Knowledge-Based Systems 89 (2015) 541–552

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A novel decorrelated neural network ensemble algorithm for face recognition[☆]

Kankan Dai, Jianwei Zhao, Feilong Cao[*]

Department of Applied Mathematics, College of Sciences, China Jiliang University, Hangzhou 310018, Zhejiang Province, PR China

ARTICLE INFO

Article history:
Received 28 December 2014
Received in revised form 28 August 2015
Accepted 1 September 2015
Available online 11 September 2015

Keywords:
Feed-forward neural networks
Ensemble learning
Randomness
Face recognition

ABSTRACT

The main purpose of negative correlation learning (NCL) is to produce ensembles with sound generalization capability through controlling the disagreement among base learners' outputs. This paper uses neural networks with random weights (NNRWs) to implement such learning scheme in the study of face recognition. Particularly, two-dimensional (2D) feed-forward neural networks (2D-FNNs) with random weights (2D-NNRWs) are employed as base components, and which are incorporated with the NCL strategy for building neural network ensembles, where the basis functions of the base networks are generated randomly and the free parameters of the 2D-FNNs can be determined by solving a linear equation system. Also, an analytical solution is derived for these parameters. To examine the merits of the proposed algorithm, a series of comparative experiments are performed. The experimental results indicate that the proposed approach outperforms existing approaches.

1. Introduction

Ensemble learning framework has received considerable attention and resulted in many novel machine learning techniques [1–4]. As an efficient ensemble learning approach, negative correlation learning (NCL) [5], which amended the cost function of each base learner with a penalty term that weakens the relationship with other individuals and encourages the diversity among ensemble components [6], has also generated considerable interest. This concept was extended by Liu and Yao [7], and an improved approach, which can automatically determine an optimal size of ensemble by thoroughly exploring the ensemble hypotheses space using NCL, was proposed by Liu et al. [8]. In [9], a regularized version of NCL was proposed to reduce over-fitting risk for noisy data.

In addition to diversity concerns in ensemble learning, the method of generating the component models is crucial. Feed-forward neural networks (FNNs) with back-propagation (BP) learning algorithm are mostly employed for this purpose [6]. Unfortunately, the gradient based learning algorithms suffer from slow convergence, local minima problem, and very poor sensitivity to the learning rate setting. To overcome these difficulties, neural networks with random weights (NNRWs) were proposed [10], where the weights and basis in the hidden layer could be randomly assigned and have no need to be tuned. The output weights can be solved by using least squares methods, and represented by the generalized pseudo-inverse of output matrix [11]. This flat-net architecture universally approximates any continuous function and dramatically reduces the training time [12,13]. Recently, a local learning algorithm for random weights networks [14], and a new learning approach that used NNRW as ensemble components and employed NCL strategy to learn the free weights (termed as DNNE NNRW) [15] were proposed.

In [16], a network model based on two-dimensional FNN (2D-FNN) has been proposed to classify matrix data directly. Compared with traditional vector based networks, 2D-FNN employs left and right projecting vectors to regress matrix inputs, which not only reduces the complexity of the networks but also helps to preserve the 2D feature structure. Lu et al. [16] used randomization to train the network, i.e., taking random sets of the left and right projecting vectors and the hidden biases, and then determining the output weights by using the least squares method. As an extension to NNRW, we call this 2D-NNRW algorithm. The resultant method is superior to NNRW for face recognition.

Randomness in the NNRW algorithm can be understood from the function approximation with the Monte-Carlo (MC) method. It has been shown in [12] that any continuous function defined on a compact set can be represented by a limited integral of a multivariate continuous function over parameter space. Using the MC

☆ Supported by the National Natural Science Foundation of China (Nos. 61571410, 91330118, and 61272023) and Zhejiang Provincial Natural Science Foundation of China (No. LY14A010027)

* Corresponding author.
E-mail address: feilongcao@gmail.com (F. Cao).

http://dx.doi.org/10.1016/j.knosys.2015.09.002
0950-7051/© 2015 Elsevier B.V. All rights reserved.

method, this multiple integral can be approximated by drawing random samples of the parameter vector from a uniform distribution defined over the limited integral domain. Although using the random weights algorithm can simplify the learning steps of FNN and 2D-FNN, some issues require careful attention in practice [17]: (1) The number of hidden nodes should be sufficiently large and supervised initialization is required to model and compensate for the system uncertainties. (2) There is over-fitting caused by many additional hidden nodes in the NNRW method as a result of the MC approximations. (3) There is predictive instability caused by random assignment of nonlinear weights. However, these issues can be relaxed when a collection of NNRWs or 2D-NNRWs are combined [12,15].

This paper proposes a new ensemble approach for face recognition. The 2D-FNNs are used as base components, and the NCL strategy is combined with the 2D-NNRW algorithm to build ensembles (DNNE 2D-NNRW). The linear weights of base 2D-FNNs can be analytically calculated by solving a system of linear equations. A minimum norm least squares solution is derived and formulated in a matrix form. The proposed method applied to test datasets shows promising, and does have better performance compared with existing methods.

The rest of this paper is organized as follows. FNN models and corresponding learning algorithms are reviewed in Section 2. Our proposed method is detailed description in Section 3. An evaluation of the performance of our proposed algorithm applied to a series of face datasets is presented in Section 4. The conclusion and further study are presented in Section 5.

2. Background

2.1. Review for NNRW

Generally, an FNN with a single hidden layer (SLFN) can be described as

$$f(x)=\sum_{j=1}^{L}\beta_j g(w_j^\top x+b_j)+\alpha, \tag{1}$$

where x is the input pattern vector; $w_j=[w_{j1},w_{j2},\ldots,w_{jd}]^\top$ and b_j are hidden layer weights and biases, respectively; and $\beta_j=[\beta_{j1},\beta_{j2},\ldots,\beta_{jo}]^\top$ and $\alpha=[\alpha_1,\alpha_2,\ldots,\alpha_o]^\top$ are the output layer weights and biases, respectively.

From conventional FNN theory, all the parameters (w_k,b_k,β_k,α) in SLFN require free adjustment, and need to be trained and tuned properly using training samples. Many optimization methods, such as gradient descent [18], conjugate gradient (CG) [19], Levenberg–Marquardt [20], Broyden–Fletcher–Golfarb–Shanno (BFGS) [21], and so forth, can be used to solve these parameters. However, they generally suffer from slow convergence, local minima problem, and very poor sensitivity to the learning rate setting.

NNRW has been proposed to overcome these difficulties [10]. Even when the hidden layer weights and biases are randomly assigned in an appropriate range, SLFNs are still universal approximators [12], and the NNRW's approximation error converges to zero with order $\left(C/\sqrt{L}\right)$ [12,13], where L is the number of hidden nodes and C is a constant. Where weights and biases are randomly assigned, i.e., the hidden layer weights, w_k, and biases, b_k, are defined on a probabilistic space, $S_p(\Omega,P)$, where (Ω,P) should be determined in the learning stage, and generally P is a uniform distribution, then it just needs to tune the linear weights, (β_k,α), which can be expressed as

$$\tilde{\beta}_0=\arg\min_{\tilde{\beta}}\sum_{p=1}^{N}\left\|\sum_{j=1}^{L}\beta_j g(w_j^\top x_p+b_j)+\alpha-t_p\right\|_2^2,$$

where $\tilde{\beta}=[\beta_1,\beta_2,\ldots,\beta_L,\alpha]^\top$ need to be learned using the training samples, $(x_p,t_p)(p=1,2,\ldots,N)$. Rewriting this in matrix form,

$$\tilde{\beta}_0=\arg\min_{\tilde{\beta}}\|\mathbf{H}\tilde{\beta}-\mathbf{T}\|_F^2,$$

where

$$\mathbf{H}=\begin{bmatrix} g(w_1^\top x^{(1)}+b_1) & \ldots & g(w_L^\top x^{(1)}+b_L) & 1\\ \vdots & \ddots & \vdots & \vdots\\ g(w_1^\top x^{(N)}+b_1) & \ldots & g(w_L^\top x^{(N)}+b_L) & 1\end{bmatrix}_{N\times(L+1)}$$

is the hidden output matrix; $\mathbf{T}=[t_1,t_2,\ldots,t_N]^\top$ are the desired output targets; and $\|\cdot\|_F$ denotes the Frobenius norm of the matrix. The optimal linear weights are

$$\tilde{\beta}_0=\mathbf{H}^\dagger\mathbf{T}, \tag{2}$$

where $\mathbf{H}^\dagger$ is the generalized pseudo-inverse of $\mathbf{H}$. We call such an approach NNRW [10].

For a simple test, we may use NNRW to approximate the function

$$y=e^x-x\sin(x)\cos(x).$$

We randomly generate 50 points $(x\in(-7,2))$ to represent the function (see the left pane of Fig. 1). The network hidden layer weights and bias are randomly generated from $(-\Omega,\Omega)$, and the linear weights are determined from Eq. (2). The right pane of Fig. 1 shows the training mean square error (MSE) in terms of Ω and different base network sizes. The optimal Ω changes as the number of hidden nodes of the network increases. Further tests show that for the same dataset the optimal Ω is not unique for different preprocessing. This is due to MC approach [12] and the characteristics of NNRW.

2.2. Review for 2D-NNRW

While FNNs are universal approximators [22–24], which means that the networks can fit any training data, it does not mean that the networks will generalize to make predictions on new data. This can be explicitly formulated in the bias-variance decomposition [25].

Traditional FNNs are vector-based, so when they are used to classify matrix data, such as images, the matrices must be decomposed into vectors. This unavoidably destroys the structural correlation among elements and may subsequently influence recognition performance. To classify the matrix data directly and preserve the structural information of the 2D-input features, Lu et al. [16] designed a new network model, extended 2D-FNN, which is described as

$$f(X)=\sum_{j=1}^{L}\beta_j g(\mathbf{u}_j^\top X\mathbf{v}_j+b_j)+\alpha,$$

where $\mathbf{u}_j=[u_{j1},u_{j2},\ldots,u_{jm}]^\top$ and $\mathbf{v}_j=[v_{1j},v_{2j},\ldots,v_{nj}]^\top$ are the left and right projection vectors, respectively; X is the input pattern matrix; and b_j,β_j, and α are the hidden layer biases, output layer weights, and biases, respectively. This concept was inspired, to some extent, by many popular 2D methods, such as 2D principle component analysis (2D-PCA) [26,27], 2D linear discriminant analysis (2D-LDA) [28,29], and multiple rank regression (MRR) [30].

The 2D-FNN is more suitable for image recognition because of the implicit sharing weights when the left and right projecting vectors, $\mathbf{u}^\top$ and $\mathbf{v}$, are used to regress the matrix input X, i.e., the elements of the ith row of X are all multiplied by the ith weight of $\mathbf{u}$, so the elements in the same row of X are connected by their corresponding shared weight of the left projecting vector. Similarly, the elements in the same column are connected by their

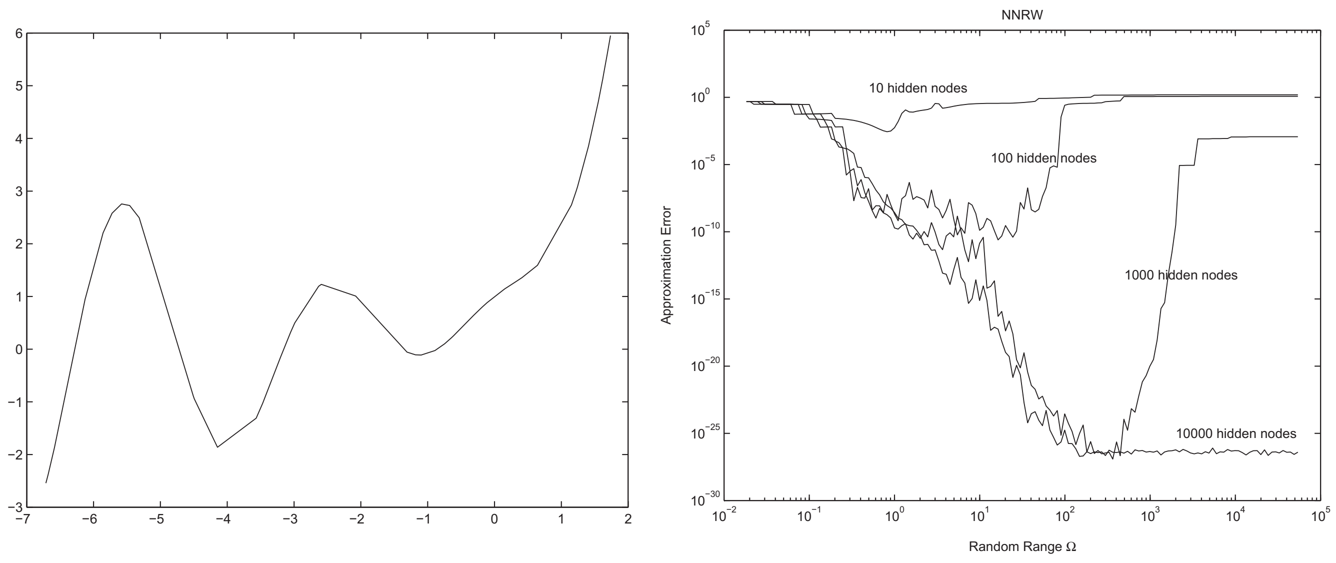

Fig. 1. The left pane shows the shape of the testing function, where horizontal axis is for *x*, and vertical axis is for *y*. The right pant is the approximating error for **different number of hidden nodes** and for **different random ranges** using NNRW.

corresponding sharing weight of the right projecting vector. Such processes preserve the structural information of the original matrix input, such as images.

The network model not only affects the generalization performance, but the learning algorithm also plays an important role in pattern recognition. In [31], we implemented a gradient based full network learning algorithm for the 2D-FNN (2D-BP), and showed that the 2D-FNN is superior to FNN model for image recognition. However, this kind of learning algorithm suffers from slow convergence and poor sensitivity to the learning rate setting. We also found that classifying facial images using 2D-BP was more likely to cause over-fitting than for handwriting image recognition, even when the number of hidden nodes of 2D-FNN is suitable, and produces too many iterations in the training process. This is due to the complexity and noise level of facial images.

If the weights are tuned precisely on the training set, then the network may not perform well on future data (over-fitting), and if tune the weights are not tuned sufficiently, then it is possible to lead poor performance (under-fitting). To obtain best performance, a balanced trade-off between the bias and variance is required. This can be realized in NNRWs, where the hidden layer weights and biases are randomly assigned and only the output linear weights need to be tuned.

Inspired by the concept of NNRWs [10], Lu et al. [16] randomly assigned the left and right projecting vectors of 2D-FNN. So the output linear weights can be solved by the optimization

$$\tilde{\beta}_0 = \arg\min_{\tilde{\beta}} \sum_{p=1}^{N} \left\| \sum_{j=1}^{L} \beta_j g(\mathbf{u}_j^\top X_p \mathbf{v}_j + b_j) + \alpha - t_p \right\|_2^2,$$

where $\tilde{\beta} = [\beta_1, \beta_2, \ldots, \beta_L, \alpha]^\top$, $(X_p, t_p)(p = 1, 2, \ldots, N)$ are the training samples expressed in matrix format. The hidden output matrix becomes

$$\mathbf{G} = \begin{bmatrix} g(\mathbf{u}_1^\top X_1 \mathbf{v}_1 + b_1) & \ldots & g(\mathbf{u}_L^\top X_1 \mathbf{v}_L + b_L) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ g(\mathbf{u}_1^\top X_N \mathbf{v}_1 + b_1) & \ldots & g(\mathbf{u}_L^\top X_N \mathbf{v}_L + b_L) & 1 \end{bmatrix}_{N\times(L+1)}.$$

With the least squares method, $\tilde{\beta}_0 = \mathbf{G}^\dagger \mathbf{T}$, where $\mathbf{T} = [t_1, t_2, \ldots, t_N]^\top$, and $\mathbf{G}^\dagger$ is the generalized pseudo-inverse of $\mathbf{G}$. This algorithm is called 2D-NNRW.

2.3. Review for DNNE NNRW

For a collection of M base models, $f_1, \ldots, f_M$, and a training dataset, $(x_p, t_p)(p = 1, 2, \ldots, N)$, the ensemble collective output on the pth pattern is given by

$$\bar{f}(x_p) = \sum_{i=1}^{M} a_i f_i(x_p),$$

where a_i is the weight associated with the ith component model. The learning errors of the base models and the ensemble model can be defined, respectively, as follows

$$e_i = \sum_{n=1}^{N} \frac{1}{2} (f_i(x_p) - t_p)^2, \quad i = 1, \ldots, M, \tag{3}$$

and

$$E_{\text{ens}} = \frac{1}{N} \sum_{p=1}^{N} (\bar{f}(x_p) - t_p)^2.$$

The bias-variance–covariance decomposition [3] illustrates that in addition to the bias and variance of the individual estimators, the generalization ability of an ensemble also depends on the covariance between the estimators.

Bagging [1] and boosting [2] are two popular ensemble methods. They are, to some extent, similar. Both consist of a base model trained on different datasets to encourage diversity. In bagging, for each base model, the training dataset of N examples is randomly sampled with replacement. Thus, it is possible that some of the training examples may be repeated while others may be left out. In boosting, the training examples are sampled according to the performance of the previous base models. In boosting, predictors are combined with different weights that are decided in the learning stage, while in bagging, the output of the ensemble is a simple average of predictions from the base models. However, both algorithms focus only on minimizing the individual learning errors given in Eq. (3).

Unlike traditional ensemble learning approaches, NCL was introduced to train base models simultaneously in a cooperative manner that decorrelates the individual errors [5,32]. The learning error of the ith base model, given in Eq. (3), is modified to include a penalty term, P_i,

$$e_i = \sum_{n=1}^{N}\left[\frac{1}{2}(f_i(x_p)-t_p)^2 + \lambda P_i(x_p)\right], \tag{4}$$

where $\lambda \in [0,1]$ is a regularizing factor; and, usually,

$$P_i(x_p) = (f_i(x_p) - \bar{f}(x_p))\sum_{j\neq i}(f_j(x_p) - \bar{f}(x_p)).$$

NCL was proposed to reduce the covariance among ensemble individuals while the variance and bias terms were not increased. It has been shown in [6] to achieve this for multilayer perceptron (MLP) and radial basis function (RBF) networks. Recently, it was shown that NNRW also fits within the in NCL framework [15]. That is, given an ensemble of SLFN and a training dataset of N samples, the learning error of the ith base model, Eq. (3), was modified to include a decorrelation penalty term such as Eq. (4). When the weights and bias of the hidden layer of the base networks are randomly assigned, then the linear weights of each base SLFN can be determined by the optimization

$$\hat{\beta}_i = \arg\min_{\beta_i}\sum_{p=1}^{N}\left[\frac{1}{2}(f_i(x_p)-t_p)_2^2 - \lambda(f_i(x_p)-\bar{f}(x_p))^2\right], \tag{5}$$

where β_i is the output weights of the ith base network. Alhamdoosh and Wang [15] used uniform averaging weights to combine ensemble base SLFNs, i.e.,

$$\bar{f}(x_p) = \frac{1}{M}\sum_{i=1}^{M} f_i(x_p)$$

and

$$\sum_{j\neq i}(f_j(x_p)-\bar{f}(x_p)) = -(f_i(x_p)-\bar{f}(x_p)),$$

and here

$$f_i(x_p) = \sum_{j=1}^{L}\beta_{ij} g(w_{ij}x_p + b_{ij}).$$

Solving Eq. (5), we have [15]

$$\sum_{k=1}^{L} C_1\phi(i,j,i,k)\beta_{ik} + \sum_{l=1,l\neq i}^{M}\sum_{k=1}^{L} C_2\phi(i,j,l,k)\beta_{lk} = \psi(i,j),$$

where $i = 1,\dots,M,\ j = 1,\dots,L$; and

$$C_1 = 1 - 2\lambda\frac{(M-1)^2}{M^2}, \quad C_2 = 2\lambda\frac{M-1}{M^2};$$

$$\phi(i,j,k,l) = \sum_{p=1}^{N} g_{ij}(x_p)g_{kl}(x_p), \quad \psi(i,j) = \sum_{p=1}^{N} g_{ij}(x_p)t_p, \tag{6}$$

where

$$g_{ij}(X_p) = \begin{cases} g(\mathbf{u}_{ij}^\top X_p \mathbf{v}_{ij} + b_{ij}) & \text{if } j = 1,2,\dots,L; \\ 1 & \text{if } j = L+1. \end{cases}$$

This is a linear system that can be written in a matrix form as

$$\mathbf{H}_{\text{corr}}\mathbf{B}_{\text{ens}} = \mathbf{T}_h,$$

where $\mathbf{H}_{\text{corr}}$ is called the hidden correlation matrix,

$$\mathbf{H}_{\text{corr}} = \mathbf{H}_{\text{corr}}(p,q) = \begin{cases} C_1\phi(m,n,k,l) & \text{if } m = k; \\ C_2\phi(m,n,k,l) & \text{otherwise}, \end{cases}$$

where $p,\ q = 1,\dots,M\times L,\ m = \lceil \frac{p}{L} \rceil,\ n = ((p-1)\text{mod}L)+1,\ k = \lceil \frac{q}{L} \rceil$, $l = ((q-1)\text{mod}L)+1$; mod is the modulo operation; and $\mathbf{B}_{\text{ens}}$ and $\mathbf{T}_h$ are the global output weights matrix and hidden-target matrix, defined respectively as

$$\mathbf{B}_{\text{ens}} = [\beta_{11},\dots,\beta_{1L},\beta_{21},\dots,\beta_{2L},\dots,\beta_{M1},\dots,\beta_{ML}]^\top,$$

and

$$\mathbf{T}_h = [\psi(1,1),\dots,\psi(1,L),\psi(2,1),\dots,\psi(2,L),\dots,\psi(M,1),\dots,\psi(M,L)]^\top.$$

Then we have the least squares solution

$$\mathbf{B}_{\text{ens}} = \mathbf{H}_{\text{corr}}^\dagger\mathbf{T}_h,$$

by means of the generalized pseudo-inverse $\mathbf{H}_{\text{corr}}^\dagger$ of $\mathbf{H}_{\text{corr}}$. That is the DNNE NNRW algorithm. In our comparative experiments below, for consistency, we will use the model described in Eq. (1) that has output layer biases for this algorithm. To emphasize the difference of our method, we will call this algorithm DNNE the one-dimensional (1D) form of DNNE-NNRW (DDNE 1D-NNRW).

3. Decorrelated 2D neural network ensembles with random weights

To build well generalized neural network models for face recognition, we propose a new ensemble learning approach that uses 2D-FNNs as base components and combines NCL strategy with 2D-NNRW.

Given a collection of 2D-FNNs and a training dataset of N samples, the decorrelated error of the ith individual network modified by Eq. (5) is

$$e_i = \sum_{p=1}^{N} e_i(x_p) = \sum_{p=1}^{N}\left[\frac{1}{2}(f_i(X_p)-t_p)^2 - \lambda(f_i(X_p)-\bar{f}(X_p))^2\right], \tag{7}$$

here

$$f_i(X_p) = \sum_{j=1}^{L}\beta_{ij} g(\mathbf{u}_{ij}^\top X_p \mathbf{v}_{ij} + b_{ij}) + \alpha_i,$$

and X_p is the pth input pattern matrix. After applying uniform averaging weights and combining base 2D-SLFN,

$$\bar{f}(X_p) = \frac{1}{M}\sum_{i}^{M} f_i(X_p).$$

As discussed in Section 2, parameters $\mathbf{u}_{ij},\ \mathbf{v}_{ij}$, and b_{ij} of the basis function, g_{ij}, are randomly set, and the only free parameters to be tuned are the output layer weights and biases. Therefore, our NCL ensemble models attain optimal performance when the gradient of the error function in Eq. (7) vanishes with respect to the linear parameters β_{ij} and α_i. That is,

$$\nabla e_i = 0,\ i = 1,\dots,M,$$

which leads to

$$\frac{\partial e_i}{\partial \beta_{ij}} = \sum_{p=1}^{N}\frac{\partial e_i(x_p)}{\partial \beta_{ij}} = 0,\ j = 1,\dots,L, \quad \text{and} \quad \frac{\partial e_i}{\partial \alpha_i} = \sum_{p=1}^{N}\frac{\partial e_i(x_p)}{\partial \alpha_i} = 0.$$

Eq. (6) can be modified as [15]

$$\phi(i,j,k,l) = \sum_{p=1}^{N} g_{ij}(X_p)g_{kl}(X_p), \quad \psi(i,j) = \sum_{p=1}^{N} g_{ij}(X_p)t_p,$$

where

$$g_{ij}(X_p) = \begin{cases} g(\mathbf{u}_{ij}^\top X_p \mathbf{v}_{ij} + b_{ij}) & \text{if } j = 1,2,\dots,L; \\ 1 & \text{if } j = L+1. \end{cases}$$

The linear equation system can be defined as

$\mathbf{G}_{\text{corr}}\tilde{\mathbf{B}}_{\text{ens}} = \mathbf{T}_h,$

where $\mathbf{G}_{\text{corr}}$ is the hidden correlation matrix,

$$\mathbf{G}_{\text{corr}} = \mathbf{G}_{\text{corr}}(p,q) = \begin{cases} C_1\phi(m,n,k,l) & \text{if} \quad m = k; \\ C_2\phi(m,n,k,l) & \text{otherwise}; \end{cases}$$

where $p,\ q = 1,\dots,M \times \widetilde{L},\ m = \left\lceil \frac{p}{\widetilde{L}} \right\rceil,\ n = ((p-1)\text{mod}\widetilde{L}) + 1,\quad k = \left\lceil \frac{q}{\widetilde{L}} \right\rceil,\ l = ((q-1)\text{mod}\widetilde{L}) + 1$, and $\widetilde{L} = L + 1$. The global linear weights and the hidden-target matrix are, respectively,

$\tilde{\mathbf{B}}_{\text{ens}} = [\beta_{11},\dots,\beta_{1\widetilde{L}},\beta_{21},\dots,\beta_{2\widetilde{L}},\dots,\beta_{M1},\dots,\beta_{M\widetilde{L}}]^\top,$

and

$\mathbf{T}_h = [\psi(1,1),\dots,\psi(1,\widetilde{L}),\psi(2,1),\dots,\psi(2,\widetilde{L}),\dots,\psi(M,1),\dots,\psi(M,\widetilde{L})]^\top.$

Thus, we have the least squares solution

$$\tilde{\mathbf{B}}_{\text{ens}} = \mathbf{G}^\dagger_{\text{corr}}\mathbf{T}_h, \tag{8}$$

where $\mathbf{G}^\dagger_{\text{corr}}$ is the generalized pseudo-inverse of $\mathbf{G}_{\text{corr}}$.

Algorithm 1 describes our proposed learning scheme for building decorrelated 2D-FNN ensemble with random weights (DNNE 2D-NNRW).

Algorithm 1. DNNE 2D-NNRW Algorithm.

```
Require: Training dataset (matrix inputs); a proper basis function, $g : \mathbf{R} \mapsto \mathbf{R}$; penalty coefficient, $\lambda \in [0,1]$; the base model size, $L$; and the ensemble size, $M$.
Ensure: Trained 2D-FNN ensemble model (DNNE).
 1: Initialize the left and right projecting vectors and hidden biases of $M$ base 2D-SLFN models, i.e., $\mathbf{u}_{ij},\ \mathbf{v}_{ij}$, and $b_i$ are randomly set.
 2: Calculate the constants $C_1,\ C_2$, and the outputs of the hidden layers of base models for all patterns in training dataset, i.e., calculate $g_{ij}(X_p)$.
 3: for $p \leftarrow 1$ to $M \times \widetilde{L}$ do
 4:   $m \leftarrow \left\lceil \frac{p}{\widetilde{L}} \right\rceil$ ;  $n \leftarrow ((p-1)\text{mod}\widetilde{L}) + 1$
 5:   for $q \leftarrow 1$ to $M \times \widetilde{L}$ do
 6:     $k \leftarrow \left\lceil \frac{q}{\widetilde{L}} \right\rceil$ ;  $l \leftarrow ((q-1)\text{mod}\widetilde{L}) + 1$
 7:     if $m = k$ then
 8:       $\mathbf{G}_{\text{corr}}(p,q) \leftarrow C_1\phi(m,n,k,l)$
 9:     else
10:       $\mathbf{G}_{\text{corr}}(p,q) \leftarrow C_2\phi(m,n,k,l)$
11:     end if
12:   end for
13: end for
14: $k \leftarrow 1$
15: for $i \leftarrow 1$ to $M$ do
16:   for $j \leftarrow 1$ to $\widetilde{L}$ do
17:     $\mathbf{T}_h[k] \leftarrow \varphi(i,j)$ ;  $k \leftarrow k+1$
18:   end for
19: end for
20: Calculate $\mathbf{G}^\dagger_{\text{corr}}$, the generalized pseudo-inverse of $\mathbf{G}_{\text{corr}}$.
21: Calculate the estimated global linear weights $\tilde{\mathbf{B}}_{\text{ens}}$ from Eq. (8).
22: for $i \leftarrow 1$ to $M$ do
23:   Output weights and biases of base model $i \leftarrow \tilde{\mathbf{B}}_{\text{ens}}[(i-1)\widetilde{L}+1 : i\widetilde{L}]$
24: end for
25: return 2D-FNN ensemble model (DNNE).
```

4. Performance evaluation

4.1. Databases

We select twelve datasets to analyse the performance of our method. These datasets are all face images for classification tasks, summary information is listed in Table 1. For each dataset, we randomly chose half of the images of each subject for training, and the remaining images for testing. The input attributes of all datasets were normalized to $[0,1]$ after they are transformed to greyscale images. We also employ downsampling and cropping to unify image sizes for each dataset.

- YALE: The YALE dataset (http://cvc.yale.edu/projects/yalefaces/yalefaces.html) was created by the computational vision and control center of Yale University. It contains 165 frontal face images of 15 individuals, under varying lighting conditions with different facial expressions.
- YALE_B: The extended Yale face database B [33] contains 16,128 images of 28 human subjects under 9 poses and 64 illumination conditions. We select 10 individuals with 64 images for each, totally 640 samples for our experiments.
- UMIST: The UMIST (now 'Sheffield') face database (https://www.sheffield.ac.uk/eee/research/iel/research/face) consists of 574 images of 20 individuals (mixed race/gender/appearance). Each individual is shown in a range of poses from profile to frontal views. We get the cropped images from http://www.cs.nyu.edu/roweis/data.html.
- PIE: The CMU pose, illumination, and expression (PIE) database [34] was collected between October and December 2000. There are 41,368 images of 68 people. By extending the CMU 3D Room they were able to image each person under 13 different poses, 43 different illumination conditions, and with 4 different expressions. We select 50 subjects with 10 images per person for the experiments.
- Olivetti: The Olivetti database (formerly the ORL database of faces) (http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html) contains 10 images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement). We source the data from http://www.cs.nyu.edu/roweis/data.html.
- JEFFE: The Japanese female facial expression database (http://www.kasrl.org/jaffe.html) contains 213 images of 7 facial expressions (6 basic facial expressions + 1 neutral) posed by 10 Japanese female models. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects. We choose 10 subjects with 20 images per person for the experiments.
- IMM: The IMM dataset contains 240 still images of 40 different human faces, all without glasses. The gender distribution is 7 females and 33 males. Images were acquired in January 2001 using a 640×480 JPEG format with a Sony DV video camera, DCR-TRV900E PAL.
- GT: The Georgia Tech (GT) face database contains images of 50 people. For each individual, there are 15 color images captured between 1 June and 15 November, 1999. Most of the images were taken in two sessions to account for variations in illumination conditions, facial expression, and appearance. The faces were captured at different scales and orientations.

Table 1
Summary of facial datasets used.

Dataset	Size $(m \times n)$	Class	No. of data	No. of training	No. of testing
YALE	15	50×50	165	82	82
YALE_B	10	42×48	640	320	320
UMIST	20	56×46	574	287	287
PIE	50	32×32	1000	500	500
Olivetti	40	64×64	400	200	200
JEFFE	10	64×64	200	100	100
IMM	40	47×63	240	120	120
GT	50	77×55	750	375	375
FERET	70	56×46	420	210	210
FEI	200	48×64	2800	1400	1400
Faces94	154	50×45	3080	1540	1540
AR	100	60×43	1400	700	700

- FERET: The FERET database (http://www.nist.gov/itl/iad/ig/colorferet.cfm) images were collected in a semi-controlled environment and contains a total of 14,126 images from 1199 individuals. For some individuals, over 2 years had elapsed between their first and last sittings, with some subjects being photographed multiple times. We chose 70 subjects with 6 frontal images per person for the experiments.
- FEI: A Brazilian face database that contains images taken between June 2005 and March 2006 at the Artificial Intelligence Laboratory of FEI in São Bernardo do Cãmpo, Sao Paulo, Brazil (http://fei.edu.br/cet/facedatabase.html). There are 14 images for each of 200 individuals, a total of 2800 images. All images are in color and taken against a white homogenous background in an upright frontal position with profile rotation of up to 180°.
- Face94: The Face94 database contains 10 images of each of 154 individuals including males and females. The subjects of this database sat a fixed distance from the camera and were asked to speak while a sequence of images were taken. The speech is used to introduce facial expression variation.
- AR: The AR database contains 4000 color images corresponding to 126 faces (70 men and 56 women). They are frontal view faces with different facial expressions, illumination conditions, and occlusions (sun glasses and scarf). We select 100 individuals with 14 images for each, totally 1400 images for our experiments.

4.2. Experiment setup

For classification with neural networks, the objective value of each sample is a label vector. That is, if there are c classes to be classified, then we set $t_p = [0, \dots, 0, 1, 0, \dots, 0] \in \mathbf{R}^c$, where 1 only appears in the ith position, and represents the pth pattern belongs to the ith class. In the case of recognition, we can judge the class that x belongs to by finding the position of the maximum component of its real output, $f(x) \in \mathbf{R}^c$.

In all neural networks, we use the sigmoidal activation function $g(x) = 1/(1 + e^{-x})$. All the results are the average of 10 runs with independent initialization. The performance evaluations were carried out in Matlab R2014b environment running on a multiple-CPU (2.7 GHz) computer with 2 GB of RAM.

4.3. Results and discussion

Fig. 2 shows the test accuracy rates on the test sets for DNNE 2D-NNRW (solid lines), Simple averaging 2D-NNRW (dashed lines), and DNNE 1D-NNRW (dashed lines with circles) for different base sizes for all the datasets. The ensemble size is 8 for all the methods and the penalty coefficient is 0.5 for all DNNE methods. For most datasets, the accuracy rate improves and becomes more stable as the base size increases. For the AR dataset, the accuracy rate is still rising as the base network size approaches 45, whereas for YALE dataset, over-fitting is evident once the base size exceeds 25 using DNNE 1D-NNRW and our DNNE 2D-NNRW. This is due to the complexities, scale, and noise level of the different datasets. In all cases, our proposed algorithm attains better accuracy than other two methods. We used these outcomes to choose a suitable base network size for each dataset for the subsequent comparisons.

It is not only the base size that impacts the generalization ability of an ensemble method. Fig. 3 shows how the ensemble size affects accuracy on the test sets using DNNE 2D-NNRW (solid lines), Simple averaging 2D-NNRW (dashed lines), and DNNE 1D-NNRW (dashed lines with circles). The penalty coefficient was 0.5 for all DNNE methods, and the number of hidden nodes of base networks, i.e., the base model sizes, were chosen from the previous outcomes (Table 4). Increasing the ensemble size is always beneficial for the generalization performance and our DNNE 2D-NNRW algorithm is always better than DNNE 1D-NNRW and simple averaging 2D-NNRW methods regardless of the ensemble size. Approximately $8 \sim 15$ base networks are sufficient to constitute well generalized ensemble models for our approach.

The penalty coefficient of NCL also plays a key role in the performance of DNNE methods. Fig. 4 shows how different penalty coefficients affect accuracy on the test sets using DNNE 2D-NNRW and DNNE NNRW. The same base model sizes were used as in Table 4 and the ensemble sizes were chosen appropriately from the previous tests (see Fig. 3). The penalty coefficient range was $[0, 1]$, with step 0.05. We only show the efficient ranges, and our results are consistent with the 'upper bound' $(\lambda = \frac{M}{M-1})$ [6].

We also compared our method with bagging [1] and adaboost [4] approaches, as well as K-nearest neighbor (KNN) algorithm and the gradient-based 2D-BP learning algorithm [31]. Table 2 summarizes the comparison of all the methods, DNNE NNRW, single 2D-NNRW, simple averaging 2D-NNRW, bagging 2D-NNRW, adaboost, 2D-NNRW, 2D-BP, and KNN. The training time of the methods are shown in Table 3, and the parameter settings used for the various methods for these outcomes are listed in Table 4.

The base network size is constant for all the methods except 2D-BP. The ensemble sizes for our DNNE 2D-NNRW and DNNE 1D-NNRW were set to $8 \sim 15$, because we showed above that it is sufficient to constitute well generalized ensemble models using our algorithm. An exhaustive linear search was adopted to find optimal values for the other parameters shown in Table 4. The penalty coefficients, λ, for DNNE methods were searched in the range $[0, 1]$ with step 0.05. The ensemble sizes, M, for bagging and adaboost were searched in the range $[2, 30]$ with step 1. Note that for the GT, FEI, and Face94 datasets, adaboost failed, due to the limitation of its particular training process, i.e., when the training accuracy rate of the base components are below 0.5, adaboost

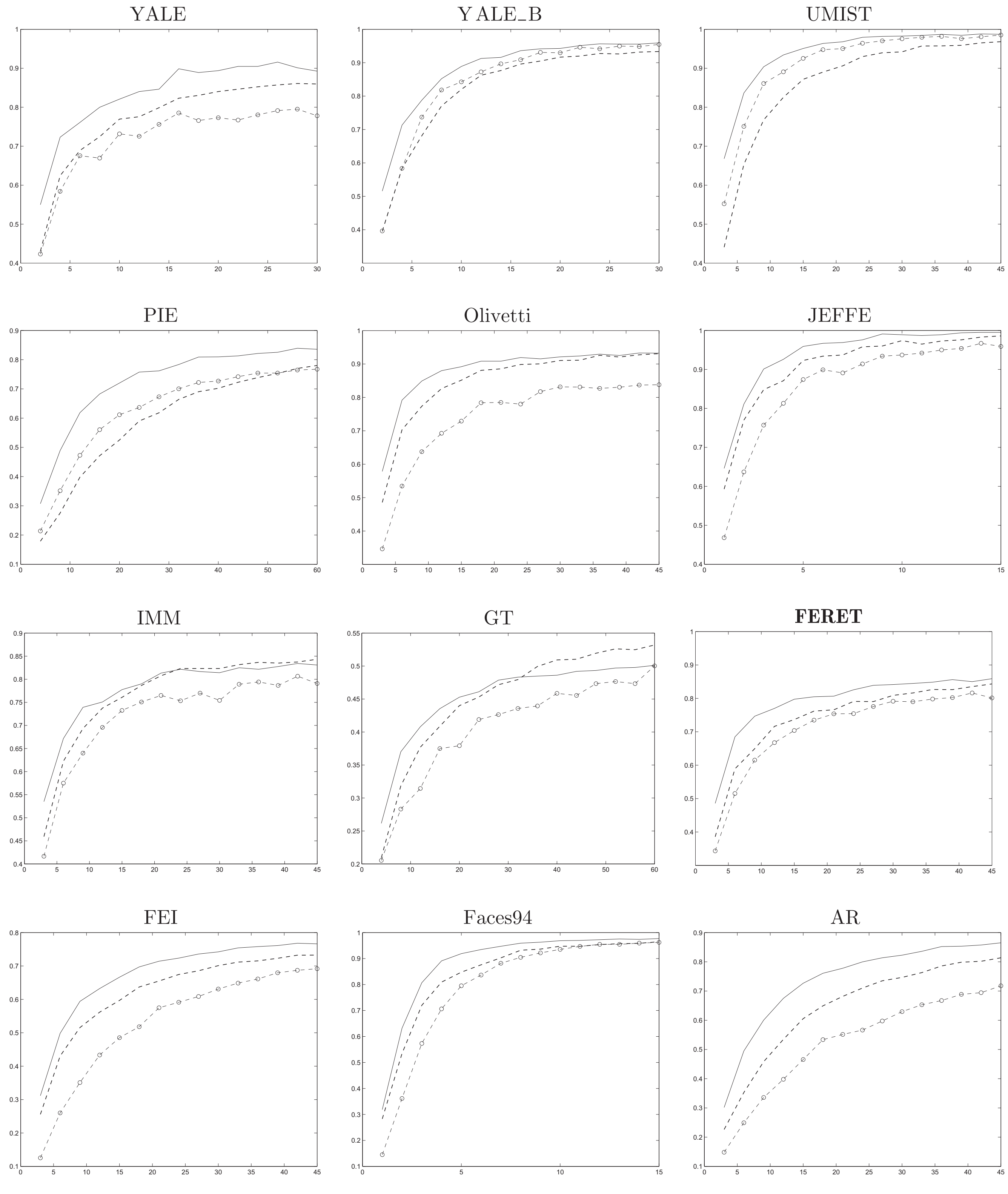

Fig. 2. Average accuracy rates (vertical axis) on the test sets for **different base sizes** (horizontal axis). Solid lines = DNNE 2D-NNRW, dashed lines = Simple Averaging 2D-NNRW, dashed lines with circles = DNNE 1D-NNRW.

fails. Bagging and adaboost approaches achieve their best performance when they use almost three times as many base models as our proposed DNNE 2D-NNRW method. However, their testing accuracy rate remains always are still lower than those from the proposed method.

The optimal base network sizes for 2D-BP were searched in the range [10, 100] with step 10. There are other two parameters to be set for 2D-BP, the leaning rate and momentum, which were set to 0.1 and 0.8, respectively. For gradient based BP learning algorithms, we use early stop to prevent over-fitting. 2D-BP always used significantly more training time than NNRW approaches. We can also see that 2D-BP was not always successful for face image classification. Because it is a relatively strong approximating learning algorithm, it is more affected by noise. Thus, 2D-BP is

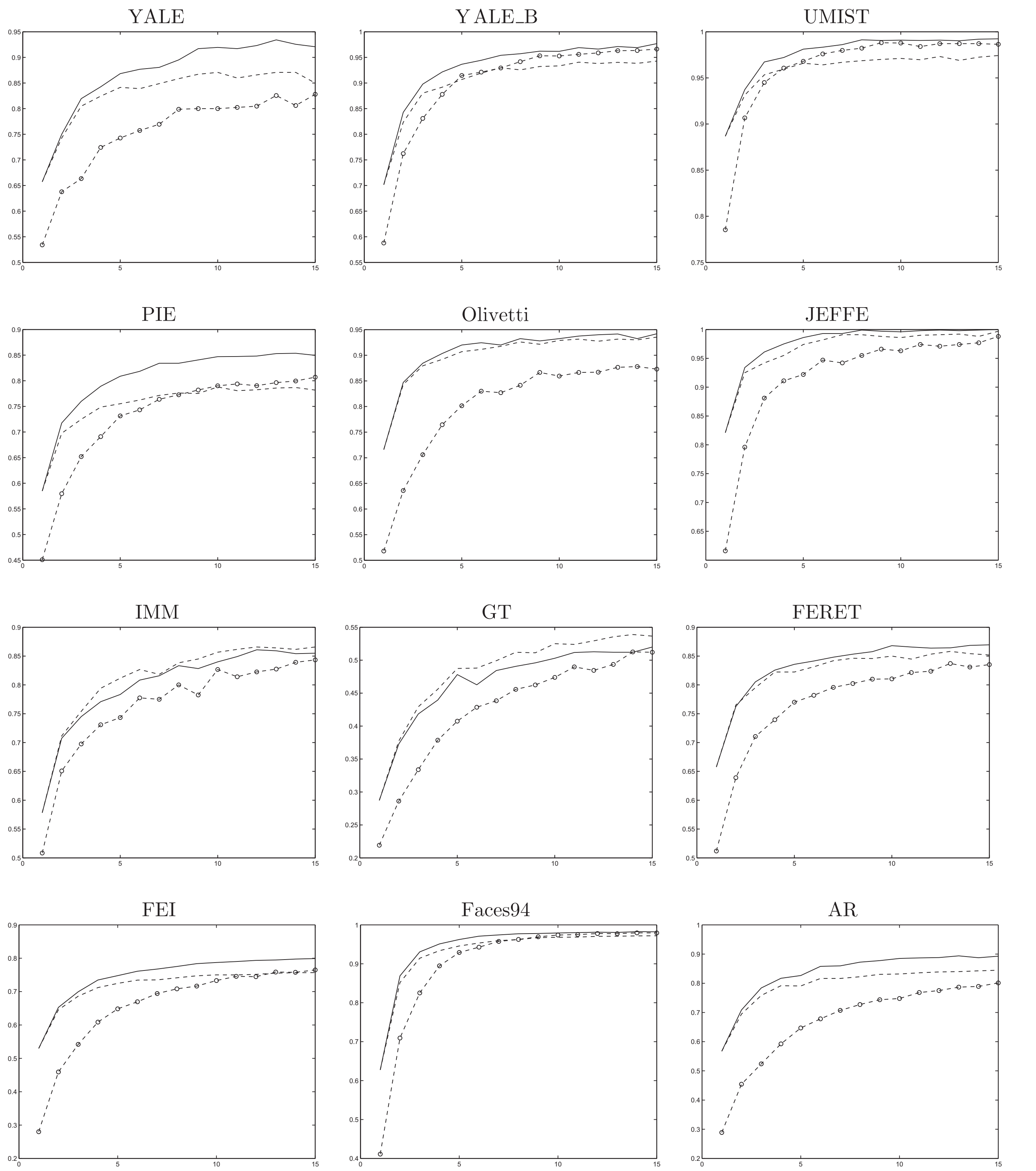

Fig. 3. Average accuracy rate (vertical axis) on the test sets for **different ensemble size** (horizontal axis). Solid lines = DNNE 2D-NNRW, dashed lines = Simple Averaging 2D-NNRW, and dashed lines with circles = DNNE 1D-NNRW.

more suitable for classifying images with less distortion and noise, such as handwritten digits images and face datasets such as UMIST rather than the GT, FEI, or AR datasets.

The KNN algorithm is an efficient classifier. We searched for the optimal number of nearest neighbors in the range [1, 10]. KNN, of course, requires no training time. However, from a practical point of view, the most serious problem with KNN is the high computational complexity, especially for multiple classification tasks with large scale data. Comparing the accuracy on test sets (Table 2), our proposed DNNE 2D-NNRW method is generally superior to KNN.

We set the random range $\Omega = 0.5$ for all NNRW approaches. This is an important factor that should be well selected for NNRW (see Fig. 1) and 2D-NNRW methods. However, it can be relaxed

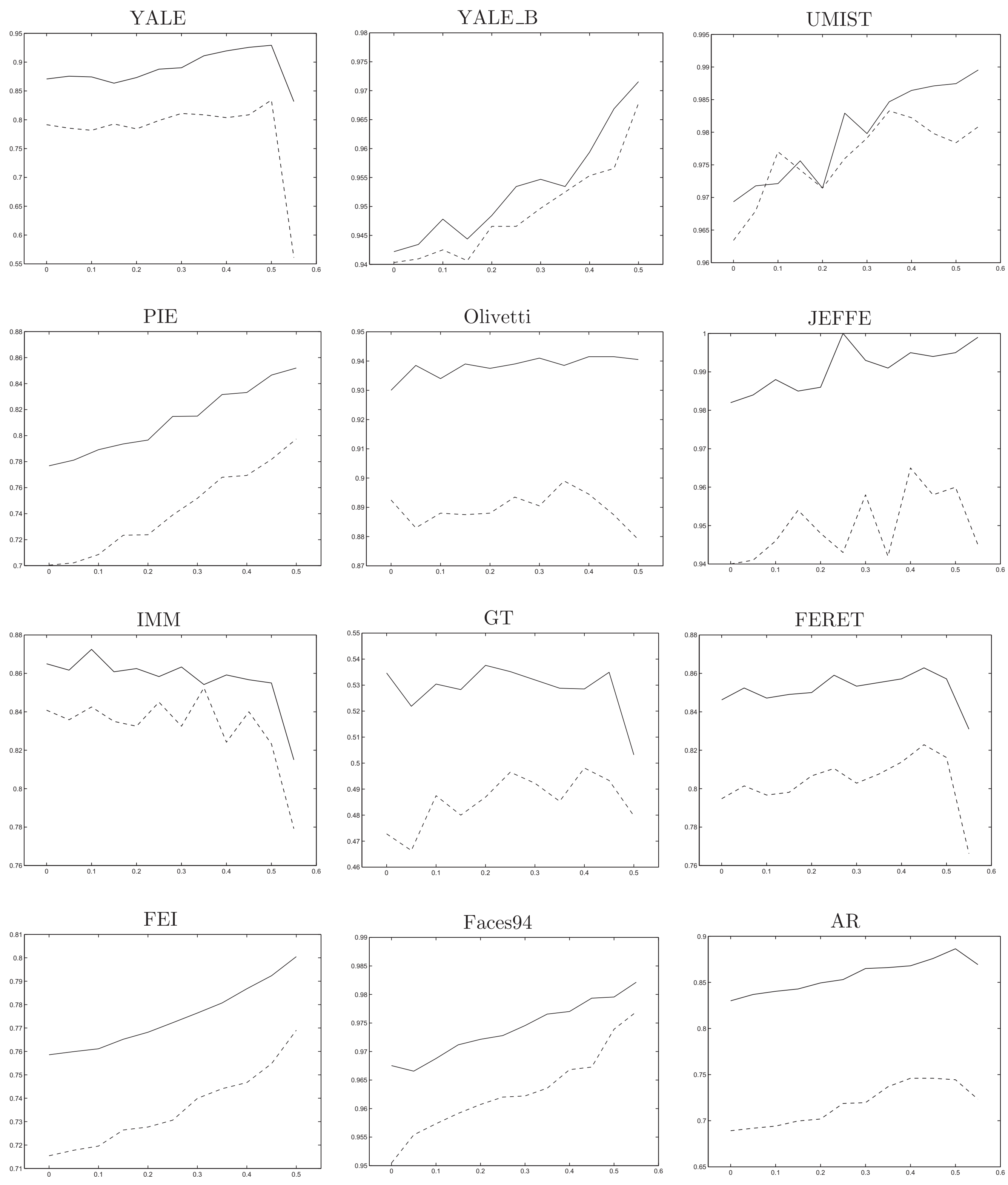

Fig. 4. Average accuracy rate (vertical axis) on test sets for **different penalty coefficient values** (horizontal axis). Solid lines = DNNE 2D-NNRW, dashed lines = DNNE 1D-NNRW.

using ensemble approaches. The random range affects ensemble significantly less than single 2D-NNRW methods (see Fig. 5). Hence, it is unnecessary to search for an optimal Ω using our proposed DNNE 2D-NNRW method, and here the setting $\Omega = 0.5$ is sufficient.

We now use the Wilcoxon signed-ranks method adopted in [35] to compare the statistical performance of the eight algorithms. All statistical comparisons are conducted under the significance level of 0.1. According to the average results in Table 2 and the statistical analysis with Wilcoxon signed-ranks method, we see that com-

Table 2
Comparison of testing accuracy rates.

Dataset	DNNE 2D-NNRW	DNNE 1D-NNRW	Single 2D-NNRW	Simple 2D-NNRW	Bagging 2D-NNRW	Adaboost 2D-NNRW	2D-BP	KNN
YALE	$\mathbf{0.929 \pm 0.025}$	0.834 ± 0.029	0.664 ± 0.031	0.870 ± 0.024	0.882 ± 0.022	0.885 ± 0.031	0.872 ± 0.035	0.792 ± 0.000
YALE_B	0.972 ± 0.003	0.967 ± 0.008	0.744 ± 0.019	0.942 ± 0.009	0.948 ± 0.006	0.972 ± 0.008	$\mathbf{0.983 \pm 0.005}$	0.850 ± 0.000
UMIST	0.990 ± 0.003	0.983 ± 0.008	0.874 ± 0.022	0.968 ± 0.006	0.971 ± 0.007	0.993 ± 0.003	0.972 ± 0.007	$\mathbf{0.996 \pm 0.000}$
PIE	$\mathbf{0.852 \pm 0.005}$	0.797 ± 0.008	0.578 ± 0.022	0.786 ± 0.011	0.778 ± 0.008	0.805 ± 0.011	0.579 ± 0.029	0.370 ± 0.000
Olivetti	$\mathbf{0.941 \pm 0.006}$	0.899 ± 0.017	0.740 ± 0.040	0.935 ± 0.006	0.939 ± 0.007	0.947 ± 0.007	0.677 ± 0.062	0.915 ± 0.000
JEFFE	$\mathbf{1.000 \pm 0.000}$	0.965 ± 0.018	0.830 ± 0.047	0.991 ± 0.013	0.997 ± 0.005	$\mathbf{1.000 \pm 0.000}$	0.993 ± 0.008	0.990 ± 0.000
IMM	$\mathbf{0.873 \pm 0.017}$	0.852 ± 0.022	0.590 ± 0.044	0.865 ± 0.019	0.863 ± 0.008	0.869 ± 0.029	0.726 ± 0.038	0.837 ± 0.000
GT	0.538 ± 0.015	0.498 ± 0.014	0.294 ± 0.030	0.524 ± 0.013	0.537 ± 0.013	Failed	Failed	$\mathbf{0.605 \pm 0.000}$
FERET	$\mathbf{0.863 \pm 0.014}$	0.823 ± 0.008	0.670 ± 0.023	0.850 ± 0.009	0.857 ± 0.014	0.856 ± 0.011	0.496 ± 0.056	0.614 ± 0.000
FEI	0.801 ± 0.005	0.769 ± 0.005	0.519 ± 0.021	0.758 ± 0.004	0.752 ± 0.008	Failed	Failed	$\mathbf{0.830 \pm 0.000}$
Face94	0.982 ± 0.001	0.977 ± 0.003	0.653 ± 0.031	0.968 ± 0.004	0.974 ± 0.003	Failed	0.253 ± 0.038	$\mathbf{0.986 \pm 0.000}$
AR	$\mathbf{0.886 \pm 0.010}$	0.746 ± 0.019	0.548 ± 0.024	0.832 ± 0.013	0.846 ± 0.006	0.861 ± 0.009	Failed	0.711 ± 0.000

Table 3
Comparison of training time (s).

Dataset	DNNE 2D-NNRW	DNNE 1D-NNRW	Single 2D-NNRW	Simple 2D-NNRW	Bagging 2D-NNRW	Adaboost 2D-NNRW	2D-BP
YALE	0.75	0.67	0.01	0.60	0.30	0.49	2.89
YALE_B	2.88	2.70	0.04	2.44	0.85	2.03	10.7
UMIST	2.14	2.06	0.05	1.89	1.32	2.20	13.6
PIE	18.8	18.0	0.06	13.5	1.94	3.34	21.2
Olivetti	6.01	5.09	0.03	3.92	1.10	1.74	11.5
JEFFE	0.19	0.17	0.01	0.17	0.48	0.66	3.95
IMM	2.69	2.47	0.02	1.62	0.59	1.02	5.68
GT	4.44	4.46	0.06	3.66	2.00		
FERET	3.04	3.00	0.03	2.20	1.09	1.89	10.8
FEI	38.1	37.7	0.25	34.4	7.57		
Face94	3.12	2.70	0.16	3.06	6.16		70.6
AR	9.61	9.26	0.12	8.09	3.50	6.01	

Table 4
Parameter values used for the results in Tables 2 and 3.

Dataset	DNNE 2D-NNRW			DNNE 1D-NNRW			Single 2D-NNRW	Simple 2D-NNRW		Bagging 2D-NNRW		Adaboost 2D-NNRW		2D-BP	KNN
	L_1[a]	M_1[b]	λ_1[c]	L_2[a]	M_2[b]	λ_2[c]	L_3[a]	L_4[a]	M_4[b]	L_5[a]	M_5[b]	L_6[a]	M_6[b]	L_7[a]	k[d]
YALE	25	13	0.50	25	13	0.50	25	25	13	25	24	25	24	30	2
YALE_B	25	15	0.50	25	15	0.50	25	25	15	25	20	25	28	30	1
UMIST	45	8	0.55	45	8	0.35	45	45	8	45	25	45	26	60	1
PIE	55	14	0.50	55	14	0.50	55	55	14	55	29	55	29	70	1
Olivetti	40	15	0.40	40	15	0.35	40	40	15	40	27	40	26	60	1
JEFFE	15	8	0.25	15	8	0.40	15	15	8	15	29	15	25	30	1/2/3
IMM	40	12	0.10	40	12	0.35	40	40	12	40	27	40	28	50	1
GT	40	11	0.20	40	11	0.40	40	40	11	40	26				1
FERET	45	10	0.45	45	10	0.45	45	45	10	45	29	45	30	60	1
FEI	50	15	0.45	50	15	0.40	50	50	15	50	27				1
Face94	15	10	0.55	15	10	0.55	15	15	10	15	30			50	1
AR	50	10	0.50	50	10	0.40	50	50	10	50	28	50	28		1

[a] The base size.
[b] The ensemble size.
[c] The penalty coefficient.
[d] The number of nearest neighbors.

pared with other seven algorithms, DNNE 2D-NNRW obtains considerably better performance on 12 face datasets. The counts of wins, ties, and losses regarding the comparison among the eight algorithms are shown in Table 5.

On the other hand, for a given learning algorithm, the number of wins in m datasets obeys the normal distribution $N\left(\frac{m}{2}, \frac{\sqrt{m}}{2}\right)$ under the null-hypothesis in the sign test [36]. It can be asserted that the given learning algorithm is significantly better than other one under the significance level a, when the number of wins is at least $\frac{m}{2} + Z_{a/2} \times \frac{m}{2}$. In our experiments, let $a = 0.1$ and $m = 12$, then $8 < \frac{m}{2} + Z_{a/2} \times \frac{m}{2} < 9$. Namely, DNNE 2D-NNRW will achieve considerable better performance if the number of wins on 12 face datasets reaches 8. This can be verified from Table 5.

Table 5
The counts of wins, ties, and losses regarding the comparison among the eight algorithms.

	Testing result
DNNE 2D-NNRW versus DNNE 1D-NNRW (win/tie/loss)	12/0/0
DNNE 2D-NNRW versus Single 2D-NNRW (win/tie/loss)	12/0/0
DNNE 2D-NNRW versus Simple 2D-NNRW (win/tie/loss)	12/0/0
DNNE 2D-NNRW versus Bagging 2D-NNRW (win/tie/loss)	12/0/0
DNNE 2D-NNRW versus Adaboost 2D-NNRW (win/tie/loss)	9/2/1
DNNE 2D-NNRW versus 2D-BP (win/tie/loss)	11/0/1
DNNE 2D-NNRW versus KNN (win/tie/loss)	8/0/4

In summary, our proposed ensemble method, DNNE 2D-NNRW, is an efficient algorithm for building well generalized neural network ensembles for face recognition.

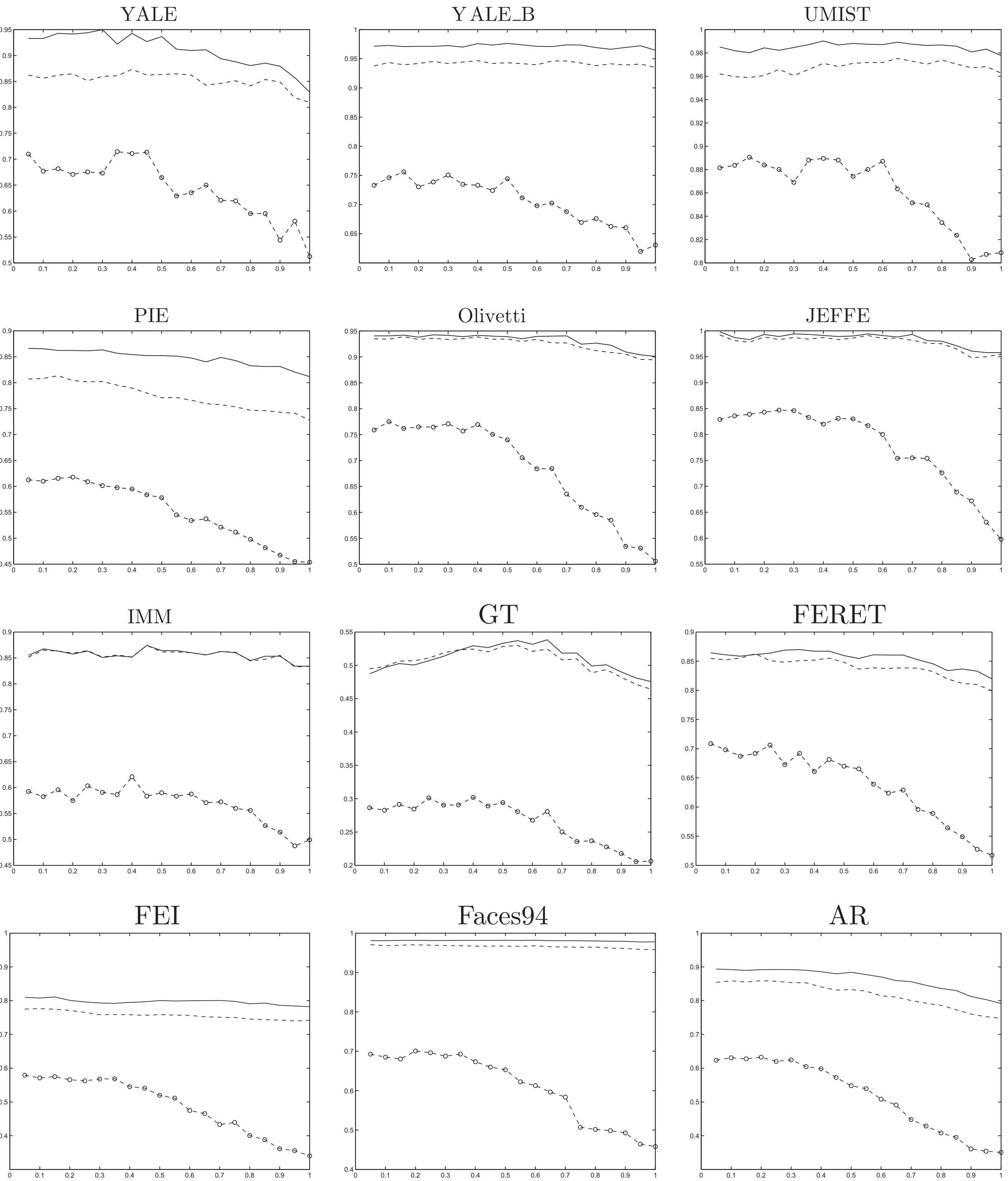

Fig. 5. Average accuracy rate (vertical axis) on the test sets for **different random range** (horizontal axis). Solid lines = DNNE 2D-NNRW, dashed lines = Simple Averaging 2D-NNRW, and dashed line with circles = Single 2D-NNRW.

5. Conclusion and further study

NCL, one of the most popular ensemble methods, is viewed as a framework rather than as an algorithm and we note that 2D-NNRW fits in this framework. We proposed a new effective ensemble learning approach called DNNE 2D-NNRW that uses 2D-FNN base models and combines 2D-NNRW with NCL for building neural network ensembles. It combines the advantages of the 2D-NNRW algorithm and NCL strategy, and produces well generalized ensemble classifiers for face recognition.

However, considering the theoretical basis of the proposed ensemble approach, the approximating property of 2D-NNRW has not yet been proved. Since our proposed DNNE 2D-NNRW algorithm was specifically designed for face recognition, its effectiveness for general image recognition certainly will be weaken. Therefore, in the future study, it is an important task to explore the approximation capability of 2D-RRNW and its effectiveness to more extensive images.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.knosys.2015.09.002.

References

[1] L. Breiman, R. Quinlan, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[2] Y. Freund, Boosting a weak learning algorithm by majority, Inform. Comput. 121 (2) (1995) 256–285.
[3] N. Ueda, R. Nakano, Generalization error of ensemble estimators, in: Proc. IEEE Inter. Conf. Neural Networks, Washington DC, USA, 1996, pp. 90–95.
[4] R. Rojas, AdaBoost and the Super Bowl of Classifiers a Tutorial Introduction to Adaptive Boosting, Tech. Rep., Freie University, Berlin, 2009.
[5] B.E. Rosen, Ensemble learning using decorrelated neural networks, Connect. Sci. 8 (3) (1996) 373–384.
[6] G. Brown, J.L. Wyatt, Tiňo P, Managing diversity in regression ensembles, J. Mach. Learn. Res. 6 (2005) 1621–1650.
[7] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks 12 (10) (1999) 1399–1404.
[8] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, IEEE Trans. Evol. Comput. 4 (4) (2000) 380–387.
[9] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Trans. Neural Netw. 20 (12) (2009) 1962–1979.
[10] W.F. Schmidt, M.A. Kraaijveld, R.P.W. Duin, Feedforward neural networks with random weights, in: Proc. 11th IAPR Inter. Conf. Pattern Recog. Meth. & Sys., 1992, pp. 1–4.
[11] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Applications, Wiley, New York, 1971.
[12] B. Igelnik, Y.H. Pao, Stochastic choice of basis functions in adaptive function approximation and the functional-link net, IEEE Trans. Neural Netw. 6 (6) (1995) 1320–1329.
[13] Y.H. Pao, G.H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[14] J.W. Zhao, Z.H. Wang, F.L. Cao, D.H. Wang, A local learning algorithm for random weights networks, Knowl.-Based Syst. 74 (2015) 159–166.
[15] M. Alhamdoosh, D.H. Wang, Fast decorrelated neural network ensembles with random weights, Inform. Sci. 264 (2014) 104–117.
[16] J. Lu, J.W. Zhao, F.L. Cao, Extended feed forward neural networks with random weights for face recognition, Neurocomputing 136 (2014) 96–102.
[17] I.Y. Tyukin, D.V. Prokhorov, Feasibility of random basis function approximators for modeling and control, in: Proceeding of Multi-Conference on Systems and Control, Saint Petersburg, Russia, 2009, pp. 1391–1396.
[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[19] R. Fletcher, C.M. Reeves, Function minimization by conjugate gradients, Comput. J. 7 (2) (1964) 149–154.
[20] S. McLoone, G.W. Irwin, Fast parallel off-line training of multilayer perceptrons, IEEE Trans. Neural Netw. 8 (3) (1997) 646–653.
[21] R. Battiti, F. Masulli, BFGS optimization for faster and automated supervised learning, in: International Neural Network Conference, Springer, Netherlands, 1990, pp. 757–760.
[22] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[23] G. Cybenko, Approximation by superposition of sigmoidal function, Math. Control, Signals Syst. 2 (1989) 303–314.
[24] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inform. Theory 39 (3) (1993) 930–945.
[25] S. Geman, E. Bienenstock, R. Doursat, Neural networks and the bias/variance dilemma, Neural Comput. 4 (1) (1992) 1–58.
[26] J. Yang, D. Zhang, A.F. Frangi, J.Y. Yang, Two-dimensional PCA: a new approach to appearance-based face representation and recognition, IEEE Trans. Pattern Anal. Mach. Intell. 26 (1) (2004) 131–137.
[27] D. Zhang, Z.H. Zhou, (2D)2PCA: two-directional two-dimensional PCA for efficient face representation and recognition, Neurocomputing 69 (1) (2005) 224–231.
[28] M. Li, B. Yuan, 2D-LDA: a statistical linear discriminant analysis for image matrix, Pattern Recogn. Lett. 26 (5) (2005) 527–532.
[29] P. Sanguansat, W. Asdornwised, S. Jitapunkul, M. Sanparith, Two-dimensional linear discriminant analysis of principle component vectors for face recognition, IEICE Trans. Inform. Syst. 89 (7) (2006) 2164–2170.
[30] C. Hou, F. Nie, D. Yi, Y. Wu, Efficient image classification via multiple rank regression, IEEE Trans. Image Process. 22 (1) (2013) 340–352.
[31] K.K. Dai, J.W. Zhao, F.L. Cao, A novel algorithm of extended neural networks for image recognition, Eng. Appl. Artif. Intell. 42 (2015) 57–66.
[32] Y. Liu, X. Yao, Negatively correlated neural networks can produce best ensembles, Aust. J. Intell. Inform. Process. Syst. 4 (3/4) (1997) 176–185.
[33] A.S. Georghiades, P.N. Belhumeur, D. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 643–660.
[34] R. Gross, I. Matthews, J. Cohn, T. Kanade, S. Baker, Multi-pie, Image Vis. Comput. 28 (5) (2010) 807–813.
[35] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.
[36] X.Z. Wang, Y.L. He, D.D. Wang, Non-naive bayesian classifiers for classification problems with continuous attributes, IEEE Trans. Cybern. 44 (1) (2014) 21–39.