2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Feature Ensemble Learning based on Sparse Autoencoders for Image Classification

Yaping Lu
School of Computer Science and Technology & Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, Jiangsu, China
20134227010@stu.suda.edu.cn

Li Zhang
School of Computer Science and Technology & Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, Jiangsu, China
zhangliml@suda.edu.cn

Bangjun Wang
School of Computer Science and Technology & Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, Jiangsu, China
wangbanjun@suda.edu.cn

Jiwen Yang
School of Computer Science and Technology & Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, Jiangsu, China
jwyang@suda.edu.cn

Abstract—**Deep networks are well known for their powerful function approximations. To train a deep network efficiently, greedy layer-wise pre-training and fine tuning are required. Typically, pre-training, aiming to initialize a deep network, is implemented via unsupervised feature learning, with multiple feature representations generated. However, in general only the last layer representation is to be employed because of its abstraction and compactness being the best with comparisons to the ones of lower layers. To make full use of the representations of all layers, this paper proposes a feature ensemble learning method based on sparse autoencoders for image classification. Specifically, we train three softmax classifiers by using the representations of different layers, instead of one classifier trained by applying the last layer representation. Of the three softmax classifiers, two are obtained by training stacked auto-encoders with fine tuning, and the other one is obtained by directly using a concatenation of two representations. To improve accuracy and stability of a single softmax classifier, the ensemble of multiple classifiers is considered, and some Naive Bayes combination rules are introduced to integrate the three classifiers. Experimental results on the MNIST and COIL datasets are presented, with comparisons to other classification methods.**

Keywords—deep network; feature representation; feature ensemble; autoencoder; softmax; Naive Bayes

I. INTRODUCTION

Deep networks can compactly represent a significantly larger set of highly nonlinear and highly varying functions than shallow networks. Consequently, there has been significantly recent interest in multilayered or deep models for representation of general data [1], such as deep belief networks (DBNs) [2], convolutional restricted Boltzmann machines (RBMs) [3] and hierarchical sparse autoencoders [4], with a particular focus on imagery and audio signals. For these deep models, upper layers are supposed to represent high-level abstractions that explain the input observation, whereas lower layers extract low-level features from it [5]. While the theoretical benefits of deep networks in terms of their compactness and expressive power have been appreciated for many decades, until recently it was not clear how to train such deep networks [6]. Since gradient-based optimization starting from random initialization appears to often get stuck in poor solutions [5]. In 2006, Hinton et al. proposed a fast, greedy layer-wise unsupervised learning algorithm to train DBNs efficiently [7], of which the key point is that, first train one layer at a time in a greedy way, namely pre-training, and then tune the whole network in terms of the final criterion, namely fine tuning. By means of this strategy, a satisfied and stable result is obtained for deep networks. As a way to implement pre-training, unsupervised feature learning plays a very important role in the training of deep networks, and is often used to produce pre-processors and feature extractors for image analysis systems [8]. In general, multiple feature representations would be generated when training a deep network, whereas only the last layer representation is to be employed because of its abstraction and compactness being the best with comparisons to the ones of lower layers.

Sparse autoencoder (SAE) [9], being a kind of model for unsupervised feature learning, is an autoencoder, imposed with a sparseness constraint on the hidden units, that tries to learn an approximation to the identity function so as to output is similar to input. Typically, a representation, instead of raw data, learned via a sparse autoencoder is used as the input to train a classifier. Softmax regression [10] is a common classifier that generalizes logistic regression to classification problems where the class label can take on more than two possible values. A deep network that combines multiple sparse autoencoders and a softmax classifier as a whole network is called stacked autoencoders [11], on which we perform fine tuning so as to obtain a final classifier with higher performance.

It is well known that an ensemble of multiple classifiers is widely considered to be an effective technique for improving accuracy and stability, with comparisons to a single classifier. However, to ensure to get better performance, there are two important issues needed to be taken into account, namely, one being the diversity and accuracy of each classifier, and the other being the combination rules or fusion rules [12]. There are two methods to implement the diversity for ensemble learning [13], one of which is to train multiple classifiers by employing different feature sets [14, 15], just being the method used in this paper. Moreover, the accuracy of an individual classifier is also very important, since the poor classifiers can suppress correct predictions of good classifiers [12]. Then, the final issue is the combination rules of multiple classifiers. Provided the labels are available, a simple voting rule [16] and the Naive Bayes combination methods including MAX rule, MIN rule and AVG rule can be used.

As discussed above, only the representation of last layer, instead of raw data, is used to train a classifier. However, it is true that multiple representations can be obtained when a deep network is trained. Is there a way to utilize all the representations, instead of the last one? Thus, in this paper, as an improvement, we propose a feature ensemble learning method based on sparse autoencoders for image classification. Specifically, we take fully all these representations learned by pre-training two sparse autoencoders into account in the way that utilizing the two representations to train three softmax classifiers, namely, the so-called feature ensemble. In detail, the first and the second softmax classifiers are achieved by respectively exploiting the corresponding two representations learned via sparse autoencoders, of course, with fine tuning used for the classification performance, and as to the third softmax classifier, we directly employ the concatenation of two representations as the input of softmax classification model. Note that although the stacked autoencoder that corresponds to the first representation is a shallow network, we would still perform fine tuning on it, since this can significantly improve the performance of classifier. The reason why we choose the present architecture is that we expect to use the way of feature ensemble to fully utilize all the representations so as to finally improve the classification performance. As for the deep motivations, it is because the representations with different abstraction levels have different advantages for classification, so the feature ensemble method can integrate this advantages so as to improve the performance finally.

After we have finished the training of three softmax classifiers via feature ensemble, next, what we need to consider is that how to integrate these classifiers so as to improve recognition performance efficiently. Here we implement diversity for classifiers by employing different feature sets to train the three softmax classifiers, and in some ways fine tuning and concatenation make the accuracies of the three softmax classifiers be guaranteed. Therefore, combining the three softmax classifiers to predict new data is feasible and promising. Finally, we choose the Naive Bayes combination methods as our combination rule, including MAX rule, MIN rule and AVG rule.

The remainder of the paper is organized as follows. In Section 2, we overview some related works about sparse autoencoder, softmax regression and deep network. The detailed processes of training the three softmax classifiers are discussed in Section 3. Experimental results on the MNIST and COIL datasets are presented in Section 4, with conclusions provided in Section 5.

II. RELATED WORKS

In this section, we briefly introduce some models used in our paper, including sparse autoencoder used to obtain feature representations, softmax regression used to classify images, and deep network composed of the previous two models.

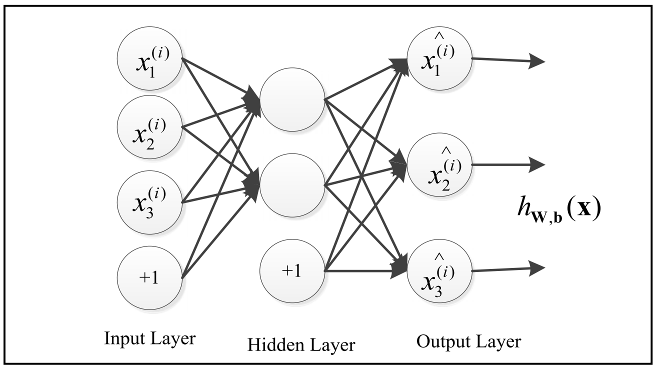

Fig. 1. An autoencoder. The circles labeled "+1" are bias units corresponding to the intercept terms. $h_{\mathbf{W},\mathbf{b}}(\mathbf{x})$ is an activation function, which the autoencoder tries to learn so as to the output $\hat{x}_j^{(i)}$ is similar to the input $x_j^{(i)}$. In general, we call the process from input layer to hidden layer as "encode", from hidden layer to output layer as "decode".

A. Sparse Autoencoder

Sparse autoencoder is an autoencoder, an unsupervised learning method that applies back propagation and sets the target values to be equal to the inputs, imposed with a sparseness constraint on the hidden units. In other words, a sparse autoencoder is trying to learn an approximation to the identity function, so as to output is similar to input.

Suppose the i th sample $\mathbf{x}^{(i)}$ with label $y^{(i)}$ is represented as $(\mathbf{x}^{(i)}, y^{(i)})$ where $\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{1,2,...,k\}$, and d denotes the dimensionality of samples and k is the number of classes. Let $x_j^{(i)}, j \in \{1,2,...,d\}$ represent the j th feature for the sample $\mathbf{x}^{(i)}$. m samples are denoted as $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$. Assume that d is 3, and the number of hidden units is 2, and $\hat{x}_j^{(i)}$ denotes the j th output of output layer for the i th sample, so the illustration of an autoencoder is given in Fig. 1.

For a sparse autoencoder, its cost function $J(\mathbf{W},\mathbf{b})$ requires minimizing

$$J(\mathbf{W},\mathbf{b}) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{\mathbf{W},\mathbf{b}}\left(\mathbf{x}^{(i)}\right) - \mathbf{x}^{(i)}\right\|^2\right)\right] + \frac{\lambda_1}{2}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 + \beta\sum_{j=1}^{s_2} KL\left(\rho \,\|\, \hat{\rho}_j\right) \tag{1}$$

where $h_{\mathbf{W},\mathbf{b}}(\mathbf{x})$ is an activation function, $\mathbf{W}$ and $\mathbf{b}$ denote the weights and biases of network respectively, also being our optimal parameters. The first term tries to minimize the difference between the output and the input. The second term represents weight decay term in order to avoid over-fitting, where λ_1 is a weight decay parameter, and L is the number of autoencoder network layers (in the context of this paper, L is set to be 3), and s_l represents the number of units for the l th layer. $W_{ji}^{(l)}$ denotes the weight between the i th unit of layer l and the j th unit of layer $l+1$, and $b_i^{(l)}$ is the bias associated with unit i in layer $l+1$. The last term is a sparse penalty term where β controls the weight of this term, and ρ is a sparseness parameter, and $\hat{\rho}_j$ denotes the average activation of hidden unit j, namely $\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(i)}\right]$ in which $a_j^{(i)}$ represents the activation of the j th unit of hidden layer for the i th sample, and $KL(\cdot)$ is the Kullback-Leible (KL) divergence given by

$$KL\left(\rho \,\|\, \hat{\rho}_j\right) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}.$$

Typically, ρ is set to be a small value close to 0. In other words, we would like the average activation of each hidden unit j to be close to 0. To satisfy this constraint, the hidden unit's activations must mostly be near 0. To achieve this, we add an extra penalty term (namely the last term) to our cost function that penalizes $\hat{\rho}_j$ deviating significantly from ρ. The reason why we choose KL divergence as our penalty term is that KL divergence is a standard function for measuring how different two different distributions are, and it has the property that $KL\left(\rho \| \hat{\rho}_j\right)=0$ if $\hat{\rho}_j = \rho$, and otherwise it increases monotonically as $\hat{\rho}_j$ diverges from ρ. Thus, minimizing this penalty term has the effect of causing $\hat{\rho}_j$ to be close to ρ.

The optimization problem (1) can be solved by using the back propagation algorithm, and the L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) method is used to update the parameters $\mathbf{W}$ and $\mathbf{b}$.

B. Softmax Regression

Softmax Regression is a classification model generalizing logistic regression to classification problems where the class label y can take on more than two possible values. Again, suppose d and k are 3, then the softmax model is as Fig. 2.

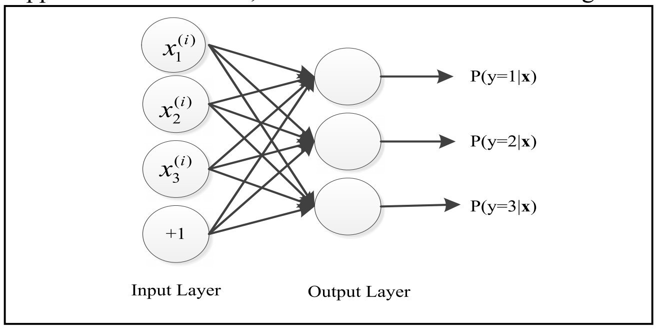

Fig. 2. Softmax model. This model outputs the possibility of each class given a sample $\mathbf{x}^{(i)}$.

For a softmax classifier, the probability output function has the form

$$h_{\boldsymbol\theta}(\mathbf{x}^{(i)}) = \begin{bmatrix} p(y^{(i)}=1 \mid \mathbf{x}^{(i)};\boldsymbol\theta) \\ p(y^{(i)}=2 \mid \mathbf{x}^{(i)};\boldsymbol\theta) \\ \vdots \\ p(y^{(i)}=k \mid \mathbf{x}^{(i)};\boldsymbol\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\boldsymbol\theta_j^T \mathbf{x}^{(i)}}} \begin{bmatrix} e^{\boldsymbol\theta_1^T \mathbf{x}^{(i)}} \\ e^{\boldsymbol\theta_2^T \mathbf{x}^{(i)}} \\ \vdots \\ e^{\boldsymbol\theta_k^T \mathbf{x}^{(i)}} \end{bmatrix}$$

where $\boldsymbol\theta \in \mathbb{R}^{k\times(d+1)}$ is the parameters of softmax model (bias unit considered together), and $\boldsymbol\theta_j$ means the j th row of matrix $\boldsymbol\theta$ which could be found by minimizing the following cost function $J(\boldsymbol\theta)$:

$$J(\boldsymbol\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)}=j\}\log\frac{e^{\boldsymbol\theta_j^T\mathbf{x}^{(i)}}}{\sum_{l=1}^{k} e^{\boldsymbol\theta_l^T\mathbf{x}^{(i)}}}\right] + \frac{\lambda_2}{2}\sum_{i=1}^{k}\sum_{j=1}^{d+1}\theta_{ij}^2 \quad (2)$$

where θ_{ij} denotes the weight between the j th unit of input layer and the i th unit of output layer, and λ_2 is also a weight decay parameter, and $1\{y^{(i)}=j\}$ is the indicator function with

$$1\left\{y^{(i)}=j\right\} = \begin{cases} 1, & if \;\; y^{(i)}=j \\ 0, & if \;\; y^{(i)} \neq j \end{cases}.$$

Once we have the cost function of the softmax model, the similar minimization procedure to SAE model is performed, resulting in the optimal parameters $\boldsymbol\theta$. For more details, refer to [10].

C. Deep Network

Recall that after using the feature representations learned by pre-training a sparse autoencoder as the input to train a softmax model, to enhance the classification performance, a fine tuning process is done on the whole network, typically being a deep network.

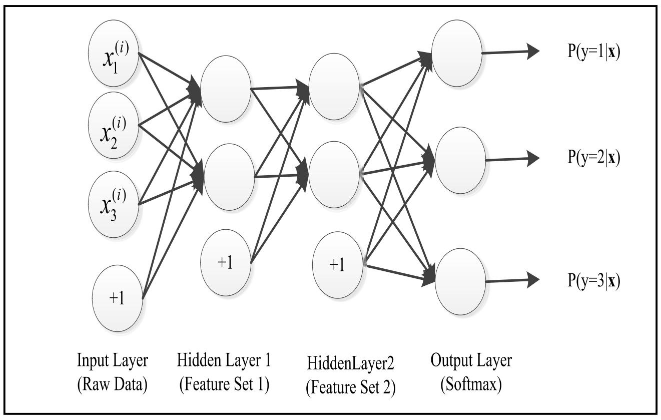

Fig. 3. Stacked Autoencoders. Here, as we can see, the whole deep network contains four layers, specifically, input layer with three units, two hidden layers with two units for each, and output layer with three units. Namely, d is assumed to be 3 as well as k.

Generally, if a neural network contains more than two hidden layers, then we call it a deep network. Trying to train such a deep network with an efficient method is called deep learning. As we know, in 2006, Hinton et al. has proposed a feasible and satisfying strategy to implement the training on DBNs in [7]. Instead of DBNs, here, we would train or fine tune stacked autoencoders network, also being a deep network. We give an illustration about this model in Fig. 3.

If we want to fine tune such a deep network, essentially a multilayered softmax classifier with the cost function of (2), the parameters learned by pre-training, instead of random generation, are selected to initialize the deep model. What's more important, when calculate the partial derivatives of cost function with respect to weights and biases, pay attentions to that the activation function of two hidden layers is different from output layer. After fine tuning is finished, a better and more stable softmax classifier is generated.

III. FEATURE ENSEMBLE LEARNING

Generally, only the feature representation of last layer, instead of raw data, is used to train a classifier. To utilize the representations of all layers, we integrate them by training multiple classifiers and employ some combination rules to make the final decision. Specifically, we discuss the case of two sparse autoencoders. Actually, our method can be extended to more than two.

As discussed above, in this paper, we would train two sparse autoencoders so as to get two representations of the raw data. The first representation is used as the input of the second sparse autoencoder in order to get the second representation. Then, the two representations are concatenated to form the third representation of the raw data. Now, we have all the conditions to train the three classifiers corresponding to the three representations. Note that for the first and second classifiers, we need to perform fine tuning on the whole network (stacked autoencoders) composed of the input layer, feature representation layers and the softmax layer so as to improve the performance of the classifiers. Even if the whole network for the first classifier is a shallow network, we would still do so. For the clear understanding of the whole process, Fig. 4 gives an overview of the process of the feature ensemble. Note that the fine tuning process does not illustrate in the Fig. 4. Next, detailed descriptions for the training of three classifiers are to be presented, respectively.

A. Softmax 2 Classifier

To make the whole system easy to understand, we shall begin with the training of the softmax 2 classifier. According to Fig. 4, we can get two kinds of feature representations corresponding to raw input. Specifically, we first train a sparse autoencoder (SAE 1 in Fig. 4) with HS_1 hidden units on raw input $\mathbf{x} \in \mathbb{R}^d$ in order to obtain the parameters between the input layer and the hidden layer, represented as $\mathbf{W}^{(1,1)} \in \mathbb{R}^{HS_1 \times d}$ and $\mathbf{b}^{(1,1)} \in \mathbb{R}^{HS_1 \times 1}$. Then the raw data $\mathbf{x}$ can be transformed into its first feature representation $\mathbf{f}_1$ in the Feature Set 1, which could be represented by

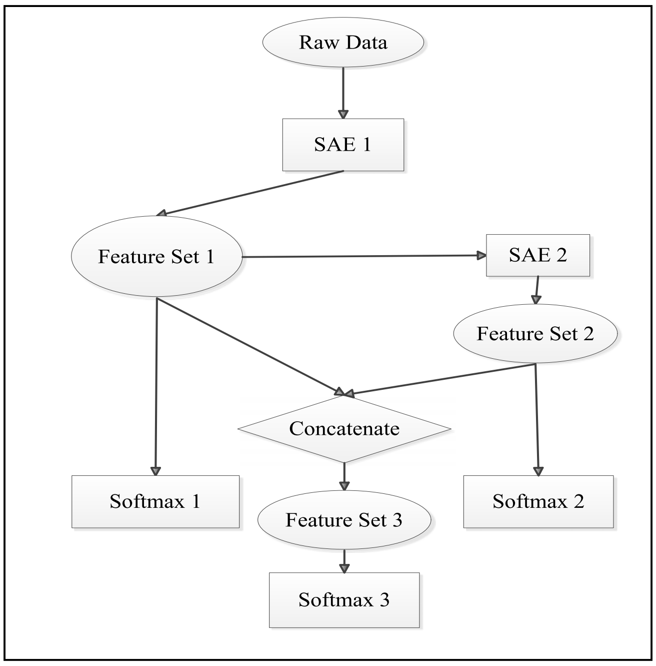

Fig. 4. Overview of Feature Ensemble, where rectangle represents the model such as sparse autoencoder or softmax, and ellipse denotes the input or output of a model, and the rhombus means some operation.

$$\mathbf{f}_1 = h_{\mathbf{W}^{(1,1)},\mathbf{b}^{(1,1)}}(\mathbf{W}^{(1,1)}\mathbf{x} + \mathbf{b}^{(1,1)}) .$$

Next, again, we train the other sparse autoencoder (SAE 2 in Fig. 4) with HS_2 hidden units on the Feature Set 1 so as to get the parameters between hidden layers, represented as $\mathbf{W}^{(2,1)} \in \mathbb{R}^{HS_2 \times HS_1}$ and $\mathbf{b}^{(2,1)} \in \mathbb{R}^{HS_2 \times 1}$. Similarly, $\mathbf{x}$ can be transformed into its second feature representation $\mathbf{f}_2$ in the Feature Set 2, which could be represented by

$$\mathbf{f}_2 = h_{\mathbf{W}^{(2,1)},\mathbf{b}^{(2,1)}}(\mathbf{W}^{(2,1)}\mathbf{f}_1 + \mathbf{b}^{(2,1)}) .$$

In the end, we would train a classification model Softmax 1 (see Fig. 4) on the input Feature Set 2 to obtain the optimal parameters $\boldsymbol{\theta}$ in (2), denoted as $\boldsymbol{\theta}_2 \in \mathbb{R}^{k \times (HS_2+1)}$.

Now, by the pre-trainings above, we have got the first classifier Softmax 2 based on the Feature Set 2. However, its classification performance is unsatisfying for the way of greedy layer-wise pre-training, so that we need to operate fine tuning on the whole network consisting of the input layer of SAE 1, the hidden layer of SAE 1, the hidden layer of SAE 2 and the output layer of Softmax 1 in Fig. 3. Before fine tuning, the initial parameters between two layers of the whole network from left to right are set to be $\{\mathbf{W}^{(1,1)},\mathbf{b}^{(1,1)}\}$, $\{\mathbf{W}^{(2,1)},\mathbf{b}^{(2,1)}\}$ and and $\boldsymbol{\theta}_2$. Then, with the raw data as input, the back propagation algorithm is employed, based on which the final optimal parameters about the whole network or deep network are calculated. Finally, we obtain the first classifier Softmax 2. Recall that the activation functions for two hidden layers and output layer are different so that we should pay attentions to the process of back propagation.

B. Softmax 1 Classifier

After the first classifier Softmax 2 has been achieved, we have got what it takes to train the second one. In other words, on the basis of the first classifier, all the parameters employed to train the second one are available. Specifically, the Feature Set 1 is used as the input to train Softmax 1 classifier (see Fig. 4), resulting in the model optimal parameters $\boldsymbol{\theta}$ in (2), denoted as $\boldsymbol{\theta}_1 \in \mathbb{R}^{k\times(HS_1+1)}$, obtained.

So far, the second classifier Softmax 1 is generated, corresponding to the Feature Set 1. Nevertheless, aiming for the better classification performance, we also perform the fine tuning on the whole network composed of the input layer of SAE 1, the hidden layer of SAE 1 and the output layer of Softmax 1, just like the model in Fig. 3 without the second hidden layer. Before fine tuning, the initial parameters between the input layer and the hidden layer, and the ones between the hidden layer and the output layer are set to be $\{\mathbf{W}^{(1,1)},\mathbf{b}^{(1,1)}\}$ and $\boldsymbol{\theta}_1$ respectively. After fine tuning, the second classifier Softmax 1 can be achieved finally. Also, we must be careful to the activation functions of the hidden layer and output layer when executing the back propagation.

Here, as we can see, the whole network fine-tuned with three layers is a shallow network instead of a deep network. Although the fine tuning is not employed to a shallow network generally, we still do it just because this is able to improve the accuracy and stability of the second classifier, so as to contribute to the performance of the following ensemble of multiple classifiers [12].

C. Softmax 3 Classifier

The two classifiers Softmax 1 and Softmax 2 with fine tuning above are trained on the Feature Set 1 and Feature Set 2, respectively. After taking into full account the representations, namely Feature Set 1 and Feature Set 2, we use the concatenation of these two representations as the input of our third classifier Softmax 3 that is to be trained (see, Fig.4). Specifically, Feature Set 1 and Feature Set 2 are concatenated into one column, denoted as Feature Set 3. Namely, the corresponding feature of $\mathbf{x}$ can be represented as $\left[\mathbf{f}_1^T,\mathbf{f}_2^T\right]^T$.

Then, the Softmax 3 is trained by using the softmax model like Fig. 2 with the representation Feature Set 3 being the input. After minimizing the cost function, we obtain the last classifier, Softmax 3 with parameters $\boldsymbol{\theta}_3 \in \mathbb{R}^{k\times(HS_1+HS_2+1)}$.

D. Naive Bayes Combination Methods for Voting

After all the work above done, the final thing is to give the voting rule for the ensemble of the three softmax classifiers. Here, the Naive Bayes combination methods are considered, in which assume that individual classifiers are mutually independent; hence the name "naive" [16, 17]. Three naive Bayes methods, namely MAX rule, MIN rule and AVG rule, are given below.

For a new sample $\mathbf{x}$ that is to be tested, when its label y takes on the different possible values $j\in\{1,2,...,k\}$, we can get the corresponding predicted probabilities for the softmax classifier $n\in\{1,2,3\}$, here denoted as $\mathbf{P}_{nj}(\mathbf{x})$ which can be expressed as

$$\mathbf{P}_{nj}(\mathbf{x})=\begin{cases} h_{\boldsymbol{\theta}_1}(\mathbf{f}_1), & if\ n=1\\ h_{\boldsymbol{\theta}_2}(\mathbf{f}_2), & if\ n=2\\ h_{\boldsymbol{\theta}_3}\left(\left[\mathbf{f}_1^T,\mathbf{f}_2^T\right]^T\right), & if\ n=3\end{cases}.$$

The final value of label y is determined by the following rules.

1) MAX Rule

Given a new sample $\mathbf{x}$, the MAX rule is to get its label y by

$$y=\arg\max_{j\in\{1,2,...,k\}}\max_{n\in\{1,2,3\}}\mathbf{P}_{nj}(\mathbf{x}). \tag{3}$$

2) MIN Rule

Given a new sample $\mathbf{x}$, the MIN rule is to get its label y by

$$y=\arg\max_{j\in\{1,2,...,k\}}\min_{n\in\{1,2,3\}}\mathbf{P}_{nj}(\mathbf{x}). \tag{4}$$

3) AVG Rule

Given a new sample $\mathbf{x}$, the AVG rule is to get its label y by

$$y=\arg\max_{j\in\{1,2,...,k\}}\sum_{n=1}^{N}\mathbf{P}_{nj}(\mathbf{x})/N \tag{5}$$

where $N=3$ here.

IV. Experimental Results

In the examples below, we take into account two datasets, MNIST dataset [18] and COIL dataset [19], to verify our strategy proposed here. Moreover, K-nearest neighbor (KNN) method is used as the comparison under the same datasets.

All numerical experiments are performed on the personal computer with a 3.40GHz Intel(R) Core(TM) i3-3240 CPU and 3.15G bytes of memory. This computer runs on Windows 7, with MATLAB 8.1.0.

A. Parameter Settings

As we can see, the sparse autoencoder construction in (1) and the softmax classifier construction in (2) may appear relatively complex, while the number of parameters that need to be set is not particularly large, and they are initialized so as to allow the sparse autoencoder to get good filters. Specifically, for the two sparse autoencoders $\lambda_1=3e-3$, $\beta=3$, $\rho=0.1$, $L=3$, $HS_1=100$, $HS_2=100$ and the activation function is set to be sigmoid function, namely $h_{\mathbf{W},\mathbf{b}}(\mathbf{x})=1/(1+e^{-(\mathbf{wx}+\mathbf{b})})$ with $\mathbf{W}$ and $\mathbf{b}$ initialized randomly for the first time, and for the softmax classifier $\lambda_2=1e-4$ and $\boldsymbol{\theta}$ is also initialized randomly for the first time. The same parameters are used in all examples and in all layers of the "deep" network, essentially a deep softmax classifier. The values of m, k and d depend on the specific dataset, and these parameters are specified for the specific examples. In all experiments, we consider the L-BFGS as optimization algorithm with 400 iterations for sparse auto-encoders and deep networks, and 100 iterations for softmax classifiers.

B. Experiments on the MNIST Dataset

Consider the widely studied MNIST data, which has a total of 60,000 training images and 10,000 testing images, each 28×28, for handwritten digits 0 through 9 (this means $k = 10$). In Fig. 5, we randomly give 100 images of the MNIST dataset.

In our experiments, 23,000 sequential images from the beginning of the whole training set are selected as our unlabeled training set to pre-train our two sparse autoencoders so as to get the optimal parameters $\{\mathbf{W}^{(1,1)}, \mathbf{b}^{(1,1)}\}$ and $\{\mathbf{W}^{(2,1)}, \mathbf{b}^{(2,1)}\}$ corresponding to the two representations, and the last 20,000 images of the whole training set are selected as training set to train our softmax classifiers. Note that, before training the classifier, the training set with labels needs to be transformed to another corresponding representation via the parameters $\{\mathbf{W}^{(1,1)}, \mathbf{b}^{(1,1)}\}$ and $\{\mathbf{W}^{(2,1)}, \mathbf{b}^{(2,1)}\}$ learned through the unlabeled data.

After the three softmax classifiers have been trained with the unlabeled training set and labeled training set above, we use the whole 10,000 testing images of MNIST data to test them and report the recognition rates for these three classifiers. In our method, by the Naive Bayes combination methods discussed above, we can obtain the corresponding ensemble results. For comparison, KNN classifier is performed by using the same training set with labels and the same testing set. We vary the parameter K from 1 to 10, and get the best recognition rate of KNN classifier with K=4. All experimental results are listed in TABLE I.

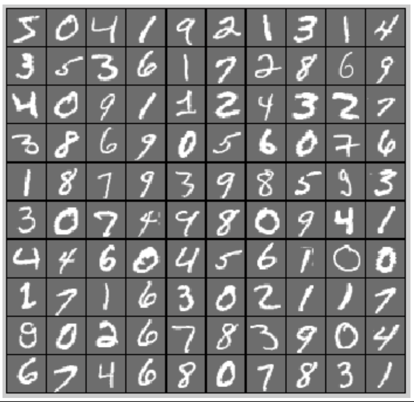
Fig. 5. Images from MNIST dataset.

As we can see in TABLE I, our methods get better recognition rates with comparison to other 4 classifiers. Moreover, when we utilize the Naive Bayes rules to make decision, the AVG rule performs more efficiently than the MIN rule and the MAX rule. For a new sample, if Softmax 2 misclassifies it, while the other 2 classifiers get the expected prediction, then the AVG method has a good chance of correcting the output of Softmax 2, and finally improves the performance.

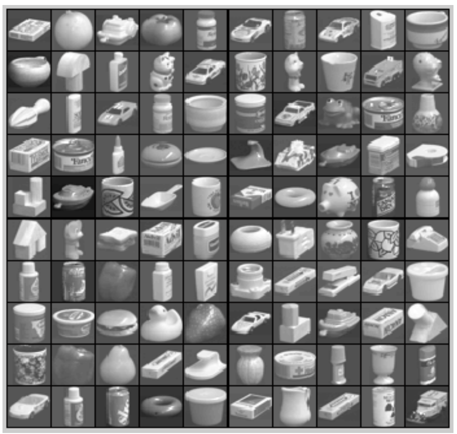
Fig. 6. Images from COIL dataset.

C. Experiments on the COIL Dataset

The COIL dataset is considered next. It has a total of 7,200 examples, each 32×32, with 100 classes (this means $k = 100$), 72 examples for each class. The original images of COIL dataset are color ones, and here we change them into grayscale ones. Fig. 6 shows 100 images from the COIL dataset, one for each class.

In our experiments, since the limitation of the number of examples, we select 6,000 samples, namely 60 examples for each class, as our training set to pre-train the two sparse autoencoders and train the three classifiers. In other words, being different from above method, here the unlabeled training set and the labeled training set are the same. The remainder 1,200 examples, namely 12 examples for each class, are used as the test set. Similar to the MNIST dataset, we report the results of KNN, three softmax classifiers and their ensemble in TABLE II. From TABLE II, we have the same conclusion as TABLE I. The feature ensemble methods achieve the best classification performance.

For the two tables above, since the representations learned by SAEs have different abstraction levels, the diversity for the three softmax classifiers is guaranteed. As a result, the Naive Bayes methods can be employed to improve the recognition performance. Specifically, Feature Set 2 in Fig. 4 is more abstract than Feature Set 1 so that Softmax 2 being a deep network should work better than Softmax 1 that is a shallow network, but we have the opposite result in our experiments. This is because the number of unlabeled training set for SAE is not enough. Since a deep network with fine tuning is more powerful in the function approximation than a softmax regression model, Softmax 2 performs better than Softmax 3. Besides, although Feature Set 1 is related to Feature Set 2, we still consider to use their concatenation to train our third classifier because compared with the classifier that trained directly with only one feature set as input, Softmax 3 indeed has a better recognition rate so as to improve the whole performance of feature ensemble.

TABLE I. CLASSIFICATION PERFORMANCE FOR THE MNIST DATA SET WITH COMPARISONS TO OTHER METHODS

Classifier	**KNN(K=4)**	**Softmax 1**	**Softmax 2**	**Softmax 3**	**Our Method**		
					MAX	***MIN***	***AVG***
Accuracy (%)	95.92	96.07	95.87	95.22	96.53	96.50	96.70

TABLE II. CLASSIFICATION PERFORMANCE FOR THE COIL DATA SET WITH COMPARISONS TO OTHER METHODS

Classifier	**KNN(K=4)**	**Softmax 1**	**Softmax 2**	**Softmax 3**	**Our Method**		
					MAX	***MIN***	***AVG***
Accuracy (%)	84.00	89.41	86.75	86.75	90.50	91.42	91.67

V. CONCLUSION

This paper proposes a feature ensemble method based on sparse autoencoders for image classification. In our method, there are three softmax classifiers, each of which is to train a different feature set. Finally, some Naive Bayes combination rules can be used for ensemble the outputs of these classifiers. From the results of the experiments on the MNIST dataset and COIL dataset, we can see that feature ensemble learning based on sparse autoencoder has better classification performance, with comparisons to KNN classifier and the other three softmax classifiers.

Although, here, we need to train multiple classifiers with the cost of time, fortunately, this can be tolerated for neural network (here, sparse autoencoder, softmax model and deep network) since the prediction for a new sample would be fast. For the future work, we may try to improve the representation of Feature Set 2 to the raw data so as to enhance the performance of Softmax 3, since the Feature Set 2 is obtained by greedy layer-wise pre-training and can't reconstruct to the raw data well. Additionally, the proposed algorithm is only compared with individual classifiers, but, for the future work, it is indeed promising to compare the results with the ones of using other classifiers with the same ensemble strategy.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61373093 and 61033013, by the Natural Science Foundation of Jiangsu Province of China under Grant Nos. BK2011284 and BK201222725, by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant No.13KJA520001, and by the Qing Lan Project.

REFERENCES

[1] B. Chen, G. Polatkan, G. Sapiro, D. Blei, D. Dunson, and L. Carin, "Deep learning with hierarchical convolutional factor analysis," IEEE Tran. Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1887-1901, 2013.

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

[3] M.R.M. Norouzi and G. Mori, "Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning," Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[4] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," Proc. Int'l Conf. Machine Learning, 2008.

[5] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise rraining of deep networks," NIPS, 2007.

[6] Y. Bengio, "Learning deep architectures for AI," Foundations and Trends in Machine Learning 21(6), pp. 1601-1621, 2009.

[7] G. Hinton, S. Osindero, and Y. The, "A fast learning algorithm for deep belief nets," Neural Computation, 18, pp. 1527-1554, 2006.

[8] M. A. Ranzato, C. Poultney, S. Chopra, and Y LeCun, "Efficient learning of sparse representations with an energy-based model," NIPS, 2006.

[9] A. Ng, "Sparse autoencoder," CS294A Lecture Notes for Stanford University, 2011.

[10] A. Ng, "Generalized linear models," CS229 Lecture Notes for Stanford University, part 3, 2003.

[11] A. Ng, J. Ngiam, C.Y. Foo, Y. Mai, and C. Suen, "UFLDL tutorial: building deep networks for classification," an online tutorial, 2013.

[12] L. Zhang and W.D. Zhou, "Sparse ensembles using weighted combination methods based on linear programming," Pattern Recongnition, 44, pp. 97-106, 2011.

[13] E.K. Tang, P.N. Suganthan, and X. Yao, "An analysis of diversity measures, Machine Learning," Machine Learning, 65, pp. 247-271, 2006.

[14] T.K. Ho, J.J. Hull, and S.N. Srihari, "Decision combination in multiple classifier systems," IEEE Transactions on Pattern Analysis and Machine Intelligence, 16, pp. 66-75, 1994.

[15] S.D. Bay, "Combining nearest neighbor classifiers through multiple feature subsets," Proceeding of the 15th International Conference on Machine Learning, pp. 37-45, 1998.

[16] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas, "On combing classifiers," IEEE Transactions on Pattern Analysis and Machine Intelligence, 20, pp. 226-239, 1998.

[17] L.I. Kuncheva, "Combining pattern classifier: method and algorithms," Wiley, Hoboken, NJ, 2004.

[18] "http://yann.lecun.com/exdb/mnist/".

[19] S.A. Nene, S.K. Nayar, and H. Murase, "Columbia object image library(COIL-100)," Techniqual Report, CUCS-006-96, Department of Comp. Science, Columbia University, 1996.