.64**	89.41	89.0
0.01	89.50	89.45	89.60	89.15	89.35
0.1	89.51	89.23	88.91	89.25	89.35
1	89.15	89.05	88.95	89.15	88.65

Table 4 Test accuracy of SSupAE on Page_blocks with different parameters

1st layer λ	2nd layer λ				
	0.0001 (%)	0.001 (%)	0.01 (%)	0.1 (%)	1 (%)
0.0001	95.88	95.88	95.72	96.06	95.98
0.001	96.28	95.97	96.06	96.31	95.86
0.01	**96.61**	96.25	96.54	96.21	96.06
0.1	96.15	96.28	96.49	96.01	96.46
1	95.89	95.98	95.57	96.17	96.38

4.2 Classification Results on UCI Datasets

In this section, we evaluate the classification performance of our method on 15 UCI datasets. We analyze the influence of regularization parameter λ on our proposed model SSupAE, and carry out comparative analyses about the classification performance and computation time with other relevant stacked auto-encoder methods.

4.2.1 The Influence of the Parameter

We use two hidden layers neural network to train our SSupAE. In pre-training stage, we greedily search the parameter λ of first and second layer in range [0.0001, 0.001, 0.01, 0.1, 1] according to the classification performance. The experimental results of SSupAE on datasets Spambase, Sat_image, Page_blocks, and Musk2 are shown in Tables 2, 3, 4 and 5.

As we can see from the Tables 2, 3, 4 and 5, parameter λ have relatively small impacts on SSupAE model for different datasets, so SSupAE is not very sensitive to parameter, but better parameter can still achieve excellent performance than other parameters. It indicates that we do not need too large classification loss penalty, it maybe because parameter λ is

Table 5 Test accuracy of SSupAE on Musk2 with different parameters

1st layer λ	2nd layer λ				
	0.0001 (%)	0.001 (%)	0.01 (%)	0.1 (%)	1 (%)
0.0001	97.68	98.52	**98.82**	97.58	98.21
0.001	98.61	98.42	98.54	97.98	97.86
0.01	98.75	98.29	98.02	97.64	97.65
0.1	98.68	98.41	98.40	98.06	97.55
1	98.16	98.27	98.35	97.98	97.68

Table 6 Performance of the first SupAE, the second SupAE, SSupAE, and the fusion results of SFSupAE on UCI datasets

Datasets	Networks	First SupAE (%)	Second SupAE (%)	SSupAE (%)	SFSupAE (%)
Spambase	57-100-50-2	90.67	91.22	92.35	92.87
CarEvaluation	6-100-50-4	93.75	96.57	97.71	98.25
Vowel	13-100-50-11	80.17	87.88	95.94	94.16
Vehicle	18-100-50-4	76.78	79.58	81.67	82.24
Shuttle	9-100-50-7	95.78	96.89	99.08	99.56
Letter	16-100-50-26	78.07	77.58	90.32	87.21
Sat_image	36-100-50-7	85.15	85.65	89.64	87.12
Segmentation	19-100-50-7	87.35	91.85	94.13	93.24
Page-blocks	10-100-50-10	95.52	95.98	96.61	97.27
Connect4	42-100-50-2	66.83	67.86	70.12	70.73
DNA	180-200-100-3	91.65	90.39	94.01	93.55
Musk2	166-200-100-2	96.45	97.91	98.82	99.29
Protein	357-200-100-3	66.76	66.35	67.98	68.54
Isolete	617-400-200-26	93.65	94.72	95.89	96.57
YaleB	1024-400-200-38	93.53	94.40	96.49	96.87

used to balance extracting underlying structure of data, as well as providing discriminative features, when classification loss penalty is large (such as $\lambda = 0.1$ or 1), SSupAE may not well capture the underlying explanatory representations of data.

4.2.2 Classification Experiments

With the best parameters, we conduct classification experiments on UCI datasets, and give the test dataset recognition results of the first SupAE, the second SupAE, SSupAE, and the fusion results of SFSupAE in Table 6.

The results indicate that with the additional classification layer, the first SupAE and second SupAE can also achieve very excellent classification performance, this strengthens our claim that combination the reconstruction loss and the classification loss can both extract underlying structures, as well as provide discriminative features. Because all the hidden layer representations are discriminative, after fine-tuning, the performances of SSupAE are significantly improved. When fuse those three classifiers, the recognition results of SFSupAE are improved again on most of the datasets. On dataset Vowel, Letter, Sat_image, Segmentation, and DNA, the fusion results of SFSupAE are lower than that of SSupAE, this is mainly caused by the poor performance of the first SupAE and second SupAE. Therefore, for our method, it may encounter problem that some weak classifiers influence the fusion result, especially for the first SupAE, the low level learned representation may achieve relatively lower accuracy compared with the final SSupAE, in the fusion stage, the fusion result of SFSupAE may lower than that SSupAE. To tackle this problem, we can just select the best results of the SSupAE and the SFSupAE as final recognition results, this process manner is simple but effective.

Our proposed method is very flexible, it has four advantages:

(1) Firstly, we can add the classifier for any auto-encoder to save the training time if the whole network is deep, we do not need to add the classifier for each auto-encoder. If we add classifier on *i*th auto-encoder, we train it with supervised loss and reconstruction loss; Otherwise, we just train it with reconstruction loss.
(2) Secondly, the classifier can be treat as a basic component, and can be added on any variants of auto-encoder to learn discriminate feature, such as SpAE, DAE, and CAE.
(3) Thirdly, the fusion step can also be flexible. We can use many other fusion algorithms. If there exist weak classifiers, we can abandon them, but fusion results of other classifiers.
(4) Meanwhile, it can be used for processing data in a semi-supervised leaning manner. For example, if dataset contains labeled samples and unlabeled samples, we can feed the network with unlabeled samples and learn representation with reconstruction loss, then we feed the network with labeled samples, and use the supervised loss and reconstruction loss to adjust the network.

To conduct some more comprehensive comparisons of our method with SAE, SSpAE, SDAE, SCAE and SLAE, we perform 10 rounds tests for each dataset, and report the average test classification error and standard deviation of different algorithms on each dataset. Table 7 shows the test classification performances on those benchmark datasets, the bold values in the text indicate the best results.

It can be seen that the basic SAE achieve the worst performance, while SSpAE, SCAE and SDAE achieves slightly improved performance on UCI datasets than SAE. SSpAE incorporates the sparse regularization, which can alleviate over-fitting and learn sparse feature, so it performs better than SAE. Usually, datasets are often corrupt with noises in practical application, SCAE and SDAE share similar motivations of learning robust representations, so they also perform better than SAE. SLAE can learn discriminative representation with graph embedding, so it performs better than SAE, SSpAE, SDAE SCAE. From the Table 7, we also noticed that our SFSupAE have slightly improvements compared with SLAE. Our proposed method SFSupAE outperforms all the other algorithms, there are about more than 0.5–1% improvement on most of the datasets. On the one hand, for our proposed SupAE, it combines the reconstruction loss and the classification loss to extract underlying structures of data, as well as provide discriminative features in the pre-training process, incorporation of supervision can facilitate learning of representative yet discriminative features, which result in the improvement of classification performance. On the other hand, we adapt an adaptive weighted voting algorithm to fuse the recognition results of all the classifiers, different levels information can be complementary with each other and our model can get a boost of classification performance, the classification results can also be improved.

Table 7 Performance comparisons on UCI benchmark datasets

Datasets	Networks	SAE (%)	SSpAE (%)	SDAE (%)	SCAE (%)	SLAE (%)	SFSupAE (%)
Spambase	57-100-50-2	91.35 ± 1.12	91.38 ± 0.92	91.58 ± 1.11	91.35 ± 1.08	92.02 ± 0.99	$\mathbf{92.58 \pm 0.87}$
CarEvaluation	6-100-50-4	95.86 ± 0.46	96.02 ± 0.46	95.56 ± 0.46	96.18 ± 0.46	96.86 ± 0.46	$\mathbf{97.98 \pm 0.46}$
Vowel	13-100-50-11	94.42 ± 0.96	94.44 ± 0.86	94.55 ± 0.89	94.95 ± 0.81	95.45 ± 0.79	$\mathbf{95.87 \pm 0.76}$
Vehicle	18-100-50-4	80.51 ± 0.95	80.51 ± 1.05	80.88 ± 1.12	80.76 ± 1.09	81.27 ± 0.97	$\mathbf{81.97 \pm 0.92}$
Shuttle	9-100-50-7	98.38 ± 0.43	98.30 ± 0.45	98.67 ± 0.45	98.52 ± 0.34	98.87 ± 0.39	$\mathbf{99.51 \pm 0.25}$
Letter	16-100-50-26	89.29 ± 0.53	89.58 ± 0.57	89.69 ± 0.49	89.45 ± 0.62	$\mathbf{90.41 \pm 0.59}$	90.25 ± 0.39
Sat_image	36-100-50-7	88.42 ± 0.25	88.51 ± 0.31	88.55 ± 0.17	88.47 ± 0.44	89.09 ± 0.44	$\mathbf{89.41 \pm 0.19}$
Segmentation	19-100-50-7	93.01 ± 0.21	93.39 ± 0.18	93.42 ± 0.25	93.44 ± 0.17	$\mathbf{94.21 \pm 0.18}$	94.02 ± 0.15
Page-blocks	10-100-50-10	95.57 ± 0.36	95.82 ± 0.22	96.36 ± 0.14	96.12 ± 0.24	96.75 ± 0.24	$\mathbf{97.07 \pm 0.24}$
Connect4	42-100-50-2	69.38 ± 0.59	69.51 ± 0.68	69.56 ± 0.63	69.67 ± 0.71	$\mathbf{70.86 \pm 0.61}$	70.61 ± 0.69
DNA	180-200-100-3	92.14 ± 0.48	92.59 ± 0.41	92.34 ± 0.51	92.66 ± 0.32	93.51 ± 0.32	$\mathbf{93.84 \pm 0.29}$
Musk2	166-200-100-2	97.45 ± 0.57	97.61 ± 0.65	97.72 ± 0.68	97.67 ± 0.51	98.17 ± 0.52	$\mathbf{98.89 \pm 0.46}$
Protein	357-200-100-3	66.39 ± 0.29	66.01 ± 0.32	66.41 ± 0.33	66.53 ± 0.26	67.48 ± 0.29	$\mathbf{68.29 \pm 0.25}$
Isolete	617-400-200-26	94.93 ± 0.59	95.37 ± 0.45	95.24 ± 0.42	95.19 ± 0.58	95.79 ± 0.47	$\mathbf{96.27 \pm 0.36}$
YaleB	1024-400-200-38	95.71 ± 0.65	96.27 ± 0.87	96.35 ± 0.66	95.90 ± 0.67	96.61 ± 0.57	$\mathbf{96.69 \pm 0.78}$

Table 8 Computational time comparison

Datasets	Models	First layer pre-training time (s)	Second layer pre-training time (s)	Fine-tuning time (s)	Total training time (s)	Testing time (s)
CarEvaluation	SAE	1.802	2.442	1.894	6.139	<0.001
	SLAE	2.663	4.213	1.986	8.862	<0.001
	SFSupAE	2.223	2.854	1.868	6.945	<0.001
Sat_image	SAE	4.437	4.826	5.169	14.432	0.002
	SLAE	7.266	7.865	5.233	20.364	0.003
	SFSupAE	5.192	5.791	5.382	16.365	0.003
Protein	SAE	32.356	23.343	25.982	81.681	0.030
	SLAE	52.353	41.285	26.853	120.491	0.028
	SFSupAE	35.856	26.758	26.555	91.181	0.033
Isolete	SAE	27.385	13.403	27.934	68.722	0.017
	SLAE	38.893	20.285	28.427	87.605	0.018
	SFSupAE	34.826	16.156	28.152	79.134	0.014
YaleB	SAE	12.036	4.347	9.561	25.944	0.009
	SLAE	15.121	6.992	9.494	31.067	0.012
	SFSupAE	18.787	8.645	9.635	37.067	0.011

4.2.3 Computation Time Analysis

Compared with traditional auto-encoder, our SupAE adds an addition classification layer on the representation layer, so our method may have higher computational complexity in pre-training stage. In this section, we conduct experiments to analyze the computation time. For comparison methods, we choose SAE and SLAE. Because SAE, SSpAE, SDAE and SCAE nearly have the same computational time, so we just choose SAE for comparison. The computation time on dataset CarEvaluation, Sat_image, Isolete, Protein and YaleB are shown in Table 8, we show the pre-training time of the first layer, the second layer, and fine-tuning training time, the sum of those three training time is the total training time, we also show the testing time. For our SFSupAE, it need to perform fusion recognition, the fusion stage is very efficient, we can just ignore the fusion time.

As we can see from Table 8, the fine-tuning time and the testing time are almost same for all methods, this is because that SAE, SLAE, SFSupAE have the same network architecture for each dataset. Our method SFSupAE needs more pre-training time than SAE on all datasets, but on dataset CarEvaluation, Sat_image, and Protein, the increment of pre-training time is very little, this is because the categories of those three datasets is just 4, 7 and 3 respectively, so the nodes of addition classification layer are very small, so just a little more time are needed to pre-training for SFSupAE. While the categories of dataset Isolete and YaleB are 26 and 38, so compared SFSupAE with SAE, the increment of pre-training time is much. Our SupAE model adds an addition classification layer, so the increment of pre-training time for SFSupAE is mainly related to the category of dataset, the more categories there are, the more pre-training time are needed. But for classification problem in practice, the category is small generally, so our method is still computational efficiency when applied in practice.

On small size dataset CarEvaluation, Sat_image, SLAE need a little more pre-training time than SAE and our method, on YaleB, the pre-training time of SLAE is more than that

Table 9 Performance comparisons on digits datasets

Methods	Optigits (%)	Usps (%)	Mnist (%)
SAE	96.67 ± 0.31	96.61 ± 0.28	98.31 ± 0.18
SSpAE	97.09 ± 0.29	96.96 ± 0.26	98.48 ± 0.16
SDAE	96.96 ± 0.37	97.12 ± 0.23	98.58 ± 0.13
SCAE	97.12 ± 0.24	97.18 ± 0.21	98.46 ± 0.17
SLAE	97.31 ± 0.31	97.41 ± 0.26	98.72 ± 0.18
SFSupAE	**97.68 ± 0.29**	**97.65 ± 0.25**	**98.77 ± 0.16**

of SAE, but less than that of our method. On large size datasets Protein and Isolete, SLAE needs much more pre-training time than SAE and our method. This is because that SLAE needs to determine neighboring samples and calculate the similarity between pairs of samples when construct the graph embedding in pre-training stage, which may be time-consuming when the size of datasets is large. Generally, the datasets may have many samples in practice especially in today's era of big data, so SLAE may encounter high computational complexity in practice.

4.3 Application on Digits Recognition

In this section, we apply our method on three handwritten digit recognition datasets, they are Optigits, Usps, and Mnist. Optigits consists of 5620 8×8 greyscale digit images. We divided it into 3823 training samples and 1797 test samples. Each image is flattened to a vector of size $8 \times 8 = 64$. Usps consists of 16×16 greyscale handwritten digit images, it is collected form US postal service. There are 7291 training samples and 2007 test samples. Each image is represented by a vector of size $16 \times 16 = 256$. Mnist is a famous handwritten digit recognition dataset which consists of 55,000 training samples and 10,000 test samples. Each sample is 28×28 greyscale images. We reshape each image to a vector of size $28 \times 28 = 784$.

We use a 64-100-50-10 network for Optigits, 256-200-100-10 network for Usps, 784-400-200-100-10 network for Mnist to trained all the models. The experiments were repeated for 10 times, and the average results were obtained for comparisons. The results are shown in Table 9.

The results show that our method can achieve higher classification accuracy than other algorithms on these three digits datasets. The superior performance of our SFSupAE has been proved once again.

As we stated before, incorporation of supervision can facilitate learning of representative yet discriminative features. So next, we conduct experiments to demonstrate the discriminability of the learned representations for our method. We adapt t-distributed stochastic neighbor embedding method(t-SNE) [43] to produce meaningful representation visualizations on 2-D space. We pre-train the first auto-encoder network 256-200-256 on Usps dataset and apply the t-SNE on the 200 dimension hidden representations and reduce them to 2-dimension data. The visualizations of different approaches are as shown in Fig. 5.

In the figures, each sample is represented by a point with different color denoting different class. As we can see, there exist some digit data points of different classes mixed for SAE, SpAE, DAE, and CAE, those methods only use the unsupervised loss to reconstruct the input raw data, though the learned representations are representative, but not discriminate and optimal for the data separation. It is suggested that points derived from our SupAE and

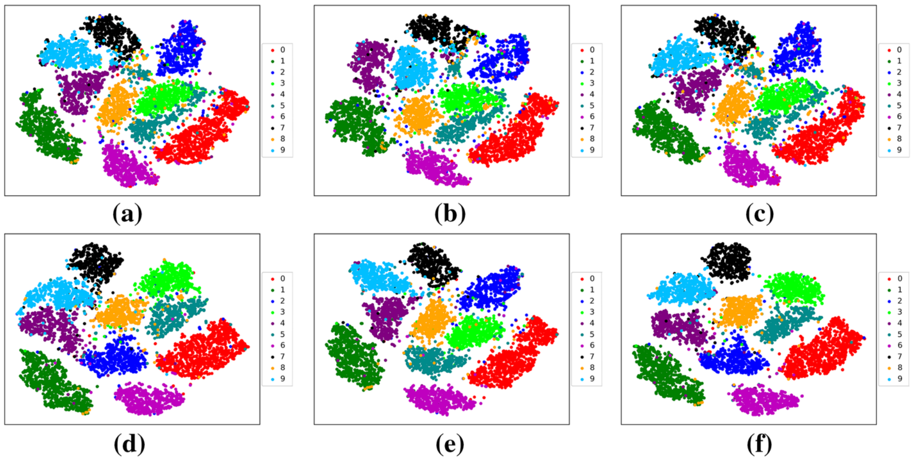

Fig. 5 2-D representations learned by different methods on Usps. **a** AE; **b** SpAE; **c** DAE; **d** CAE; **e** LAE; **f** SupAE

LAE are more distinctive than other methods, for LAE, the representation of digit 8 (marked in orange) have small margins with that of other digits when compared with SupAE, so our SupAE method provides more discriminate representation in the 2D embedding space than the other methods. For our SupAE, incorporating supervision during feature learning encodes class specific characteristics at the feature level, the discriminability of the learned representations attribute to qualitative visualization results in the 2D space. Therefore, combination of the supervised loss and reconstruction loss can better direct feature learning towards representations those are discriminate for the classification tasks.

5 Conclusion

In this paper, we investigate a SupAE to jointly predict targets and reconstruct inputs, our SupAE consist of three parts: an encoder to encode the input, a decoder to reconstruct the input, and a classifier to predict the target. The combination of the reconstruction loss and the classification loss can promote to learn representative and discriminative features, which are useful when building SSupAE for classification tasks. From the point view of classifiers ensemble, we proposed an adaptive weighted voting algorithm based on entropy to fuse the results of each individual SupAE and the final SSupAE, experimental results on several classification datasets show that our method can learn discriminative representations, and improve the classification performance.

Acknowledgements This work is supported by National Natural Science Foundation of China under Grants 61806219, 61876189, 61503407, 61703426, 61273275. This work is also supported by Young Talent fund of University Association for Science and Technology in Shaanxi, China, No. 20190108.

References

1. Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828

2. LeCun Y, Bengio Y, Hinton GE (2015) Deep learning. Nature 521:436–444
3. He KM, Zhang XY, Ren SQ et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
4. Yu J, Tao DC, Wang M, Rui Y (2015) Learning to rank using user clicks and visual features for image retrieval. IEEE Trans Cybern 45(4):767–779
5. Hong CQ, Yu J, Zhang J, Jin XN, Lee KH (2019) Multi-modal face pose estimation with multi-task manifold deep learning. IEEE Trans Ind Inform 15(7):3952–3961
6. Hong CQ, Yu J, Wan J, Tao DC, Wang M (2015) Multimodal deep autoencoder for human pose recovery. IEEE Trans Image Process 24(12):5659–5670
7. Hong CQ, Yu J, Chen XH (2013) Image-based 3D human pose recovery with locality sensitive sparse retrieval. In: Proceedings of the 2013 IEEE international conference on systems, man, and cybernetics, pp 2103–2108
8. Yu J, Tan M, Zhang HY, Tao DC, Rui Y (2019) Hierarchical deep click feature prediction for fine-grained image recognition. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2019.2932058
9. Fayek HM, Margaret L, Lawrence C (2017) Evaluating deep learning architectures for speech emotion recognition. Neural Netw 92:60–68
10. Young T, Hazarika D, Poria S, Cambria E (2017) Recent trends in deep learning based natural language processing. IEEE Comput Intell Mag 13(3):55–75
11. Bengio Y, Lamblin P, Dan P, Larochelle H (2006) Greedy layer-wise training of deep networks. In: Proceedings of the advances in neural information processing systems, vol 19, pp 153–160
12. Le QV, Ngiam I, Coates A, Lahiri A, Prochnow, B, Ng AY (2011) On optimization methods for deep learning. In: Proceedings of the 28th international conference on machine learning, pp 265–272
13. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11(12):3371–3408
14. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive auto-encoders: explicit invariance during feature extraction. In: Proceedings of the 28th international conference on machine learning, pp 833–840
15. Rifai S, Mesnil G, Vincent P, Muller X, Bengio Y, Dauphin Y, Glorot X (2011) Higher order contractive auto-encoder. In: Joint European conference on machine learning and knowledge discovery in databases, pp 645–660
16. Liu WF, Yang XH, Tao DP, Cheng J, Tang YY (2014) Multiview dimension reduction via Hessian multiset canonical correlations. Inf Fusion 41:119–128
17. Ma M, Sun C, Chen X (2018) Deep coupling autoencoder for fault diagnosis with multimodal sensory data. IEEE Trans Ind Inform 14(3):1137–1145
18. Grozdic DT, Jovicic ST (2017) Whispered speech recognition using deep denoising autoencoder and inverse filtering. IEEE Trans Audio Speech Lang Process 25(12):2313–2322
19. Dai Y, Wang G (2018) Analyzing tongue images using a conceptual alignment deep autoencoder. IEEE Access 6(3):1137–1145
20. Park D, Hoshi Y, Kemp CC (2018) A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE Robot Autom Lett 3(3):1544–1551
21. Du F, Zhang JS, Ji NN, Hu JY, Zhang CX (2019) Discriminative representation learning with supervised auto-encoder. Neural Process Lett 49(2):507–520
22. Singh M, Nagpal S, Singh R, Vatsa M (2017) Class representative autoencoder for low resolution multi-spectral gender classification. In: International joint conference on neural networks, pp 1026–1033
23. Gao SH, Zhang YT, Jia K, Lu JW, Zhang YY (2015) Single sample face recognition via learning deep supervised autoencoders. IEEE Trans Inf Forensics Secur 10:2108–2118
24. Sankaran A, Vatsa M, Singh R, Majumdar A (2017) Group sparse autoencoder. Image Vis Comput 60:64–74
25. Liu W, Ma T, Xie Q, Tao D, Cheng J (2017) LMAE: a large margin auto-encoders for classification. Signal Process 141:137–143
26. Jia K, Sun L, Gao S, Song Z, Shi BE (2015) Laplacian auto-encoders: an explicit learning of nonlinear data manifold. Neurocomputing 160:250–260
27. Liao YY, Wang Y, Liu Y (2017) Graph regularized auto-encoders for image representation. IEEE Trans Image Process 26(6):2839
28. Razakarivony S, Jurie F (2014) Discriminative autoencoders for small targets detection. In: 2014 22nd International conference on pattern recognition, pp 3528–3533
29. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747

30. Costa VS, Farias ADS, Bedregal B, Regivan HNS, Canuto AMDP (2018) Combining multiple algorithms in classifier ensembles using generalized mixture functions. Neurocomputing 313:402–414
31. Aburomman AA, Reaz MBI (2017) A survey of intrusion detection systems based on ensemble and hybrid classifiers. Comput Secur 65:135–152
32. Wang XD, Song YF (2018) Uncertainty measure in evidence theory with its applications. Appl Intell 48(7):1672–1688
33. Song YF, Wang XD, Zhu JW, Lei L (2018) Sensor dynamic reliability evaluation based on evidence and intuitionistic fuzzy sets. Appl Intell 48(11):3950–3962
34. Zhao KK, Matsukawa T, Suzuki E (2019) Experimental validation for N-ary error correcting output codes for ensemble learning of deep neural networks. J Intell Inf Syst 52(2):367–392
35. Lei L, Song YF, Luo X (2019) A new re-encoding ECOC using a reject option. Appl Intell. https://doi.org/10.1007/s10489-020-01642-2
36. Lam L, Suen SY (1997) Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Trans Syst Man Cybern Part A Syst Hum 27(5):553–568
37. Lingenfelser F, Wagner J, André E (2011) A systematic discussion of fusion techniques for multi-modal affect recognition tasks. In: Proceedings of the 13th international conference on multimodal interfaces, pp 19–26
38. Catal C, Tufekci S, Pirmit E, Kocabag G (2015) On the use of ensemble of classifiers for accelerometer-based activity recognition. Appl Soft Comput 37:1018–1022
39. Xia MM, Xu ZS (2012) Entropy/cross entropy-based group decision making under intuitionistic fuzzy environment. Inf Fusion 13(1):31–47
40. Song YF, Fu Q, Wang YF, Wang XD (2019) Divergence-based cross entropy and uncertainty measures of Atanassov's intuitionistic fuzzy sets with their application in decision making. Appl Soft Comput. https://doi.org/10.1016/j.asoc.2019.105703
41. Dua D, Graff C (2019) UCI machine learning repository. School of Information and Computer Science, University of California, Irvine, CA. http://archive.ics.uci.edu/ml
42. Kingma, DP, Ba, J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
43. Maaten LVD, Hinton GE (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605