Pattern Recognition 63 (2017) 149–157

Contents lists available at ScienceDirect

Pattern Recognition

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

Hierarchical deep neural network for multivariate regression

Jun Du*, Yong Xu

National Engineering Laboratory for Speech and Language Information Processing (NEL-SLIP), University of Science and Technology of China, No. 96, JinZhai Road, Hefei, Anhui, PR China

ARTICLE INFO

Keywords:
Divide and Conquer
Hierarchical Deep Neural Network
Multivariate Regression
Speech Enhancement
Handwritten Chinese Character Recognition

ABSTRACT

This paper presents the novel hierarchical deep neural network (HDNN) for the general multivariate regression problem. The recent insight of deep neural network (DNN) is the deep architecture with large training data can bring the best performance in many research areas. The architecture design of our proposed HDNN focuses on both "depth" and "width" of artificial neural network. Specifically for the multivariate regression, HDNN consists of multiple subnets, which is empirically more powerful than DNN by using a divide and conquer strategy. The effectiveness of HDNN as the regression model is verified on two tasks, namely speech enhancement and Chinese handwriting recognition. For the speech enhancement task, our experiments show that HDNN significantly outperforms DNN in terms of perceptual evaluation of speech quality (PESQ), which is an objective measure highly correlated to subjective testing of listening quality. And for Chinese handwriting recognition task, as a nonlinear feature mapping function, we have a very interesting observation that DNN-based approach can not even bring performance gain while HDNN-based approach yields significant improvements of recognition accuracy.

1. Introduction

The definition of *regression*, which could be traced back to Galton's hereditary research in 1886 [1], is to learn a function that captures the relationship between the input variables and their corresponding target outputs which are continuous variables [2]. One of the earliest and most widely used regression approach was the linear regression (or least square regression) which originated from Pearson's research on theory of evolution [3]. To improve the regression capability, other approaches, such as polynomial regression [4], logistic regression [5], and support vector regression [6,7] were also proposed. Based on the number of output variables, all these approaches can be divided into two categories, namely multiple regression [8] and multivariate regression [9]. Multiple regression has only one output variable to be predicted with multiple input variables. Multivariate regression extends to a more general case where multiple output variables are used, which is also the topic of this study.

In this work, we focus on another broad class of approaches, namely neural network based regression. In [10], it was rigorously proven that standard multilayer feedforward networks using arbitrary squashing functions were capable of approximating virtually any function of interest to any desired degree of accuracy, provided sufficiently many hidden units were available. Then the general regression neural networks [11] were widely used in identification and control of dynamical systems [12]. But early work using neural network in the 80 s and 90 s could not achieve satisfactory performance for two reasons. One reason is large scale training data could not be handled due to the computational limitation. The other one is random initialization of the weight parameters often suffered from "apparent local minima or plateaus" [13] especially when more hidden layers were used [14]. A breakthrough for training deep architectures came in 2006 when Hinton et al. [15,16] proposed a greedy layer-wise unsupervised learning algorithm, where each layer was pre-trained without supervision to learn a high level representation of its input (or the output of its previous layer). The unsupervised pre-training was an effective technique to alleviate the local minima problem. Later, Bengio et al. [17] further emphasized the importance of deep architectures and gave empirical analysis. Meanwhile, with the development of high performance computing, training neural networks with large scale data and many hidden layers turned into reality. There were many successful applications, e.g., deep neural networks (DNNs) in speech recognition [18], convolution neural networks (CNNs) in image classification [19,20], recurrent neural networks (RNNs) in handwriting recognition [21], etc. However, most of the previous efforts are made for classification problems. Only recently, deep learning via neural networks was adopted for solving the regression problems in several research areas. For example, in [22,23], deep recurrent neural networks (DRNNs) were used in feature enhancement for noise robust

[*] Corresponding author.
E-mail address: jundu@ustc.edu.cn (J. Du).

http://dx.doi.org/10.1016/j.patcog.2016.10.003
Received 23 June 2015; Received in revised form 12 April 2016; Accepted 1 October 2016
Available online 04 October 2016

speech recognition. DNNs with layer-wise generative training in [24] were adopted for voice conversion. More recently, DNNs were also successfully applied to speech enhancement [25–27] and speech separation [28,29], which demonstrated the superiority to traditional approaches.

The main difference between the regression and classification DNN is that each dimension of the output layer in the regression DNN is usually unbounded rather than the bounded posterior probability generated by softmax function in the classification DNN, which makes the learning of regression DNN potentially more difficult, especially when the input and output vectors are with high dimensions in the multivariate regression. Furthermore, one single regression DNN can not well accommodate for all the variabilities in many applications. For example, in speech enhancement we need to consider the speaker variabilities, different noise types and levels, etc. To address this problem, in this paper we adopt the divide and conquer strategy to design a novel architecture of neural network, namely the hierarchical deep neural network (HDNN). HDNN aims to decompose the original regression problem into multiple subproblems and solve them in different subspaces of the output vector. Compared with the regression approach using a single DNN, HDNN has the following advantages. First, HDNN is inherently more powerful than DNN as it consists of multiple deep subsets with each one solving one relatively easy problem, i.e., the mapping function between the input vector and one output subvector. Second, we can empirically prove that HDNN can achieve a better convergence property than DNN with the traditional stochastic gradient descent optimization. The experiments on two applications are designed to demonstrate the effectiveness of HDNN as a general regression model. One application is speech enhancement which could be formulated via a regression DNN to predict the clean speech given the noisy speech as the input [26,27] outperforming the traditional enhancement approaches [30,31]. Our proposed HDNN-based enhancement approach in multiple frequency subbands can achieve consistent performance improvements over the DNN-based enhancement approach in the full frequency band. The other application is Chinese handwriting recognition where a feature mapping/transformation as an irrelevant variability normalization is conducted via DNN or HDNN [32]. It is interesting that DNN-based approach can not even bring performance gain while HDNN-based approach yields significant improvements of recognition accuracy which demonstrates the more powerful model capability of HDNN.

The remainder of the paper is organized as follows. In Section 2, we first give an introduction of DNN/HDNN based multivariate regression. In Section 3, two applications using DNN/HDNN are elaborated. In Section 4, we report experimental results on speech enhancement and handwriting recognition. Finally we summarize our findings in Section 5.

2. DNN/HDNN based multivariate regression

2.1. Multivariate regression

The general multivariate regression problem can be formulated as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}; \mathbf{\Theta}) \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the D_1-dimensional input and D_2-dimensional output vectors with multiple variables, respectively. To learn the mapping function $\mathcal{F}$ with the set of parameters $\mathbf{\Theta}$, given the set of training samples pairs $\{(\mathbf{x}_r, \mathbf{y}_r)|r = 1, \ldots, R\}$, we aim at minimizing mean squared error (MMSE) function defined as:

$$E = \frac{1}{R}\sum_{r=1}^{R} \| \hat{\mathbf{y}}_r - \mathbf{y}_r \|_2^2 \tag{2}$$

where $\hat{\mathbf{y}}_r$ is the predicted output vector using Eq. (1). In the following subsections, two specific forms of $\mathcal{F}$, namely DNN and HDNN, are described to learn the corresponding set of parameters $\mathbf{\Theta}$. And empirical analysis is also given to demonstrate the superiority of HDNN over DNN for multivariate regression.

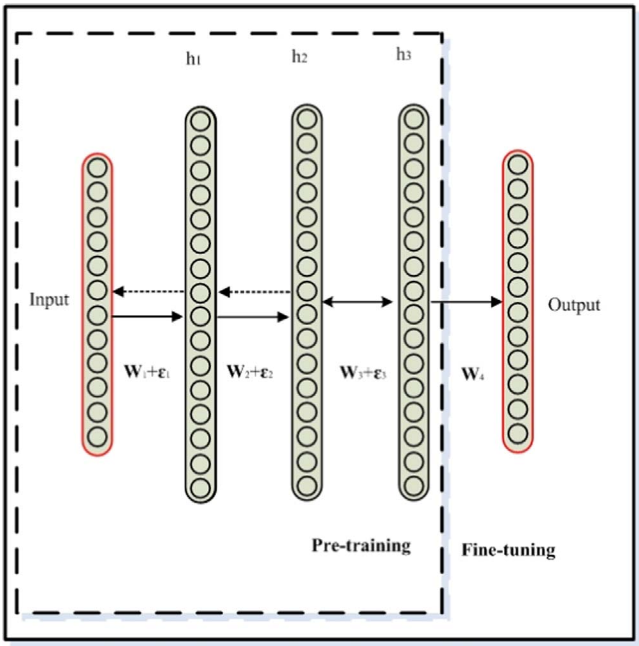

Fig. 1. DNN for multivariate regression.

2.2. Deep neural network

A deep neural network (DNN) is a feed-forward, artificial neural network with more than one layer of hidden units between its inputs and outputs [18]. In this study, DNN is adopted as a multivariate regression model to learn the mapping function between the input vector and the output vector as in Eq. (1). The architecture and training procedure of DNN is illustrated in Fig. 1, which consists of unsupervised pre-training and supervised fine-tuning..

The pre-training procedure treats each consecutive pair of layers as a restricted Boltzmann machine (RBM) [15] with the joint probability defined as:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}\exp\{-E(\mathbf{v}, \mathbf{h})\} \tag{3}$$

where $\mathbf{v}$ and $\mathbf{h}$ denote the observable variables and latent (hidden) variables, respectively. $E(\mathbf{v}, \mathbf{h})$ is an energy function and Z is the partition function to ensure $p(\mathbf{v},\mathbf{h})$ is a valid probability distribution. If both $\mathbf{v}$ and $\mathbf{h}$ are binary states, i.e., the Bernoulli-Bernoulli RBM, the energy function is given by

$$E(\mathbf{v}, \mathbf{h}) = -(\mathbf{b}_v^\top\mathbf{v} + \mathbf{b}_h^\top\mathbf{h} + \mathbf{v}^\top\mathbf{W}_{vh}\mathbf{h}) \tag{4}$$

where $\mathbf{b}_\upsilon$, $\mathbf{b}_h$ are bias vectors of $\mathbf{v}$ and $\mathbf{h}$ respectively, and $\mathbf{W}_{\upsilon h}$ is the weight matrix between them. If $\mathbf{v}$ is real-valued data and $\mathbf{h}$ is binary, i.e., the Gaussian-Bernoulli RBM, the energy function is:

$$E(\mathbf{v}, \mathbf{h}) = \tfrac{1}{2}(\mathbf{v} - \mathbf{b}_v)^\top(\mathbf{v} - \mathbf{b}_v) - \mathbf{b}_h^\top\mathbf{h} - \mathbf{v}^\top\mathbf{W}_{vh}\mathbf{h} \tag{5}$$

where we assume that the visible units follow the Gaussian noise model with an identity covariance matrix if the input data are pre-processed by mean and variance normalization.

The RBM parameters can be efficiently trained in an unsupervised fashion by maximizing the likelihood over training samples of visible units with the approximate contrastive divergence algorithm [15]. As for the DNN training, a Gaussian-Bernoulli RBM is used for the first layer while a pile of Bernoulli-Bernoulli RBMs can be stacked behind the Gaussian-Bernoulli RBM. Then the parameters of RBMs can be trained layer-by-layer. Hinton et al. indicate that this greedy layer-wise unsupervised learning procedure always helps over the traditional random initialization.

After pre-training for initializing the weights of the first several layers, a supervised fine-tuning of the parameters in the whole neural network with the final output layer is performed. Then Eq. (2) can be specified as:

$$E_{\mathrm{DNN}} = \frac{1}{R}\sum_{r=1}^{R} \| \mathcal{F}(\mathbf{x}_r; \mathbf{W}, \mathbf{b}) - \mathbf{y}_r \|_2^2 \tag{6}$$

where $\mathbf{W}$ and $\mathbf{b}$ denote all the weight and bias parameters of DNN. This objective function is optimized using back-propagation procedure with conjugate gradient method in mini-batch mode.

2.3. Hierarchical deep neural network

Although the powerful modeling capability of DNN has been demonstrated in many research areas, for the general multivariate regression it can not always guarantee to achieve a good performance as the mapping function might be very complicated, especially when both input and output vectors are with high dimension. For example, in one of the subsequent applications, namely Chinese handwriting recognition, DNN as a feature transformation even leads to the recognition performance degradation because the optimization of the MMSE criterion via the gradient approach might converge to a bad local optimal. To address this problem, we propose a more powerful model called hierarchical deep neural network illustrated in Fig. 2, which achieves both deep and wide architectures of neural network. The main principle of HDNN is to divide the vector of the output layer in Fig. 1 into K sub-vectors, each associated with a subnet or DNN using the same input layer and hidden layers, which adopts the divide and conquer strategy to make the learning of the regression function $\mathcal{F}$ easier. Then the formulation of HDNN is extended from Eq. (2) as:

$$E_{\mathrm{HDNN}} = \frac{1}{R}\sum_{r=1}^{R} \| \hat{\mathbf{y}}_r - \mathbf{y}_r \|_2^2 = \frac{1}{R}\sum_{r=1}^{R}\sum_{k=1}^{K} \| \hat{\mathbf{y}}_{r,k} - \mathbf{y}_{r,k} \|_2^2 = \sum_{k=1}^{K} E_k \tag{7}$$

and

$$E_k = \frac{1}{R}\sum_{r=1}^{R} \| \hat{\mathbf{y}}_{r,k} - \mathbf{y}_{r,k} \|_2^2 = \frac{1}{R}\sum_{r=1}^{R} \| \mathcal{F}(\mathbf{x}_r; \mathbf{W}_k^{\mathrm{H}}, \mathbf{b}_k^{\mathrm{H}}, \mathbf{W}_k^{\mathrm{O}}, \mathbf{b}_k^{\mathrm{O}}) - \mathbf{y}_{r,k} \|_2^2 \tag{8}$$

where $\hat{\mathbf{y}}_{r,k}$ and $\mathbf{y}_{r,k}$ are the kth subvectors of $\hat{\mathbf{y}}_r$ and $\mathbf{y}_r$, respectively. DNN is a special case of HDNN when K=1. The dimension of kth subvector is $D_{2,k}$ with the constraint $D_2 = \sum_{k=1}^{K} D_{2,k}$. Obviously, the optimization of Eq. (7) can be divided into K subproblems as in Eq. (8), each associated with one subnet corresponding to one output subvector in Fig. 2. For the kth subnet, there are two sets of weight and bias parameters, namely the parameters linked to all the hidden layers $(\mathbf{W}_k^{\mathrm{H}}, \mathbf{b}_k^{\mathrm{H}})$ and the output layer $(\mathbf{W}_k^{\mathrm{O}}, \mathbf{b}_k^{\mathrm{O}})$, respectively. Please note that all those subnets share the same input. One important issue in HDNN is the design of the subvectors which depends on the specific application. For example, in this study, it refers to the log-power spectra of clean speech in the frequency subband for the speech enhancement task while the dimensions with more useful information of linear discriminant analysis (LDA) transformed feature vector are selected as the subvectors for the Chinese handwriting recognition task. Furthermore, the clustering techniques can also be adopted to collect the similar dimensions in one subvector which might make the learning of each subproblem easier..

The training procedure of HDNN consists of two steps. The first step is to train a single DNN as in Section 2.2 with two sets of parameters $(\mathbf{W}_0^{\mathrm{H}}, \mathbf{b}_0^{\mathrm{H}})$ and $(\mathbf{W}_0^{\mathrm{O}}, \mathbf{b}_0^{\mathrm{O}})$. And the set $(\mathbf{W}_0^{\mathrm{O}}, \mathbf{b}_0^{\mathrm{O}})$ can be decomposed into K subsets $\{(\mathbf{W}_{0,k}^{\mathrm{O}}, \mathbf{b}_{0,k}^{\mathrm{O}}) | k = 1, \ldots, K\}$, each linked to the corresponding output subvector. Then the parameter sets of kth subnet of HDNN, namely $(\mathbf{W}_k^{\mathrm{H}}, \mathbf{b}_k^{\mathrm{H}})$ and $(\mathbf{W}_k^{\mathrm{O}}, \mathbf{b}_k^{\mathrm{O}})$, are initialized by $(\mathbf{W}_0^{\mathrm{H}}, \mathbf{b}_0^{\mathrm{H}})$ and $(\mathbf{W}_{0,k}^{\mathrm{O}}, \mathbf{b}_{0,k}^{\mathrm{O}})$, respectively. It is noted that all the subnets share the same parameters linked to all hidden layers and differ only in the parameters linked to the output subvectors at the initialization stage. The second step is to fine-tune all the parameters in each subnet independently using the objective function in Eq. (8) via the back-propagation with conjugate gradient method in mini-batch mode. With the initialization of HDNN using DNN in the first step, the second step can empirically guarantee that the learning curve of HDNN achieves a better optimal than that of DNN as shown in Fig. 3..

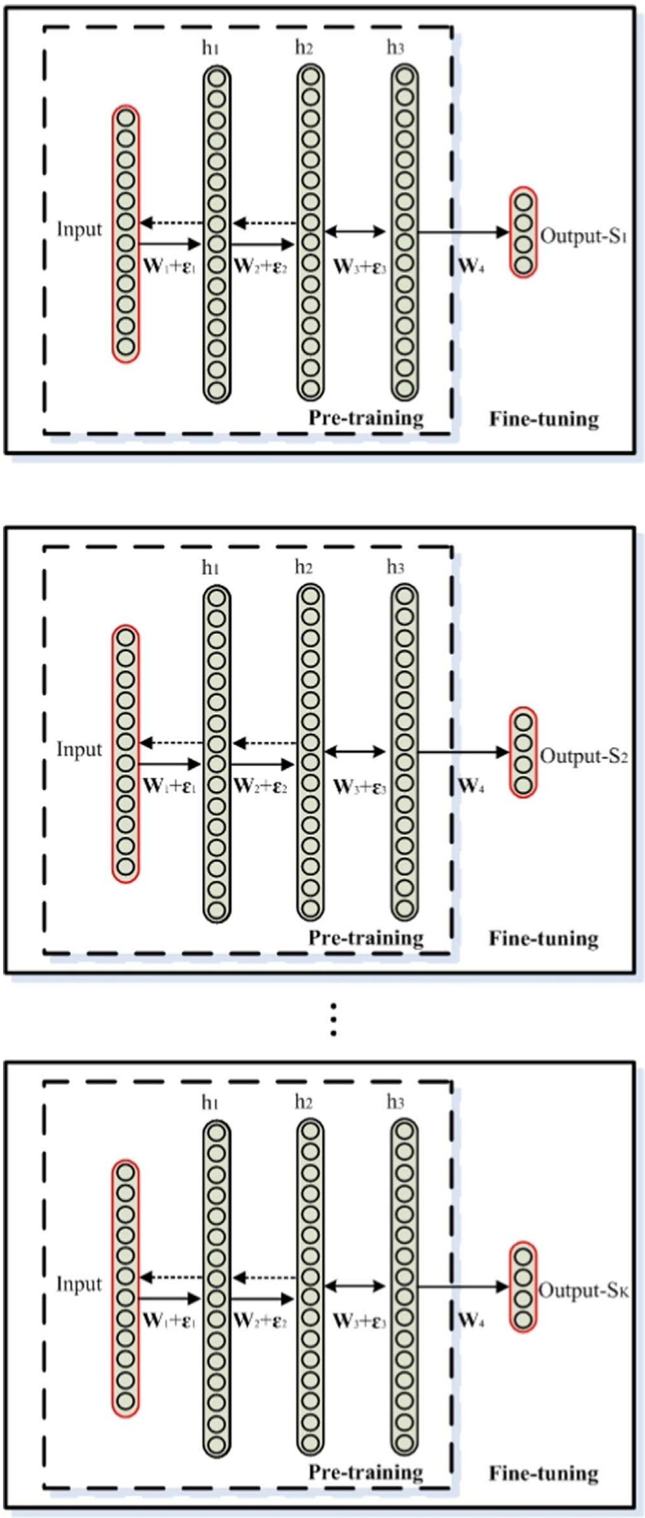

Fig. 2. HDNN for multivariate regression.

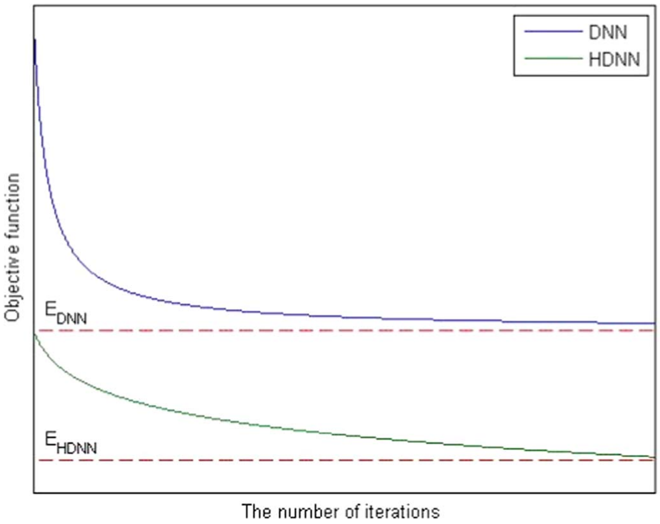

Fig. 3. Illustration of the learning curve for DNN and HDNN.

The main advantage of HDNN is a complicated regression problem using one single DNN can be divided into multiple subproblems which are optimized independently. It should be highlighted that this novel architecture only perfectly works in the regression problem. As in the DNN-based classification problem, the output variables of neural network are the class posterior probabilities which are interrelated and difficult to be decomposed into several independent parts. One question maybe you want to ask is whether the similar effect as HDNN can be achieved if more hidden layers are used in the DNN. At least in the following applications, we demonstrate that HDNN can always achieve much better performance than the DNN with the best configuration. Especially in the handwriting recognition, DNN-based feature transformation even leads to the performance degradation. This might be explained in two aspects. One reason is the use of too many hidden layers in DNN easily leads to over-fitting with the gradient-based optimization method. But in HDNN, the optimization of each subnet is independent and the parameter size of each subnet is smaller than that of one single DNN based regression. The other reason is the independent optimization of each subproblem can potentially make the learning of original regression problem converged to a better optimal. The limitation of the single DNN architecture is that for the reduction of the squared error of each dimension in the output layer, the most of parameters in DNN, namely ($\mathbf{W}_0^{\mathrm{H}}$, $\mathbf{b}_0^{\mathrm{H}}$), are shared to produce the output of the final hidden layer. And ($\mathbf{W}_0^{\mathrm{O}}$, $\mathbf{b}_0^{\mathrm{O}}$) are the only tuneable parameters between the outputs of the final hidden layer and output layer. Meanwhile in HDNN, ($\mathbf{W}_0^{\mathrm{H}}$, $\mathbf{b}_0^{\mathrm{H}}$) are untied in a free style to better serve for the error reduction of each subnet.

3. Applications

3.1. Speech enhancement

Single-channel speech enhancement is a fundamental research problem in many real-world applications, including mobile speech communication, hearing aids design and robust speech recognition [33]. The goal of speech enhancement is to improve the intelligibility and quality of a noisy speech signal degraded in adverse conditions [34]. Recently, the DNN-based approaches become more and more popular for speech enhancement area [25–27]. In this study, we focus on one approach which adopts the DNN as a regression model to find a mapping function between noisy and clean speech signals [26]. We give a system overview of DNN/HDNN based speech enhancement in Fig. 4, which consists of two stages. In the training stage, a DNN/HDNN based regression model is trained using the extracted features from pairs of noisy and clean speech data. As for the feature extraction, the log-power spectra are adopted [35] which offer perceptually relevant parameters. Therefore, short-time Fourier analysis is first applied to the input signal, computing the discrete Fourier transform (DFT) of each overlapping windowed frame. Then the log-power spectra are calculated. In Fig. 5, we demonstrate the input and output of DNN/HDNN by the spectrogram which is a 2-D visual representation of the spectra where the horizontal axis represents the time while the vertical axis is frequency. The input layer of DNN/HDNN is fed with the log-power spectra of noisy speech with the acoustic context information along both time axis (with multiple neighboring frames) and frequency axis (with full frequency bands). Meanwhile the output layer only predicts the log-power spectra of clean speech in the current frame. As the training of this regression DNN/HDNN requires a large amount of time-synchronized stereo-data with clean and noisy speech pairs, which are difficult and expensive to be collected from real scenarios, the noisy speech utterances are synthesized by corrupting the clean speech utterances with additive noises in different types and signal-to-noise ratios (SNRs). In the enhancement stage, the noisy speech features are processed by the well-trained DNN/HDNN model to predict the clean speech features. Then the reconstructed spectrum of the current frame is calculated using the estimated log-power spectra of clean speech and the original noisy phase information. Finally, an overlap-add method is used to synthesize the waveform of the whole utterance [35]..

For HDNN, as the output layer is one frame of log-power spectra for the clean speech with the full frequency band, the subvectors correspond to log-power spectra of the clean speech in the different frequency subbands. This design is quite reasonable for speech enhancement from two aspects. First, the higher sampling rate the speech waveform is with, the more frequency bins should be handled, which makes it more difficult to learn the relationship between the noisy speech and clean speech via a single DNN. Second, there are clear physical meanings of the spectra in the high frequency band and low frequency band. The speech energies are mostly concentrated on the low frequency band while the speech details lie on the high frequency band. It is natural to handle the different frequency bands separately by HDNN. In this study, the full frequency band is equally divided into *K* subbands which correspond to the *K* subnets in HDNN.

3.2. Chinese handwriting recognition

To further demonstrate the effectiveness of HDNN based regression, we apply it to a classification problem, namely Chinese handwriting recognition, rather than a regression problem like speech enhancement. In the mobile internet era, online handwritten Chinese character recognition as an input mode on a portable device has been becoming increasingly popular. Several solutions have been developed to build product engines for online handwritten Chinese character recognition [36,37]. But in real applications, there are many irrelevant variabilities (e.g., writing styles) in character samples, which may lead to the degradation of recognition performance. In this work, we adopt the concept of irrelevant variability normalization (IVN) [38] to tackle the above problem via DNN/HDNN as a highly nonlinear feature transform, rather than the widely used linear transforms [39,40], for online handwritten Chinese character recognition. One of the state-of-

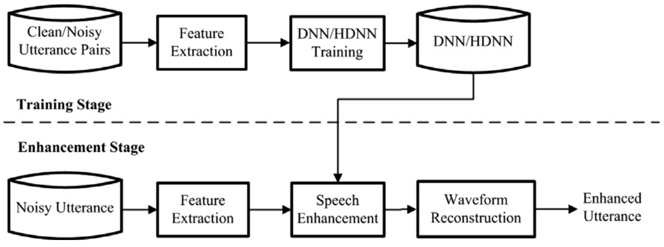

Fig. 4. A block diagram of our speech enhancement system.

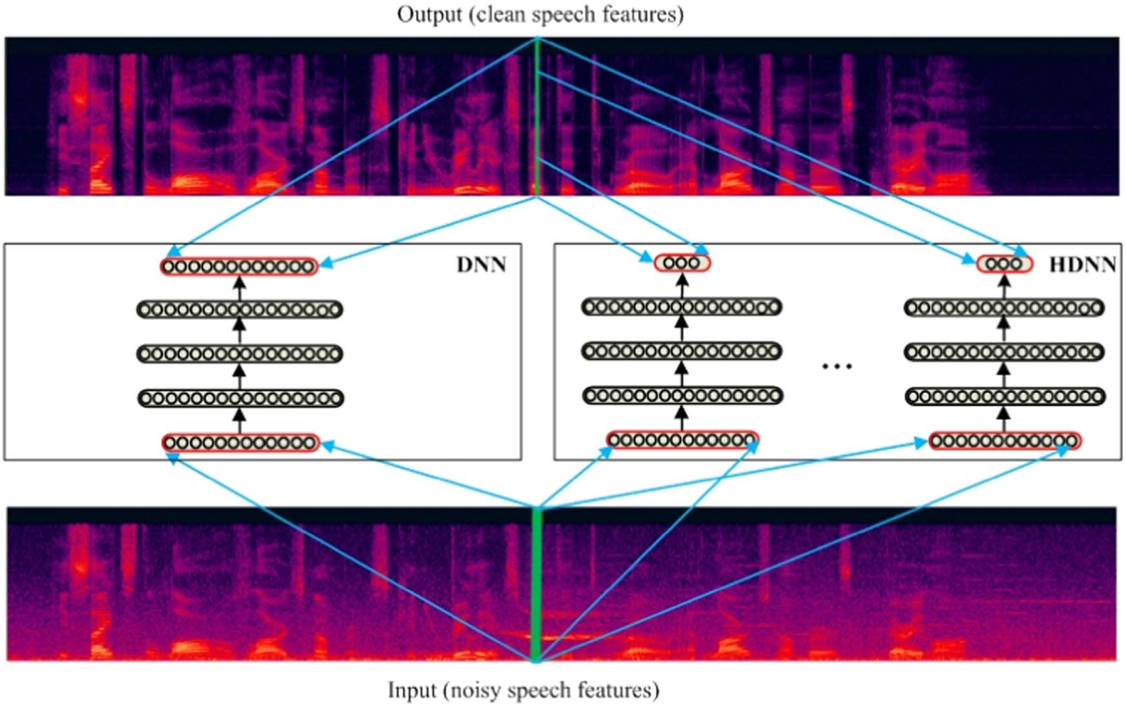

Fig. 5. DNN/HDNN based speech enhancement.

the-art techniques to build a Chinese handwriting recognizer is to use a so-called sample separation margin based minimum classification error (SSM-MCE) criterion [41] to train a prototype-based classifier as reported in [37]. In spite of the large vocabulary of Chinese characters, such a classifier can be made both compact and efficient in the recognition stage [37]. Based on this classifier, we propose to use DNN/HDNN for normalizing the irrelevant variabilities in handwritten samples.

First, an overall system development flow and architecture of IVN via DNN/HDNN is illustrated in Fig. 6. In the training stage, first a raw feature vector is extracted from each training sample [42], which is followed by LDA transformation to obtain a lower dimensional feature vector. After that, the prototype-based classifier is constructed by using LBG clustering algorithm, which can be refined by SSM-MCE training. With prototypes for each character class and feature vectors of training samples, DNN/HDNN training is performed to learn the mapping function between the feature vector of each character sample and its corresponding prototype. Then new transformed features are generated via DNN/HDNN, which are used for the prototype-based classifier training. At the recognition stage, with the feature vector extracted from the unknown sample, feature transform via DNN/HDNN is conducted. Finally the transformed feature vector is fed to recognizer. The details of classifier training and DNN/HDNN based IVN are elaborated as follows..

Suppose our classifier can recognize M character classes denoted as $\{C_i | i = 1,\ldots,M\}$. For a multi-prototype based classifier, each class C_i is represented by K_i prototypes, $\lambda_i = \{\mathbf{m}_{ik} | k = 1,\ldots,K_i\}$, where $\mathbf{m}_{ik}$ is the kth prototype of the ith class with the dimension D. Let's use $\boldsymbol{\Lambda} = \{\lambda_i\}$ to denote the set of prototypes. In the classification stage, a D-dimensional feature vector $\mathbf{z}$ is first extracted. Then $\mathbf{z}$ is compared with each of the M classes by evaluating a Euclidean distance based discriminant function for each class C_i as follows

$$g_i(\mathbf{z};\lambda_i) = -\min_k \left\|\mathbf{z} - \mathbf{m}_{ik}\right\|^2. \tag{9}$$

The class with the maximum discriminant function score is chosen as the recognized class $r(\mathbf{z};\boldsymbol{\Lambda})$, i.e.,

$$r(\mathbf{z};\Lambda) = \arg\max_i g_i(\mathbf{z};\lambda_i). \tag{10}$$

In the training stage, given a set of training feature vectors $\mathcal{Z} = \{\mathbf{z}_r | r = 1,\ldots,R\}$, first we initialize $\boldsymbol{\Lambda}$ by LBG clustering. Then $\boldsymbol{\Lambda}$ can be re-estimated by minimizing the following SSM-MCE objective

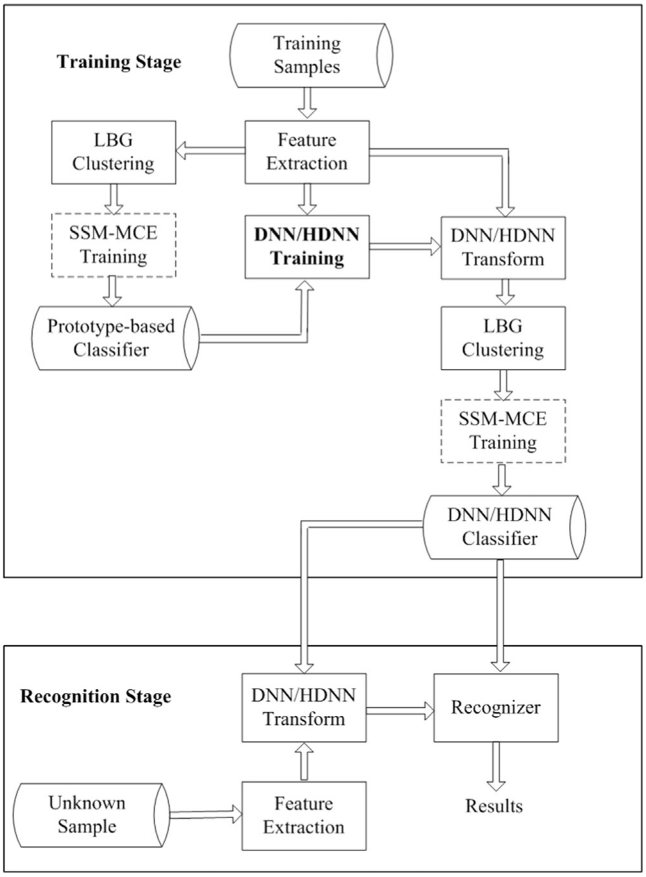

Fig. 6. Overall development flow of Chinese handwriting recognition system.

function:

$$l(\mathcal{Z};\boldsymbol{\Lambda}) = \frac{1}{R}\sum_{r=1}^{R}\frac{1}{1+\exp[-\alpha d(\mathbf{z}_r;\boldsymbol{\Lambda})+\beta]} \tag{11}$$

where α, β are two control parameters, and $d(\mathbf{z}_r;\boldsymbol{\Lambda})$ is a misclassification measure defined by using a so-called sample separation margin (SSM) as follows [41]:

$$d(\mathbf{z}_r;\boldsymbol{\Lambda}) = \frac{-g_p(\mathbf{z}_r;\boldsymbol{\lambda}_p)+g_q(\mathbf{z}_r;\boldsymbol{\lambda}_q)}{2\|\mathbf{m}_{p\hat{k}}-\mathbf{m}_{q\bar{k}}\|} \tag{12}$$

where

$$\hat{k} = \arg\min_{k}\left\|\mathbf{z}_r-\mathbf{m}_{pk}\right\|^2 \tag{13}$$

$$q = \arg\max_{i\in\mathcal{M}_r} g_i(\mathbf{z}_r;\boldsymbol{\lambda}_i) \tag{14}$$

$$\bar{k} = \arg\min_{k}\left\|\mathbf{z}_r-\mathbf{m}_{qk}\right\|^2 \tag{15}$$

and $\mathcal{M}_r$ is the hypothesis space for the rth sample, excluding the true label p. To optimize the objective function in Eq. (11), the same implementation of Quickprop algorithm as described in [43] is adopted.

In this study, the concept of IVN is implemented by using a feature transformation via the Eq. (1), where $\mathbf{x}$ refers to the original feature vector $\mathbf{z}$ while $\mathbf{y}$ represents the transformed feature vector. Then the same MMSE criterion as in Eq. (2) is used for learning the mapping function $\mathcal{F}$. And the reference vector $\mathbf{y}_r$ in Eq. (2) is set as the prototype with the smallest Euclidean distance to $\mathbf{z}_r$ for the corresponding character class. Ideally if the input feature vector can be transformed to the corresponding prototype in that class, then the recognition results are always correct. For the experiments, DNN is directly used for learning the feature transform. By considering the peculiarity of LDA transformed feature vector, we design a specific implementation of HDNN. First, K of HDNN is set to D which implies that each output dimension is associated with a subnet. Then for the final transformed feature vector, only the first D_{sub} ($D_{\text{sub}} < D$) outputs of HDNN is used and the remaining $D - D_{\text{sub}}$ dimensions are set as the same values of the input feature vector. In other words, we only need to train D_{sub} subnets for HDNN. This is inspired by the fact that the most useful information of LDA transformed feature vector lies in the first several dimensions while the remaining dimensions are noisy.

4. Experiments

4.1. Experiments on speech enhancement

The speech enhancement experiments were conducted on TIMIT database [44]. Additive white Gaussian noise (AWGN) and three other types of noise recordings from the Aurora2 database [45], namely Babble, Restaurant and Street, were used as our noise signals for training. All 4620 utterances from the training set of the TIMIT database were added with the abovementioned four types of noise and six levels of SNR from −5 to 20 dB with an increment of 5 dB, to build a multi-condition stereo training set. This resulted in a collection of about 100 h of noisy training data (including one case of clean training data) used to train the DNN/HDNN based speech enhancement models. As for the test set, another 200 randomly selected utterances from the TIMIT test set were used for each combination of noise types and SNR levels. Two other unseen noise types, namely Car and Exhibition, were adopted for mismatch evaluation.

As for signal analysis, speech waveform was down-sampled to 8 kHz. The frame length was set to 256 samples (or 32 ms) with a frame shift of 128 samples. The 129-dimensional log-power spectra features were used to train DNN/HDNN. The standard DNN architecture used in the experiments was 1419-2048-2048-2048-129, which denoted that the sizes were 1419 (129*11) for the input layer with a 11-frame context, 2048 for three hidden layers, and 129 for the output layer. For HDNN, three hidden layers were also used for each subnet. K=2 and K=4 were compared for the division of the output layer. For K=2, the full frequency band (129-dimensional output vector) was divided into two subbands with the indices of frequency bins 1–65 and 66–129. This represented the architectures of the two subnets were 1419-2048-2048-2048-65 and 1419-2048-2048-2048-64. For K=4, the four subband indices were 1–32, 33–64, 65–96, and 97–129. The number of epoch for each layer of RBM pre-training was 20. Learning rate of pre-training was 0.0005. As for the fine-tuning, learning rate was set at 0.1 for the first 10 epochs, then decreased by 10% after every epoch. Total number of epoch was 50. The mini-batch size was set to 128. Input and output features of DNN/HDNN were normalized to zero mean and unit variance. The objective quality measure, namely perceptual evaluation of speech quality (PESQ), which has a high correlation with subjective score [46], was used for performance comparison.

4.1.1. Comparison among DNNs with different configurations

First, PESQ comparison of DNN systems with different configurations on the test set across four noise conditions at different SNR levels is listed in Table 2. LogMMSE [30,31] was used as one conventional speech enhancement approach for comparison. The configurations of DNN_1 to DNN_5 are given in Table 1 with different number of hidden layers and hidden nodes. All the DNN systems were trained using the same 100-h data. Obviously, both DNN_1 and DNN_2 were shallow neural networks with only one hidden layer. We observed that all DNN approaches significantly outperformed LogMMSE approach indicating that the DNNs were capable of making more accurate estimation of the target speech corrupted by noise. For the shallow neural network, the increase of hidden units from 2048 (DNN_2) to 6144 (DNN_1) could improve the PESQ performance. The DNNs with more hidden layers (no more than 3 hidden layers) were demonstrated more effective and the DNN_4 achieved the best performance. The improvement of PESQ measures over the DNN_1 which have the same number of hidden nodes with DNN_4 demonstrated that deeper architectures had a much stronger regression capability. In the following experiments, the DNN_4 is used as default.

4.1.2. Comparison between DNN and other regression model

To further demonstrate the powerful modeling capability of the DNN-based multivariate regression, the comparison with another conventional regression model, namely support vector regression (SVR) [47], is investigated. We adopted a publicly available SVR tool [48] to conduct the experiments. Based on a set of preliminary experiments, several limitations of SVR can be summarized as: a) SVR can not well handle the large scale training data due to the computational complexity and memory requirement; b) with a high dimension (129 in our experiment) for both input and output variables, the generalization capability of SVR to other testing cases is very poor. We found in our experiments, if the clean speech utterances or noise types were unseen in the test set, SVR tool could not even generate the meaningful results. By considering these limitations, only the well-match training-testing conditions, namely both the clean speech and noise should be seen from the training set, could be adopted for comparison between SVR and DNN, as shown in Table 3. 120 utterance

Table 1
The configurations for different DNN systems.

Configuration	DNN_1	DNN_2	DNN_3	DNN_4	DNN_5
# of hidden layers	1	1	2	3	4
# of hidden nodes	6144	2048	2048	2048	2048

Table 2
PESQ comparison of DNN systems with different configurations on the test set across four noise conditions (AWGN, Babble, Restaurant and Street) at different SNR levels.

	SNR20	SNR15	SNR10	SNR5	SNR0	SNR-5	Avg.
Noisy	2.99	2.65	2.32	1.98	1.65	1.38	2.16
LogMMSE	3.32	2.99	2.65	2.30	1.93	1.55	2.46
DNN_1	3.48	3.26	2.99	2.68	2.32	1.92	2.78
DNN_2	3.46	3.24	2.97	2.65	2.29	1.89	2.75
DNN_3	3.59	3.35	3.08	2.76	2.38	1.95	2.85
DNN_4	3.60	3.36	3.10	2.78	2.41	1.97	2.87
DNN_5	3.59	3.36	3.09	2.78	2.41	1.97	2.87

Table 3
PESQ comparison of SVR and DNN based systems under the well-match training-testing condition.

		SNR10	SNR0	SNR-5
AWGN	Noisy	2.08	1.45	1.23
	SVR	3.76	3.57	3.44
	DNN	3.76	3.53	3.36
Car	Noisy	2.43	1.78	1.49
	SVR	3.80	3.55	3.41
	DNN	3.80	3.57	3.38
Exhibition	Noisy	2.22	1.59	1.28
	SVR	3.76	3.53	3.38
	DNN	3.77	3.52	3.35

Table 4
PESQ comparison of DNN and HDNN based systems on the test set under three noise conditions at different SNR levels.

		SNR20	SNR15	SNR10	SNR5	SNR0	SNR-5	Avg.
AWGN (Seen)	DNN	3.55	3.32	3.06	2.78	2.46	2.08	2.87
	HDNN (*K*=2)	3.61	3.38	3.14	2.87	2.57	2.22	2.96
	HDNN (*K*=4)	3.62	3.41	3.18	2.93	2.64	2.32	3.02
	HDNN-C(*K*=2)	3.59	3.36	3.12	2.86	2.56	2.21	2.95
Car (Unseen)	DNN	3.58	3.31	3.03	2.71	2.35	1.96	2.83
	HDNN (*K*=2)	3.60	3.35	3.07	2.77	2.42	2.06	2.88
	HDNN (*K*=4)	3.62	3.38	3.10	2.79	2.45	2.10	2.91
	HDNN-C(*K*=2)	3.59	3.34	3.06	2.76	2.42	2.05	2.87
Exhibition (Unseen)	DNN	3.30	3.01	2.69	2.33	1.93	1.54	2.47
	HDNN (*K*=2)	3.37	3.10	2.80	2.45	2.07	1.68	2.58
	HDNN (*K*=4)	3.37	3.11	2.82	2.49	2.15	1.80	2.62
	HDNN-C(*K*=2)	3.34	3.08	2.79	2.45	2.09	1.71	2.58

Table 5
The computational complexity of different approaches.

	Model size (in MB)	Training data (in hour)	Training time (in hour)	Test data (in hour)	Test time (in second)
DNN	44	100	52	3.9	116
HDNN(*K*=2)	87	100	117	3.9	230
HDNN(*K*=4)	173	100	182	3.9	457
HDNN-C(*K*=2)	28	100	30	3.9	74

Table 6
Performance (character error rate in %) comparison of different systems using prototype-based classifiers with LBG clustering on the test set.

Methods	Baseline	DNN-1L	DNN-2L	DNN-3L	HDNN-1L	HDNN-2L
CER(%)	16.13	29.26	23.30	25.63	13.44	12.37

Table 7
Performance (character error rate in %) comparison of systems using prototype-based classifiers with different features and training criteria on the test set.

	# prototype	LBG	SSM-MCE
Baseline	1	16.13	12.26
	4	13.68	11.64
HDNN (LBG)	1	12.37	11.64
	4	11.84	11.32
HDNN (SSM-MCE)	1	11.38	10.82
	4	10.96	10.61

pairs of noisy and clean speech, covering the three noise conditions with a specific SNR level, were used to train each SNR-dependent model for both SVR and DNN approaches. The amount of such data set already made the use of system memory close to the maximum during the SVR training. Obviously, DNN approach could generate comparable PESQ performances with SVR approach and both significantly outperformed the unprocessed system (Noisy) under the well-match training-testing condition. However, the SVR approach failed to generalize well to the unseen cases. In this sense, the DNN-based regression is inherently more powerful in handling the large scale training data with high-dimensional input and output, which has the good generalization capability demonstrated by the following experiments.

4.1.3. Comparison between DNN and HDNN

Table 4 shows the PESQ results among DNN and HDNN with different settings on the test set under three noise conditions at different SNR levels. Note that the best configuration was used for DNN. HDNN(*K*=2) and HDNN(*K*=4) were the two HDNN-based speech enhancement systems using the same 2048 nodes for the hidden layer of each subnet as the DNN system with *K*=2 and *K*=4, respectively. Obviously the model sizes of HDNN(*K*=2) and HDNN(*K*=4) were bigger than that of DNN although using more hidden nodes or layers in DNN could not further improve the performance. To make it a more fair comparison, we designed a compact model using 1024 nodes for the hidden layer of each subnet in HDNN with *K*=2, denoted as HDNN-C(*K*=2), with a much smaller size than DNN. From those results, several observations could be made. First, all the HDNN-based systems consistently outperformed DNN-based system for different noise types and levels, especially in the case of low SNR levels, e.g., an increment of 0.26 PESQ was achieved under −5 dB SNR of Exhibition noise from DNN to HDNN(*K*=4). Second, the performance gains of HDNN over DNN on the unseen noise types, e.g., an average PESQ gain of 0.15 from 2.47 in DNN to 2.62 in HDNN(*K*=4) under Exhibition condition, were comparable to those on the seen noise types, e.g., an average PESQ gain of 0.15 from 2.87 in DNN to 3.02 in HDNN(*K*=4) under AWGN condition. This indicated the good generalization capability of HDNN for the unseen noise conditions. Third, the higher resolution HDNN with more subnets could generate better PESQ performances. For example, from *K*=2 to *K*=4 for HDNN under Car noise, the PESQ gains of 0.02 and 0.04 were yielded at 20 dB SNR and −5 dB SNR, respectively. Finally, by the comparison of HDNN(*K*=2) and HDNN-C(*K*=2), it seemed that using less parameters would not lead to significant performance degradation, which was reasonable as the subproblem represented by each subnet was inherently simpler than the original regression problem. Overall, the HDNN-C(K=2) based system achieved better performance but more compact than a single DNN based system. This implied that by decomposing a complicated regression problem into multiple subproblems and solving them via HDNN, we could achieve both effectiveness and compactness.

4.1.4. Computational complexity

Table 5 gives the computational complexity of four approaches as in Table 4. Our experiments were conducted on a machine using Intel Xeon E5-2620 CPU with a clock rate of 2.1 GHz. The training of DNN and HDNN was accelerated by the Tesla K20 GPU. Although the model size and training/test time of HDNN(K=2) and HDNN(K=4) are much larger than those of DNN, HDNN-C(K=2) which can significantly outperform DNN in Table 4 has a smaller model size and runtime latency. This implies the superiority of HDNN architecture design compared with DNN.

4.2. Experiments on Chinese handwriting recognition

The experiments were conducted on the task of recognizing isolated online handwritten Chinese characters with a vocabulary of 3926 character classes via a public database released by the Institute of Automation of Chinese Academy of Sciences (CASIA) [49]. There were 939561 samples in the training set and 234798 samples in the test set. For feature extraction, a 512-dimensional raw feature vector was extracted as described in [42], which was followed by LDA transformation to obtain a 128-dimensional feature vector. For Quickprop-based SSM-MCE training, the setting of the control parameters could refer to [43]. The number of prototype for each character class of the classifier used for DNN/HDNN training was set to 1. D_{sub} was set as 48. The tuning parameters of DNN were set according to [50]. The number of units in each hidden layer of DNN/HDNN was 1024. To handle the large-scale training data, the computations of LBG clustering, SSM-MCE training with Quickprop algorithm were parallelized on the CPU cluster while DNN/HDNN training was implemented and optimized on GPUs.

Table 6 shows a performance (character error rate in %) comparison of different systems using prototype-based classifiers with LBG clustering on the test set. Baseline denoted the system without using IVN. DNN-1L to DNN-3L represented the systems using DNN-based IVN with 1 hidden layer to 3 hidden layers, respectively. HDNN-1L and HDNN-2L referred to the systems using HDNN-based IVN with 1 hidden layer and 2 hidden layers for each subnet, respectively. Note that the prototype-based classifiers for both IVN and classification in Fig. 6 are generated using LBG clustering with 1 prototype. First, DNN-based IVN systems yielded much worse performance over the baseline system, which indicated that DNN totally failed in learning the mapping function between the LDA transformed feature vector and the corresponding prototype even using deep architectures. Second, HDNN-based IVN systems achieved significant error reductions over the baseline system. In terms of recognition error rate, HDNN showed much more powerful learning capability than DNN. Furthermore, HDNN-2L system gave the best recognition performance with deeper architecture than HDNN-1L system.

Table 7 gives a performance (character error rate in %) comparison of systems using prototype-based classifiers with different features and training criteria on the test set. HDNN(LBG) and HDNN(SSM-MCE) denoted HDNN-based IVN systems using two prototype-based classifiers with LBG clustering and SSM-MCE training for HDNN training, respectively. 2 hidden layers were used in each subnet of HDNN. Several observations can be made. First, all the HDNN-based IVN systems yielded consistently significant performance gain over baseline system in the corresponding setting of different number of prototypes and training criteria, especially for prototype-based classifier trained using LBG clustering. Second, without SSM-MCE based discriminative training, namely, the system using prototype-based classifier trained by LBG clustering in the case of HDNN(LBG) could still generate comparable recognition accuracy with SSM-MCE trained baseline system, which implied that the feature space after transformation via HDNN-based IVN brought more discriminative information. Third, the HDNN-based IVN system using SSM-MCE generated prototypes for HDNN training could always outperform the system using LBG generated prototypes. Finally, the best HDNN-based IVN system achieved about absolute 1% error reduction over the best baseline system under the setting of 4 prototypes and SSM-MCE training (from 11.64% to 10.61%).

5. Discussion and conclusion

Theoretically, with a huge amount of diversified training data, the DNN with adequate hidden layers can be adopted for approximating an arbitrary regression problem. However, in real applications, both the gradient-based learning method and the amount of training data make the DNN not optimal especially for the regression problem with high-dimensional input and output. Motivated by this, in this study we proposes a novel architecture of neural network, namely HDNN. HDNN could be perfectly applied for the general multivariate regression problem, which focuses on both “depth” and “width” of artificial neural network. HDNN adopts the divide and conquer strategy to decompose a complicated multivariate regression problem into multiple subproblems for learning the parameters, which can empirically achieve better learning convergence than DNN. Two applications, namely speech enhancement and Chinese handwriting recognition, demonstrate the effectiveness of HDNN. Both empirical analysis and experiment results demonstrate that HDNN is more powerful than DNN for the regression problems.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants No. 61671422, National Key Technology Support Program under Grants No. 2014BAK15B05, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB02070006, MOE-Microsoft Key Laboratory of USTC.

References

[1] F. Galton, Regression towards mediocrity in hereditary stature, J. Anthropol. Inst. Gt. Br. Irel. 15 (1886) 246–263.
[2] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[3] K. Pearson, Mathematical contributions to the theory of evolution. III. regression, heredity and panmixia, Philos. Trans. R. Soc. Lond. 187 (1896) 253–318.
[4] K. Smith, On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of the observations, Biometrika 12 (1918) 1–85.
[5] J. Berkson, Application of the logistic function to bio-assay, J. Am. Stat. Assoc. 39 (1944) 357–365.
[6] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Adv. Neural Inf. Process. Syst. (1997) 155–161.
[7] V. Vapnik, S. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, Adv. Neural Inf. Process. Syst. (1997) 281–287.
[8] K. Pearson, On the generalized probable error in multiple normal correlation, Biometrika 6 (1908) 59–68.
[9] T.W. Anderson, An Introduction to Multivariate Analysis, Wiley, New York, 1958.
[10] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[11] D.F. Specht, A general regression neural network, IEEE Trans. Neural Netw. 2 (6) (1991) 568–576.
[12] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990) 4–27.
[13] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[14] D. Erhan, A. Courville, Y. Bengio, P. Vincent, Why does unsupervised pre-training help deep learning? The Journal of Machine Learning Research, No. 11, 2010, pp. 625–660.

[15] G.E. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.

[16] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[17] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 5 (2007) 153–160.

[18] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.

[19] L. Yann, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 11 (1986) 2278–2324.

[20] O. Russakovsky, et al., ImageNet large scale visual recognition challenge, 2014.

[21] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber, A novel connectionist system for improved unconstrained handwriting recognition, IEEE Trans. Pattern Anal. Mach. Intell. 31 (5) (2009) 855–868.

[22] A.L. Maas, Q.V. Le, T.M. O′Neil, O. Vinyals, P. Nguyen, A.Y. Ng, Recurrent neural networks for noise reduction in robust ASR, Proceedings Interspeech, 2012, pp. 22–25.

[23] M. Wollmer, Z. Zhang, F. Weninger, B. Schuller, G. Rigoll, Feature enhancement by bidirectional LSTM networks for conversational speech recognition in highly nonstationary noise, Proceedings ICASSP, 2013, pp. 6822–6826.

[24] L.-H. Chen, Z.-H. Ling, L.-J. Liu, L.-R. Dai, Voice conversion using deep neural networks with layer-wise generative training, IEEE/ACM Trans. Audio, Speech Lang. Process. 22 (12) (2014) 1859–1872.

[25] Y.X. Wang, D.L. Wang, Towards scaling up classification-based speech separation, IEEE Trans. Audio, Speech Lang. Process. 21 (7) (2013) 1381–1390.

[26] Y. Xu, J. Du, L.-R. Dai, C.-H. Lee, An experimental study on speech enhancement based on deep neural networks, IEEE Signal Process. Lett. 21 (1) (2014) 65–68.

[27] Y. Xu, J. Du, L.-R. Dai, C.-H. Lee, A regression approach to speech enhancement based on deep neural networks, IEEE/ACM Trans. Audio, Speech Lang. Process. 23 (1) (2015) 7–19.

[28] Y.-H. Tu, J. Du, Y. Xu, L.-R. Dai, C.-H. Lee, Speech separation based on improved deep neural networks with dual outputs of speech features for both target and interfering speakers, Proceedings ISCSLP, 2014, pp. 250–254.

[29] J. Du, Y.-H. Tu, Y. Xu, L.-R. Dai, C.-H. Lee, Speech separation of a target speaker based on deep neural networks, Proceedings ICSP, 2014, pp. 473–477.

[30] I. Cohen, B. Berdugo, Speech enhancement for nonstationary noise environments, Signal Process. 81 (11) (2001) 2403–2418.

[31] I. Cohen, Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging, IEEE Trans. Speech Audio Process. 11 (5) (2003) 466–475.

[32] J. Du, Irrelevant variability normalization via hierarchical deep neural networks for online handwritten Chinese character recognition, Proceedings ICFHR, 2014, pp. 303–308.

[33] P.C. Loizou, Speech Enhancement: theory and Practice, CRC Press, 2013.

[34] J. Benesty, S. Makino, J.D. Chen, Speech Enhancement, Springer, 2005.

[35] J. Du, Q. Huo, A speech enhancement approach using piecewise linear approximation of an explicit model of environmental distortions, Proceedings INTERSPEECH, 2008, pp. 569–572.

[36] C.-L. Liu, S. Jaeger, M. Nakagawa, Online recognition of Chinese characters: the state-of-the-art, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 198–213.

[37] J. Du, Q. Huo, Designing compact classifiers for rotation-free recognition of large vocabulary online handwritten Chinese characters, Proceedings ICASSP, 2012, pp. 1721–1724.

[38] Q. Huo, B. Ma, Irrelevant variability normalization in learning HMM state tying from data based on phonetic decision-tree, Proceedings ICASSP, 1999, pp. 577–580.

[39] X.-Y. Zhang, K. Huang, C.-L. Liu, Pattern field classification with style normalized transformation, Proceedings IJCAI, 2011, pp. 1621–1626.

[40] J. Du, Q. Huo, An irrelevant variability normalization approach to discriminative training of multi-prototype based classifiers and its applications for online handwritten Chinese character recognition, Pattern Recognit. 47 (12) (2014) 3959–3966.

[41] T. He, Q. Huo, A study of a new misclassification measure for minimum classification error training of prototype-based pattern classifiers, Proceedings ICPR, 2008.

[42] Z.-L. Bai, Q. Huo, A study on the use of 8-directional features for online handwritten Chinese character recognition, Proceedings ICDAR, 2005, pp. 262–266.

[43] Y.-Q. Wang, Q. Huo, A study of designing compact recognizers of handwritten Chinese characters using multiple-prototype based classifiers, Proceedings ICPR, 2010, pp. 1872–1875.

[44] J.S. Garofolo, Getting started with the DARPA TIMIT CD-ROM: an acoustic phonetic continuous speech database, NIST Tech Report, 1988.

[45] H.G. Hirsch, D. Pearce, The AURORA experimental framework for the preformance evaluations of speech recognition systems under noisy conditions, Proceedings ISCA ITRW ASR, 2000, pp. 181–188.

[46] ITU-T, Recommendation P.862, Perceptual evaluation of speech quality (PESQ): an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs, International Telecommunication Union-Telecommunication Standardisation Sector, 2001.

[47] M. Sanchez-Fernandez, M. dePrado-Cumplido, J. Arenas-Garcia, F. Perez-Cruz, SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems, IEEE Trans. Signal Process. 52 (8) (2004) 2298–2307.

[48] ⟨https://github.com/wjb19/mimo-svr⟩

[49] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang, Online and offline handwritten Chinese character recognition: benchmarking on new databases, Pattern Recognit. 46 (1) (2013) 155–162.

[50] G. Hinton, A practical guide to training restricted Boltzmann machines, Technical Report, University of Toronto, 2010.

Jun Du received the B.Eng. and Ph.D. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC) in 2004 and 2009, respectively. From 2004 to 2009, he was with iFlytek Speech Lab of USTC. During the above period, he worked as an Intern twice for 9 months at Microsoft Research Asia (MSRA), Beijing. In 2007, he also worked as a Research Assistant for 6 months in the Department of Computer Science, The University of Hong Kong. From July 2009 to June 2010, he worked at iFlytek Research on speech recognition. From July 2010 to January 2013, he joined MSRA as an Associate Researcher, working on handwriting recognition, OCR, and speech recognition. Since February 2013, he has been with the National Engineering Laboratory for Speech and Language Information Processing (NEL-SLIP) of USTC.

Yong Xu received the B.S. degree in communication engineering from Anhui University in 2010. He is currently a Ph.D. candidate of University of Science and Technology of China (USTC). From July 2012 to December 2012, he was an intern at iFlytek. From September 2014 to April 2015, he is a visiting student at Georgia Institute of Technology, USA. His current research interests include deep learning for speech enhancement and noise robust speech recognition.