Neural Comput & Applic (2017) 28 (Suppl 1):S1177–S1196
DOI 10.1007/s00521-016-2438-x

ORIGINAL ARTICLE

Toward growing modular deep neural networks for continuous speech recognition

Zohreh Ansari[1] · **Seyyed Ali Seyyedsalehi**[1]

Received: 18 December 2014 / Accepted: 14 June 2016 / Published online: 22 June 2016

Abstract The performance drop of typical automatic speech recognition systems in real applications is related to their not properly designed structure and training procedure. In this article, a growing modular deep neural network (MDNN) for speech recognition is introduced. According to its structure, this network is pre-trained in a special manner. The ability of the MDNN to grow enables it to implement spatiotemporal information of the frame sequences at the input and their labels at the output layer at the same time. The trained network with such a double spatiotemporal (DST) structure has learned valid phonetic sequences subspace. Therefore, it can filter out invalid output sequences in its own structure. In order to improve the proposed network performance in speaker variations, two speaker adaptation methods are also presented in this work. In these adaptation methods, the network trains how to move distorted input representations nonlinearly to their optimal positions or to adapt itself based on the input information. To evaluate the proposed MDNN structure and its modified versions, two Persian speech datasets are used: FARSDAT and Large FARSDAT. As there is no frame-level transcription for large vocabulary speech datasets, a semi-supervised learning algorithm is explored to train MDNN on Large FARSDAT. Experimental results on FARSDAT verify that implementing the DST structure besides speaker adaptation methods achieves up to 7.3 and 10.6 % absolute phone accuracy rate improvement over the MDNN and typical hidden Markov model, respectively. Likewise, semi-supervised training of the grown MDNN on Large FARSDAT improves its recognition performance up to 5 %.

Keywords Deep neural networks · Modular neural networks · Pre-training · Nonlinear filtering · Double spatiotemporal · Speaker adaptation · Continuous speech recognition

✉ Seyyed Ali Seyyedsalehi
ssalehi@aut.ac.ir

[1] Speech Processing Lab., Faculty of Biomedical Engineering, Amirkabir University of Technology (Tehran Polytechnic), Hafez Ave., Tehran, Iran

1 Introduction

Many successful commercial automatic speech recognition (ASR) systems have been developed over the past few decades. However, they cannot handle a wide range of variability as humans do [1–3]. Therefore, there is an open research area to develop speech recognition systems that operate well in speech variations.

Most current ASR systems use hidden Markov models (HMMs) to model the sequence structure of speech representations into speech units. In each HMM state, Gaussian mixture models (GMMs) are used to model spectral representations of the speech signal. Some advances have been proposed during the years to improve the recognition accuracy of GMM-HMM systems [4–6]. Two decades ago, some developments were achieved using artificial neural networks (ANNs) with one or two nonlinear hidden layers and hybrid ANN-HMM models in speech recognition [7–12]. However, neither the high-speed hardware products nor the learning methods for training deep neural networks existed. Therefore, the performance of ANNs was kept limited. Subsequently, HMMs were implemented more than ANNs for acoustic modeling.

Recently, progresses in learning methods explored for ANNs with more than two hidden layers and in high-speed hardware products have led to implementing deep structures in speech recognition. Using deep neural network (DNN) models for speech recognition on many different datasets has shown that DNNs can outperform GMMs in hybrid structures with HMMs [3, 13–17].

In this paper, a growing modular deep neural network (MDNN) is proposed. The main novelty of this network is that it can be grown to learn the sequential structure of the speech signal. Moreover, two innovative speaker adaptation methods are proposed to improve this network performance in speaker variability. In this paper, we illustrate the ingredients of the growing MDNN, describe the proposed learning procedure for it, investigate the proposed adaptation methods, and analyze how different design choices affect the network recognition performance. To the best of our knowledge, the proposed MDNN is the first DNN that can be trained to model both short time representations and temporal variability of the speech signal in an integrative manner.

1.1 Previous works on using DNNs in speech recognition

Hybrid structures of DNNs with graphical models like HMMs or conditional random fields (CRFs) are widely used for phone recognition [15, 17–20]. In these models, as it is shown in Fig. 1, the output of the trained static DNN on a moving window of sequential frames is combined with the graphical models. Graphical models are used to deal with linear dependencies associated with the DNN outputs. In fact, the variations along the time axis are handled by a HMM or CRF. In such speech recognition systems, DNNs are implemented as acoustic models and the graphical models are used as phonetic models that are trained separately.

DNNs used as acoustic models typically contain many nonlinear hidden layers with the same number of neurons in each layer and a nonlinear output layer with many neurons. The large output layer is considered to accommodate the large number of HMM states. Therefore, there are many parameters to be trained. Although layer-wise pre-training methods have been proposed to make the training of these networks feasible [21, 22], training of each layer is time consuming and requires a massive memory. Recently, convolutional neural networks (CNNs) have been investigated to reduce DNN parameters by using replicated weights across time and frequency dimensions of speech signal [23]. However, experimental results have shown that CNNs approach their best performance when their number of parameters matches the DNN parameters [24].

Speech recognition is a sequential learning program, and discriminative information at sequence level improves recognition accuracy [25]. Therefore, many algorithms have been explored lately to train sequential information to DNNs [25–29]. However, all of the proposed algorithms are applicable for hybrid structures of DNNs. None of the proposed methods considers a unified DNN structure that extracts phonetic sequence as well as acoustic information from the training data. If such a unified neural network could be designed and trained effectively, both acoustic and phonetic information would be able to interact with each other to access the best recognition results.

DNN-based ASR systems like other statistical techniques may fail to show the same level of performance in mismatched test conditions. Thus, performing effective speaker adaptation for DNNs has become very important. Speaker adaptation methods adapt either the model or acoustic feature vectors to reduce the mismatches between training and test conditions. Several speaker adaptation methods have been proposed for ANNs. A comprehensive review of these methods can be found in [30]. Here, those techniques that motivated the work conducted in this article are reviewed. Some of the model adaptation methods modify the ANN structure, e.g., by adding some layers to it or by feeding it with speaker data [31–33]. In such cases, during adaptation, only new parameters are calculated using adaptation data. However, these methods cannot be used in conditions that no adaptation data exist. The presented speaker adaptation method in [34] modifies the ANN structure by adding speaker information to its first hidden layer. Although this method does not require any adaptation data, it has been tested on a very small dataset. The other group of methods transforms the input feature space. These methods such as vocal tract length normalization (VTLN) and feature space maximum likelihood linear regression (fMLLR) can be directly used to normalize input acoustic features prior to DNNs [35, 36]. Although indicated feature space transformations improve the recognition performance of DNNs, they are linear and cannot deal with nonlinear variations of the speech signal. In the nonlinear feature normalization method proposed in [37], a feed-forward neural network is first trained to map the input representations into both phonetic and speaker codes. Then, a training speaker with the highest phone accuracy is considered as the reference speaker. Subsequently, input features are normalized using the error back-propagation algorithm to be adapted to the reference speaker speech.

1.2 Introducing MDNN for speech recognition

In this paper, in order to step forward to achieve efficient DNN architectures for speech recognition, which can do both acoustic and phonetic modeling, a growing MDNN is proposed. The initial structure of this network has been explored in [38]. This network implements some concepts

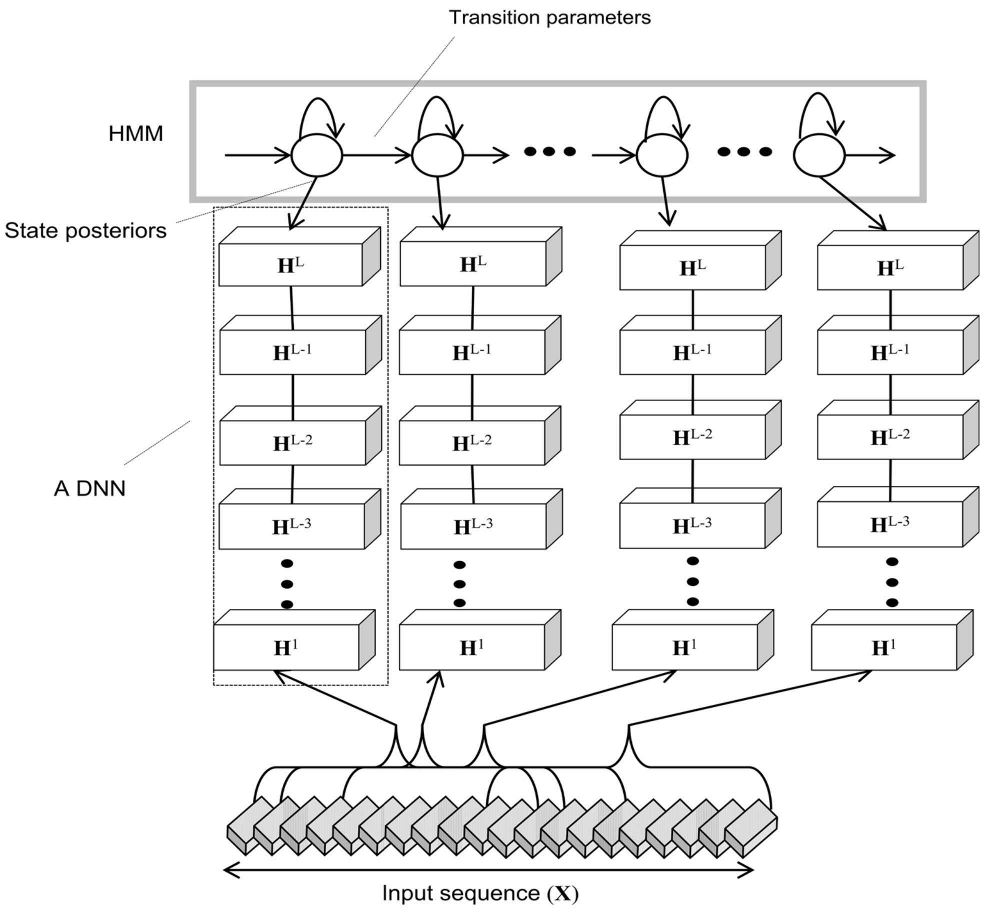


Fig. 1 Hybrid DNN-HMM model used in most previous works applying DNNs to speech recognition tasks. In this model, the output of the DNN over a sliding window on speech frames is combined with HMM to model their linear dependencies

developed in DNNs and modular neural networks. It consists of a deep structure with a series of feature detector units in each layer. Each feature detector is a filter applied on a small local part of the previous layer nodes to process the dynamic information of that part. However, in the first hidden layer, dynamic information of the input is not considered. In this layer, each input representation is projected to the corresponding higher dimensional feature detector to be processed further. Lower layers detect simple features and feed them into higher layers which in turn detect more complex features. Sharing the same weights among all the filters applied on one layer not only decreases the training parameters, but also makes the filters robust to inaccurate phonetic transcribed speech data.

The proposed MDNN is hard to optimize. Therefore, it must initially be pre-trained to be located in a much better starting point for the global fine-tuning phase. In this article, a proportional pre-training method for the presented network is introduced. In this method, the network is trained layer-wisely. Due to the modular structure of this network and shared weights between the modules of one layer, only one module of each layer is trained. Consequently, the trained weights are replicated for other similar modules.

One of the key characteristics of the MDNN is that it can be grown. When it is required to train the network with more complex information (like phonetic sequence information or word information) than it was trained before, it is grown. Growing of the network enables it to consider the spatiotemporal sequence of acoustic frames in its input and the spatiotemporal sequence of their corresponding phonetic labels in the output layer, at the same time. The grown structure in this way is a MDNN with double spatiotemporal (DST) structure. By training the MDNN with DST structure properly, it learns the valid phonetic sequence subspace of frames. Thus, it becomes able to filter out invalid output sequences in its own structure.

It is possible to use the MDNN expanded to DST structure as one module of a DNN-based large vocabulary continuous speech recognition (LVCSR) system. However, it must be tuned on the large vocabulary dataset at first. In all usages of DNNs in combination with HMMs for LVCSR tasks, they are initially tuned on large datasets [3, 10, 17, 39]. Since there is no frame-level transcription for large datasets, baseline GMM-HMMs trained on those datasets are used to obtain the required transcriptions through forced alignment. Thus, a trained GMM-HMM model on each large dataset must exist. In this paper, a semi-supervised learning algorithm for tuning the neural networks on large datasets is introduced. This algorithm uses only available information and does not require any other trained model on the large datasets.

It has been shown that the robust performance of the human perception system relates to the special mode of signal processing in the brain. Some of the characteristics of this mode of processing are: (1) handling the information in both forward and backward passes, (2) the ability of analyzing linear and nonlinear components of its input, and (3) omitting the effect of each component if necessary [37, 40, 41]. According to this information, two speaker adaptation methods are proposed to approach a MDNN that can handle speaker variability. These methods are as follows:

1. Nonlinear feature normalization method. In this method, input features are normalized iteratively to be located in a proper position with respect to the speech recognition model. To this end, the speech recognizer must be trained on how to move distorted inputs toward the best positions at first. After each training epoch, feature normalization is performed based on the information extracted from the current trained network.
2. Model adaptation method. The basic idea of this method is to apply speaker information extracted by a separate speaker recognition network to the MDNN layers. The speaker recognition network has been trained to recognize each speaker based on its acoustic features. The connection weights from speaker to speech recognition network must be trained using whole training data. In the test phase, the speaker recognition network yields the similarity between the acoustic spaces of the test speaker with each of the training speakers. Then, this information is fed to the speech recognition model to produce nonlinear transformation in the model space. This method does not require any adaptation data. Therefore, it can be used in online adaptation of MDNN.

The performance of the proposed MDNN, its grown version into DST structure and speaker adaptation methods were evaluated on two Persian speech datasets: FARSDAT and Large FARSDAT. Experimental results have shown that training the MDNN with DST structure significantly improves its recognition performance. Moreover, the presented speaker adaptation methods further increase the recognition results achieved by the grown MDNN. The remainder of this paper is organized as follows: in Sect. 2 the proposed growing MDNN for speech recognition is introduced. Section 3 describes how the MDNN is expanded into DST structure to learn sequence information in an integrative manner. In Sect. 4, the suggested pre-training and fine-tuning procedures for these structures are described. The recommended speaker adaptation methods to improve the performance of the proposed MDNN in speaker mismatches are presented in Sect. 5. The experimental results and their analyses are given in Sect. 6. Finally, the paper is concluded in Sect. 7.

2 Modular deep neural networks (MDNN)

A modular neural network is an ANN including a series of independent neural networks (NNs) managed through some moderators. Each independent neural network, called a module, operates on a separate section of the input to perform some subtasks of the main task. The basic structure of the proposed growing MDNN for phone recognition is illustrated in Fig. 2. This network like other DNNs has L nonlinear hidden layers. As it is shown in Fig. 2, each feature detector in each layer, shown by a cuboid, contains a number of neurons to detect different features from its input. Moreover, each feature detector has a limited receptive field and shares the same weights with other feature detectors in the same layer. A feature detector with its input acts as a module in the MDNN. All the modules in the MDNN, unlike the common modular neural networks, interact with each other during training. Each time, a context window of some successive speech representations or speech frames is fed to the network. The network outputs its recognized phone for the central frame of the input.

This special structure of MDNN is designed based on the following reasons: (1) as it has been indicated in [7, 42], distinguishing sounds in phone recognition are short in duration. Thus, each feature detector only analyzes the information of a small input receptive field. (2) The network's replicated structure in the time domain enables it to be invariant to local translations of the input. Moreover, this structure reduces its trainable parameters. In this way, it is possible to scale the network to a larger and more effective one without much increment in required time and computations. (3) The first hidden layer feature detectors receive input only from one frame of the input representations. This projection from the input space to another space makes it possible for the network to analyze each representation vector separately. This transformation from each input representation to each feature detector not only decreases the required learning time of fully connected networks, but also improves the learning ability of the network. Moreover, this projection is similar to the tonotopic organization of the human peripheral auditory system [43]. Therefore, it has been used in many ANN structures for speech recognition in diverse researches [37, 43, 44].

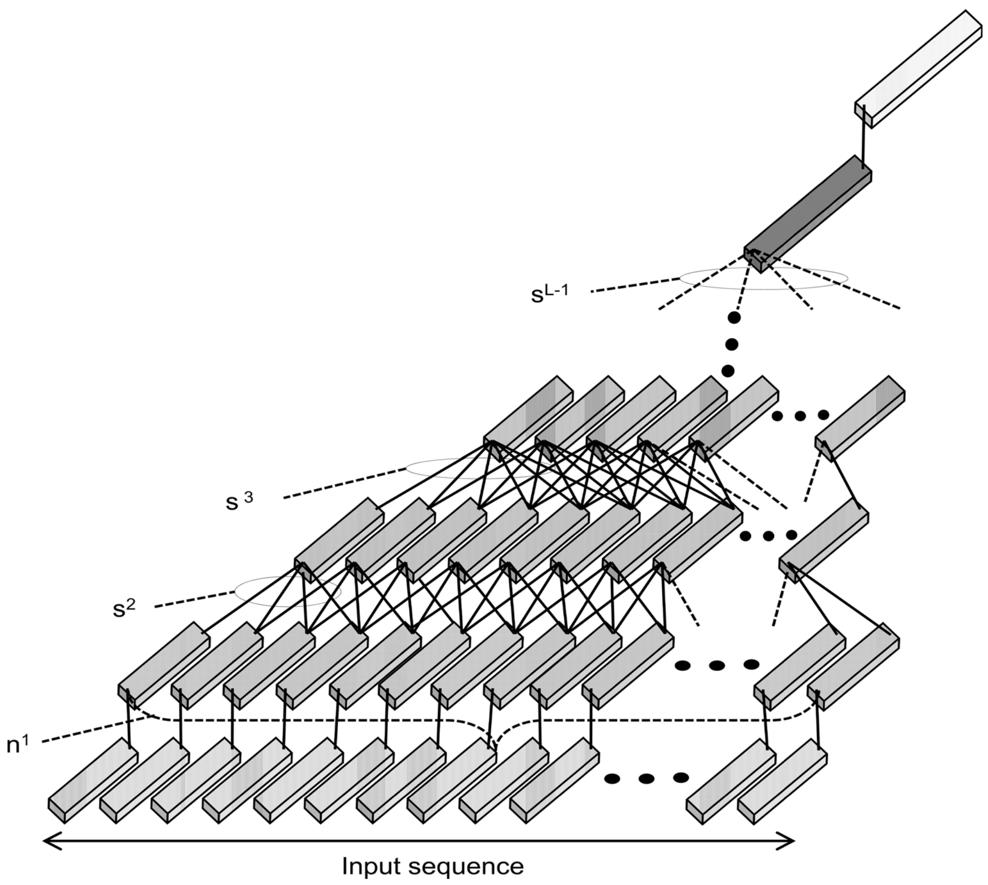


Fig. 2 Basic structure of the modular deep neural network (MDNN). In this structure, each feature detector including a sequence of neurons is displayed by a cuboid. Each line in this figure illustrates weighted connections between sequences of nodes or neurons in the lower and higher layers

(4) After projecting input representations to the first hidden layer, the higher layer detectors detect dynamic information of their inputs and feed them into higher layers. Lower layer detectors explore simple dynamic features, so their receptive field is small. This is while higher layer detectors detect more complex features from the small dynamics.

3 Stages of growing MDNN into double spatiotemporal (DST) structure

Assume that the proposed MDNN is to be used in a typical speech recognition system as an acoustic model. Thus, it must be combined with a phonetic model like HMM to model the sequential property of the speech signal. Figure 1 illustrates this hybrid architecture. In this construction, acoustic and phonetic models are trained independently. However, in this article, it is claimed that by expanding the MDNN to DST structure as shown in Fig. 3, and training it, the sequential information of phonetic labels can be learned by the network. Therefore, in this model, acoustic and phonetic information are not separated and can interact with each other to recognize an accurate phone for the input sequence.

Since the introduced MDNN is expandable, it can be simply grown in its number of nodes in the input and all the other layers to consider phonetic sequences of frames in its output layer. If the enlarged structure can be trained efficiently, the phone recognition network that is achieved by shortening this structure to the primary MDNN structure has learned not only the acoustic information of each frame, but also the temporal dependencies between phonetic sequences of frames. Therefore, in each time, for each sequence of frames fed as input to this network, it yields the recognized phone based on the information it extracts from the input representations as well as the information that its structure has obtained from the phonetic sequences subspace.

The growing procedure of the MDNN is as follows: as it is illustrated in Fig. 4, in order to generate phone sequence of I frames, the basic MDNN slides on the speech signal for I frames to yield its recognized phone of its central frame at each time. If it is required that the network recognizes the sequence of frames at the same time, it must be grown to consider all I central frames in its input. Thus, it is possible to apply the phonetic information of these I successive frames to train the expanded network. To this end, the speech representations of these frames with their right and

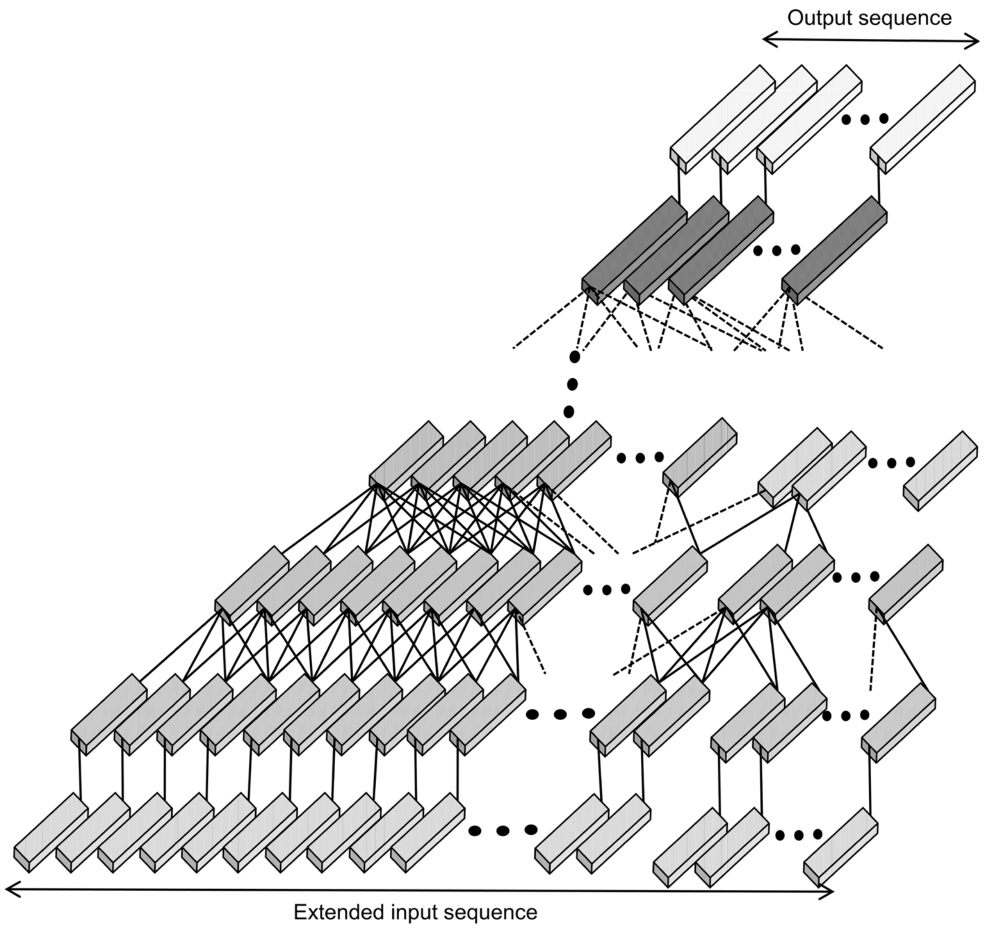


Fig. 3 MDNN with double spatiotemporal (DST) structure, obtained by growing MDNN in Fig. 2

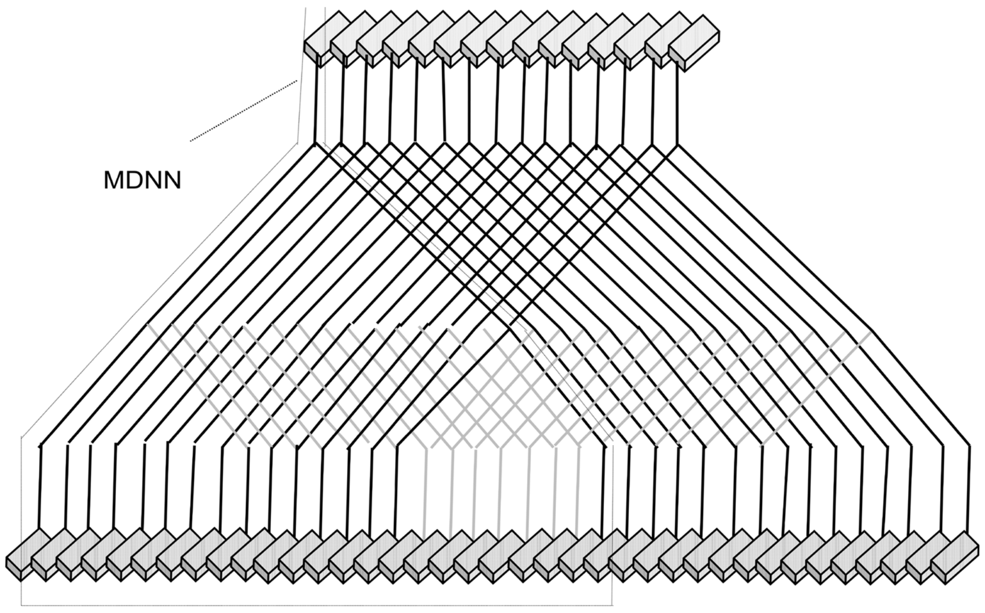


Fig. 4 Growing of the MDNN to consider sequence of labels in its output layer

left context must be fed to the network input. If *I* is set to a large value, the phonetic sequence information corresponding to these *I* successive frames is complex. In such a condition, in addition to the units of each layer, the hidden layers of the network must be increased based on the complexity of the target organization.

4 Training MDNNs

4.1 Pre-training stage

Training of DNNs is not often converged. This is related to the local minima problem which is more notable when the number of layers is increased. Therefore, pre-training methods have been proposed to initialize the DNN weights. In these methods, each layer's weights are trained separately to get rid of the local minima problem [21, 22]. One of the pre-training methods that have been presented for initializing deep auto-associative neural networks is the layer-by-layer method [45]. In this method, each layer's weights are calculated through a single-hidden-layer auto-associative neural network. The input vector of each auto-associative neural network is, in fact, the previous layer output. Based on this information, in this section, a layer-wise pre-training method is introduced for initializing the proposed MDNN weights. This pre-training procedure is designed such that it extracts efficiently the information existing in the training data in each layer to retain the maximum information required for recognition. To this end, each layer's weights are trained as if the recognition results are based on that layer's outputs. However, since input representations to the network are extracted from the speech signal through some pre-processing steps, it is required for the network to initially analyze their information. Then, the information extracted in this stage is used to exploit the discriminative information required for recognition in the following stages. According to the architecture of the MDNN with shared weights between the modules of one layer, in the presented pre-training method, only the weights of one module are trained and they are replicated for the other modules of the same layer.

The proposed pre-training procedure is as follows: MDNN is decomposed into some single-hidden-layer neural networks. In order to train the weights of the first hidden layer that analyzes input information, an auto-associative NN is considered. This NN nonlinearly decomposes the input data into components and then reconstructs the input at the output layer. Thus, it is trained so that the information loss does not occur in data analysis and reconstruction phases. Consequently, the nonlinear projection of inputs to the first hidden layer is calculated. The projected inputs to the first hidden layer are implemented as the inputs for the other single-hidden-layer NN. Since the remaining hidden layers perform multistage transforms to achieve recognition results, their weights must be trained to extract components with higher importance in recognition. Therefore, some single-hidden-layer hetero-associative neural networks, one for each layer, are trained. Each hetero-associative network receives its input from the calculated output of the previous hidden layer, and its output is the recognition layer. Figure 5 displays the proposed pre-training procedure to train the MDNN hidden layer weights.

As it is illustrated in Fig. 5, each NN consists of the weights that belong to one module of the lth layer of the MDNN ($\mathbf{W}^l$) and auxiliary weights ($\mathbf{V}^l$) used for finding out each module weights. To train single-hidden-layer NNs, a cost function must be defined. In this paper, mean

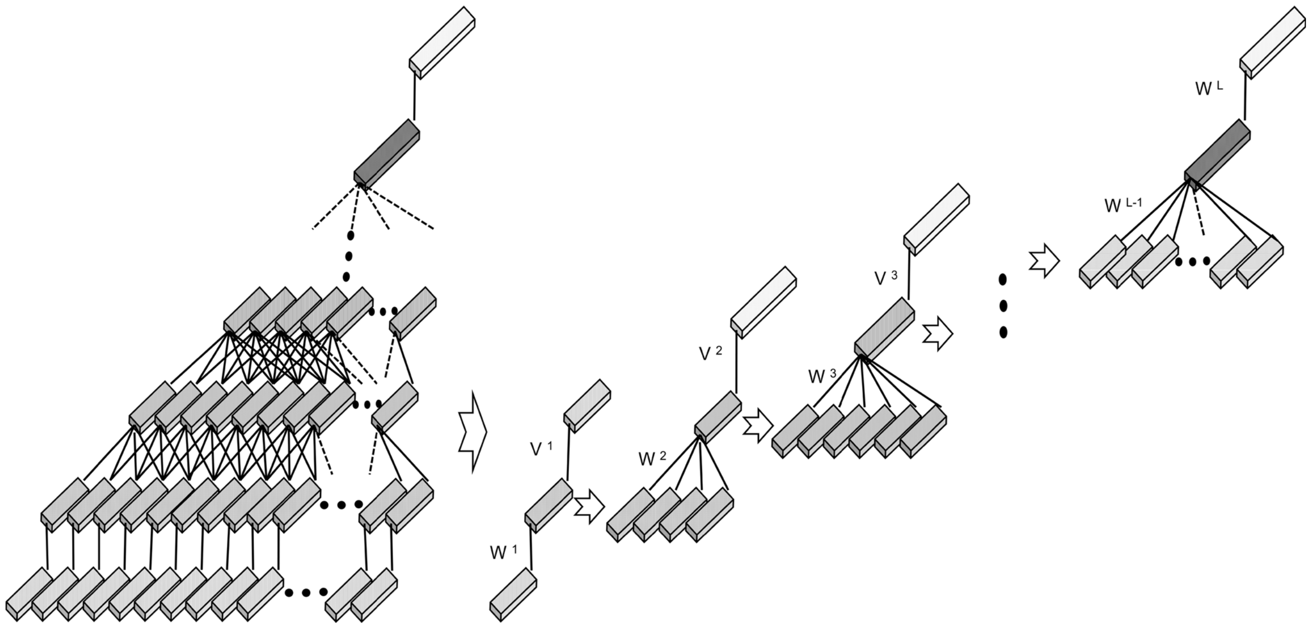


Fig. 5 Proposed pre-training procedure for training the hidden layer weights of MDNN. In this method, from each layer one module is trained, then the trained weight is copied for the same modules of that layer

square error (MSE) represented by Eq. (1) is selected as the cost function. Also, the delta rule is used to optimize this function.

$$E(\mathbf{W}^l) = \frac{1}{2}\sum_{j=1}^{J^l}\left(d_j^l - z_j^l\right)^2 \tag{1}$$

To calculate $\mathbf{W}^1$, the auto-associative NN weight, J^1 is the number of output layer units that equals the input components. In addition, $\mathbf{d}^1$ is the desired output vector, which is equal to the input representation vector and $\mathbf{z}^1$ is the output vector that auto-associative NN reconstructs. Toward training $\mathbf{W}^l$ $(l > 1)$, the input weight for each hetero-associative NN and the weight of the lth layer modules, J^l in Eq. (1) is the number of speech phones, $\mathbf{d}^l$ indicates the target phone of the central frame of each module, and $\mathbf{z}^l$ is the actual NN output of its confidence level to each phone. $\mathbf{z}^l$ in hetero-associative networks is calculated as follows:

$$\mathbf{z}^l = f\left(f\left(\left(\sum_{m=1}^{s^l}\mathbf{H}_m^{l-1}\mathbf{W}_m^l\right) - \mathbf{b}^l\right)\mathbf{V}^l - \mathbf{c}^l\right) \tag{2}$$

where, $\mathbf{b}^l$ and $\mathbf{c}^l$ are bias vectors for each feature detector in the lth layer and the recognition layer, respectively. For notational simplicity, bias vectors are added to the end of the related weight matrices. Each hidden unit typically uses the sigmoid activation function defined as:

$$f(\theta) = \frac{1}{1 + e^{-\theta}}. \tag{3}$$

As it is illustrated in Fig. 2, the receptive field of each feature detector in the lth layer, shown by s^l, is selected as an odd number in this paper. Given that n^{l-1} is the number of feature detectors in layer $l-1$, $\mathbf{H}^{l-1}$ is a matrix with n^{l-1} rows that are the feature detector outputs of that layer and $\mathbf{H}_m^{l-1}$ is the mth feature detector output. When $\mathbf{W}^l$ is calculated for each layer, it is replicated for the modules in the same layer.

4.2 Fine-tuning stage

After the pre-training stage, the initialized weights are inserted into the MDNN. Then, for fine-tuning the weights, global training of MDNN is performed. The stochastic gradient descent algorithm is used to optimize the global MSE cost function defined in Eq. (4).

$$E(\mathcal{W}) = \frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(d_{ij} - h_{ij}^L\right)^2 \tag{4}$$

where $\mathcal{W}$ denotes all connection weights in the MDNN. If the basic MDNN is fine-tuned, I is 1. Therefore, frame-level fine-tuning is conducted. Sequence level fine-tuning is performed by using the DST architecture when I is the number of sequences that the DST structure considers in its output layer at the same time. In this case, the output layer has I groups of units, each recognizes its central frame. h_{ij}^L is the network confidence level to the jth phone for the ith frame.

In both the delta and stochastic gradient descent algorithm, weight correction is done iteratively in the reverse direction of the gradient of the MSE cost function with respect to the weights.

$$\Delta\mathbf{W}^l = -\gamma\frac{\partial E}{\partial\mathbf{W}^l} \tag{5}$$

where γ is the learning rate. In the stochastic gradient descent algorithm, the gradient of the MSE cost function with respect to each layer's weights is calculated as follows:

$$\frac{\partial E}{\partial\mathbf{W}^l} = \frac{\sum_{m=1}^{n^l}\left[\mathbf{H}_m^{l-1}, \mathbf{H}_{m+1}^{l-1}, \ldots, \mathbf{H}_{m+s^l}^{l-1}, 1\right]'\mathbf{Er}_m^l}{n^l} \tag{6}$$

where $\mathbf{H}^0$ is the input frame sequence ($\mathbf{X}$) and $\mathbf{H}^L$ contains the output values of I groups of neurons in the output layer. $\left[\mathbf{H}_m^{l-1}, \mathbf{H}_{m+1}^{l-1}, \ldots, \mathbf{H}_{m+s^l}^{l-1}, 1\right]$ is constructed by concatenating the outputs of s^l feature detectors in layer $l-1$ besides the bias neuron in a vector. This vector is projected to the lth layer feature detectors by $\mathbf{W}^l$. Since $\mathbf{W}^l$ is shared for n^l modules in layer l, the gradient of the MSE cost function with respect to that weight is averaged. $\mathbf{Er}_m^L$, the error vector of the mth group of neurons in the output layer is computed as follows:

$$\mathbf{Er}_m^L = \mathbf{H}_m^L\cdot\left(1 - \mathbf{H}_m^L\right)\cdot\left(\mathbf{D}_m - \mathbf{H}_m^L\right), \quad m = 1, \ldots, n^L \tag{7}$$

In the above equation, $\cdot$ stands for elements-wise multiplication between two matrices with equal dimensions. In this equation, 1 is a vector that all of its components are 1. Error vectors of each feature detector in the lower layers are calculated by accumulating back-propagated errors of different feature detectors of higher layers getting input from that feature detector. Therefore, the error vectors of the lower layer are calculated as follows:

$$\mathbf{Er}_m^l = \mathbf{H}_m^l\cdot\left(1 - \mathbf{H}_m^l\right)\cdot\mathbf{G}_m^l \quad \text{for } l < L, \quad m = 1, \ldots, n^l \tag{8}$$

where, $\mathbf{G}_m^l$ is calculated as follows:

$$\mathbf{G}_m^l = \sum_{p=1}^{s^{l+1}}\sum_{q=1}^{n^{l+1}}\left(\mathbf{Er}_q^{l+1}\left(\mathbf{W}_p^{l+1}\right)'\right)\times\delta((m+1) - (p+q)) \tag{9}$$

where $\mathbf{W}_p^{l+1}$ is the connection weight of the pth feature detector in the lth layer which is in the receptive field of the

qth feature detector in layer $l+1$. $\delta(\cdot)$ is the Kronecker delta function. When m equals $p+q-1$, this function gives 1 and otherwise, its output is zero.

Therefore, all the error vectors of the spatiotemporal output layer that are back-propagated to the modular layers are summed in their central modules. This gives the network the potential to learn spatiotemporal information in the output layer. Figure 6 displays the error vectors back-propagating in a structure with correlated inputs. In this figure, the size of the receptive field of the feature detectors in all the layers is set to 3. As it is shown in this figure, each connection weight is trained based on the error back-propagated from the frame-level phonetic information, as well as the sequence level information in the output layer.

Generally speaking, global training of the MDNN with DST architecture or in the other word, sequence level training, can be considered as back-propagating error $(\{\mathbf{D}_1(t) \ \ -\mathbf{H}_1^L(t), \ldots, \mathbf{D}_I(t+I) \ \ -\mathbf{H}_I^L(t+I)\}|\mathbf{X}(t:t+I))$, while in frame-level training of the basic MDNN $\left(\left\{\mathbf{d}(t)-\mathbf{h}^L(t)\right\}|\mathbf{X}(t)\right)$ is back-propagated.

4.3 Semi-supervised training of MDNNs

As it was discussed in Sect. 1, in order to implement the trained MDNN for LVCSR tasks, it must be tuned on large vocabulary datasets at first. Almost any of the large vocabulary datasets has a lexicon that includes the phonemic transcription of its words. In this paper, a semi-supervised learning algorithm is proposed to fine-tune the MDNN on datasets without any frame-level phonetic transcription. This algorithm does not need any GMM-HMM models trained on those datasets to deliver frame-level labeling using forced alignment. The proposed approach makes one assumption that is the existence of a smaller dataset including frame-level labeling with similar phonetic characteristics with the large dataset. The smaller dataset can be a part of the large speech datasets.

The main steps that the proposed semi-supervised learning algorithm involves are demonstrated in Fig. 7. Primarily, MDNN is trained on the smaller dataset at first. Subsequently, the trained network is used to generate a frame-level phonetic transcription for each word of the large dataset. Extracted frame-level labels of each word are accumulated in $\boldsymbol{\beta}_{u_i}$ array. However, since this network has not been trained on larger dataset, it has some failures. Hence, the knowledge of the trained network on the smaller dataset is not enough on its own and some other knowledge must be exploited. Another canonical transcription for each frame can be generated from the phonemic transcription of the large dataset lexicon. To do this, the average length of each phone ($|ph_m|$) of the smaller dataset is computed through dividing the number of frames dedicated to this phone by the total number of frames. Then, by using this information besides the phonemic transcription of each word, frame-level transcription of that word is achieved. Assume the phonemic transcription of the ith word (u_i) with the time length of $|u_i|$ is $\{ph_1, ph_2, \ldots, ph_{l_{ui}}\}$, where l_{u_i} is the number of phonemes in this word. Thus, the frame-level transcription of this

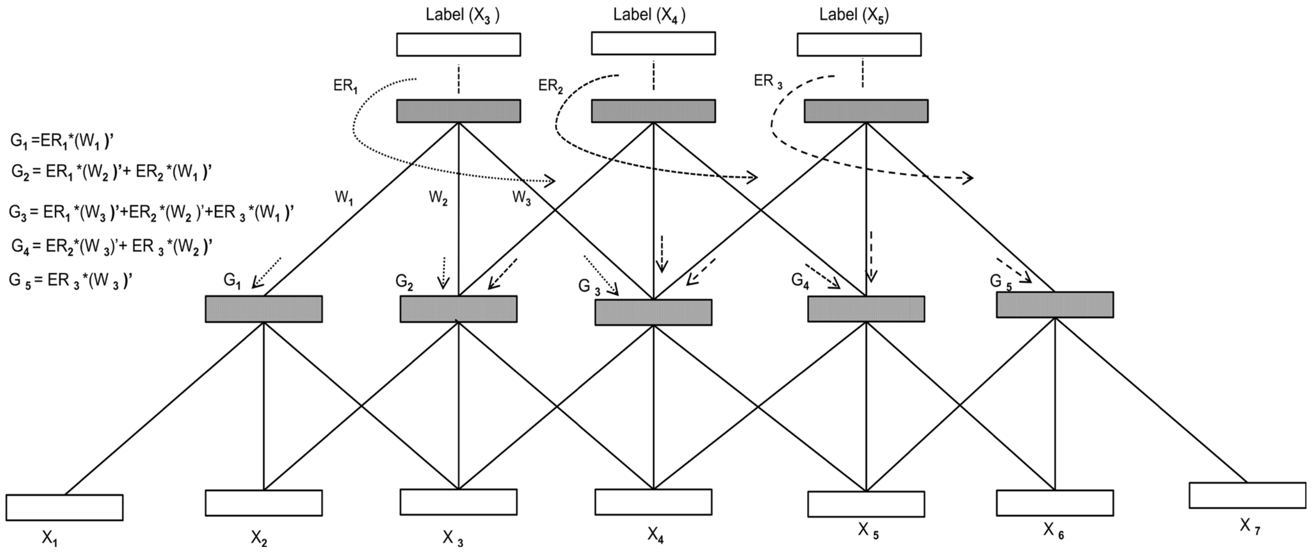
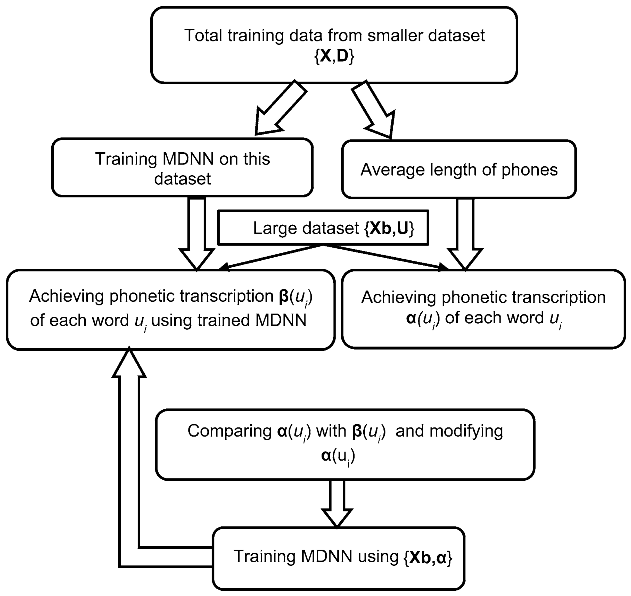


Fig. 6 Error back-propagation in a sample modular NN with replicated structure in time and limited correlated receptive fields for feature detectors as in MDNN structure. As it is illustrated, e.g., in training weight W^1, the error back-propagated from ER^1 (frame-level information) as well as $ER^1 + ER^2 + ER^3$ (sequence-level information) are involved



Total training data from smaller dataset {X,D}
Training MDNN on this dataset
Average length of phones
Large dataset {Xb,U}
Achieving phonetic transcription β(u_i) of each word u_i using trained MDNN
Achieving phonetic transcription α(u_i) of each word u_i
Comparing α(u_i) with β(u_i) and modifying α(u_i)
Training MDNN using {Xb,α}

Fig. 7 Proposed semi-supervised learning algorithm. In this algorithm, two phonetic transcriptions for each word of the large dataset (**Xb**, **U**) are achieved through different ways. Since these two transcriptions include complementary information, one of them is modified implementing the other one. The modified phonetic transcription is used for training MDNN on large dataset

word is accumulated in $\boldsymbol{\alpha}_{u_i}$ array as it is shown in Eq. (10). In this array, each phone is repeated for n_{ph_m} frames.

$$\boldsymbol{\alpha}_{u_i} = \left\{ph_1, \ldots, ph_1, ph_2, \ldots, ph_2, \ldots, ph_{|\bar{l}_{ui}|}, \ldots, ph_{|\bar{l}_{ui}|}\right\}$$
$$n_{ph_m} = \frac{|ph_m|}{\sum_{j=1}^{l_{ui}} |ph_j|} \times |u_i| \quad (10)$$

It is obvious that in this transcription, the estimated times in which transition of phones occur are not accurate. Therefore, it is required to take advantage of the information obtained by both frame-level transcriptions to accomplish the optimal one. The achieved transcription based on the average length of each phone includes partly accurate phonetic information of each word. On the other hand, the transcription obtained using the MDNN recognition results contains more reliable information about phonetic transition times in each word. In order to share the information, the transcription in array $\boldsymbol{\alpha}_{u_i}$ is compared with the transcript in array $\boldsymbol{\beta}_{u_i}$ by using dynamic programming (DP) to achieve the best compliance between these arrays. Based on the results, the time of phonetic transitions in array $\boldsymbol{\alpha}_{u_i}$ can be modified. Consequently, the modified transcription $\boldsymbol{\alpha}$ for all the training words is used to fine-tune MDNN. After some training epochs, the trained network in this stage is used to generate a new frame-level transcription. This procedure is continued until the training convergence.

5 Modifying MDNNs for nonlinear filtering of speaker variability

The proposed MDNN is trained on large amounts of data from different speakers. However, mismatches between training and test conditions such as speaker and environment differences degrade its recognition performance in test situation. Thus, in order to improve this network efficiency against the mismatched conditions, either the model or the input features must be adapted. In this section, two speaker adaptation methods for MDNN are introduced. These methods exploit the existent neural network structure efficiently and implement proportional methods for training the network with additional information. Therefore, they become capable of filtering out speaker changes nonlinearly and improving the network performance.

5.1 Nonlinear normalization of input patterns

In this speaker adaptation method, a kind of normalization is applied to the input representations against speaker changes. To this end, input representations for each phone are normalized nonlinearly based on the recognition results of the network. Therefore, the network must be trained on how to compensate the speaker information influencing the speech recognition results.

At first, the speaker-independent MDNN (SI-MDNN) is trained using speech signals from different speakers and linearly normalized input representations as follows:

$$\widehat{\mathbf{X}}_t = \frac{\mathbf{X}_t - \boldsymbol{\mu}_{\text{Total}}}{\boldsymbol{\sigma}_{\text{Total}}} \quad (11)$$

$$\boldsymbol{\mu}_{\text{Total}} = \frac{\sum_{t=1}^{T} \mathbf{X}_t}{T} \quad (12)$$

$$\boldsymbol{\sigma}_{\text{Total}} = \frac{\sum_{t=1}^{T} |\mathbf{X}_t - \boldsymbol{\mu}_{\text{Total}}|}{T} \quad (13)$$

where $\mathbf{X}_t$ is the central frame in time t and $\widehat{\mathbf{X}}_t$ is its normalized version. $\boldsymbol{\mu}_{\text{Total}}$ and $\boldsymbol{\sigma}_{\text{Total}}$ are the mean and norm 1 vectors extracted from the training data, respectively. T is the total number of training frames. The trained model forms decision regions corresponding to each phone in input space. By using these regions, the MDNN approximates the phonetic label of each test representation. Since diverse speakers articulate each phone differently, their representations must be normalized with respect to each speaker's articulatory characteristics to achieve good phone recognition results. Accordingly, the normalization must be performed separately for each phone and in a nonlinear manner. To have control on phonetic and speaker information of the input representations, input normalization is performed in a step-by-step manner.

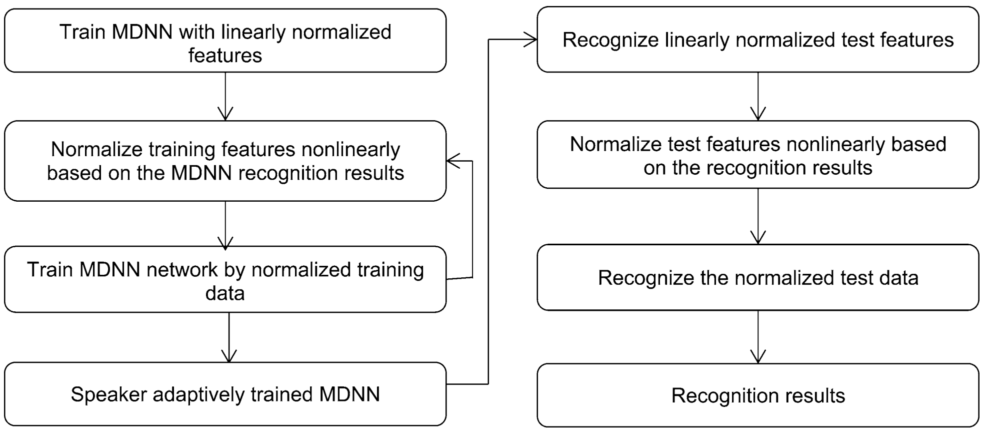

Train MDNN with linearly normalized features
Normalize training features nonlinearly based on the MDNN recognition results
Train MDNN network by normalized training data
Speaker adaptively trained MDNN
Recognize linearly normalized test features
Normalize test features nonlinearly based on the recognition results
Recognize the normalized test data
Recognition results

Fig. 8 Nonlinear normalization of the input representations in training and test conditions in the normalizing-based speaker adaptation method

In order for the MDNN to learn how to normalize each feature vector nonlinearly, it must be trained speaker adaptively at first. During speaker adaptive training, the network learns how to move the input representations in a direction in which besides maintaining their phonetic knowledge, their speaker information is filtered out. This can be achieved by normalizing the representations based on the recognition results of the trained MDNN at each iteration and in the next iteration, the MDNN is trained based on these normalized representations. This training procedure continues until the convergence is achieved. Figure 8 shows how this speaker adaptation method is conducted.

To clarify this, assume that each time a sequence of representations from the rth training utterance with the central frame $\mathbf{X}_{rt}$ and length T_r is fed to the MDNN input. The MDNN outputs its estimated phone for the central frame of this sequence. In fact, if the kth neuron has maximum value among all other output neurons, then the estimated output will be the kth phone. Following this procedure, the normalization parameters of each phone in this utterance can be achieved as shown in Eqs. (15–17). Afterward, the input representations are normalized such that their position in acoustic space moves to the recognized phone position computed from whole training data. The following equation shows how this normalization is accomplished.

$$\widehat{\mathbf{X}}_{rt} = \frac{\mathbf{X}_{rt} - \boldsymbol{\mu}^{\mathrm{Total}} - (\boldsymbol{\mu}\boldsymbol{r}_k - \boldsymbol{\mu}\boldsymbol{l}_k)}{\boldsymbol{\sigma}} \quad s.t. \quad \arg\max\left(\mathbf{H}^L(\mathbf{X}_{rt})\right) = k \tag{14}$$

where $\boldsymbol{\mu}\boldsymbol{r}_k$ is the mean vector of all the representations in the rth utterance that the network recognizes as the kth phone. It is computed as follows:

$$\boldsymbol{\mu}\boldsymbol{r}_k = \frac{\sum_{t=1|_{\arg\max(\mathbf{H}^{\mathrm{L}}(\mathbf{X}_{\mathrm{rt}}))=\mathrm{k}}}^{T_r} \mathbf{X}_{rt}}{Nr_k}$$
$$Nr_k = \mathrm{num}\left(\mathbf{X}_{rt} | \arg\max(\mathbf{H}^L(\mathbf{X}_{rt})) = k\right) \tag{15}$$

And $\boldsymbol{\mu}\boldsymbol{l}_k$ as it is computed in Eq. (16), is the mean vector of those representations of training data for which the label is phone k.

$$\boldsymbol{\mu}\boldsymbol{l}_k = \frac{\sum_{t=1|_{D(\mathbf{X}_t)=k}}^{T} \mathbf{X}_t}{Nl_k}$$
$$Nl_k = \mathrm{num}(\mathbf{X}_t | \arg\max(D(\mathbf{X}_t)) = k) \tag{16}$$

In addition, $\boldsymbol{\sigma}$ is computed as follows:

$$\boldsymbol{\sigma} = \frac{\sum_{t=1|_{\arg\max(\mathbf{H}^{\mathrm{L}}(\mathbf{X}_{\mathrm{rt}}))=\mathrm{k}}}^{T_r} |\mathbf{X}_{rt} - \boldsymbol{\mu}_{\mathrm{Total}} - (\boldsymbol{\mu}\mathbf{r}_k - \boldsymbol{\mu}\mathbf{l}_k)|}{Nr_k} \tag{17}$$

During the test phase, input representations of each test utterance are normalized iteratively via modified MDNN recognition results. The modified MDNN is the speaker adaptively trained MDNN which has learned the ways to move each distorted input to the optimum location. The procedure of normalizing the input representations and recognizing the normalized representation in the test phase is continued until the recognition results converge.

5.2 Model adaptation

In this adaptation strategy, the speaker information extracted by a speaker recognition network is fed to all hidden and output layers of the original speech recognition network through a set of connection weights. The trained speech recognition network has created decision regions for each phone using all of the training data. The connection weights between speaker and speech recognition networks are trained on whole training data to learn how to change each phone decision region based on the speaker information. An advantage of this adaptation method is that computational complexities in the training phase are reduced as only the new connection weights are trained and there is no need to train speech or speaker recognition networks adaptively. Moreover, this method does not require any adaptation data.

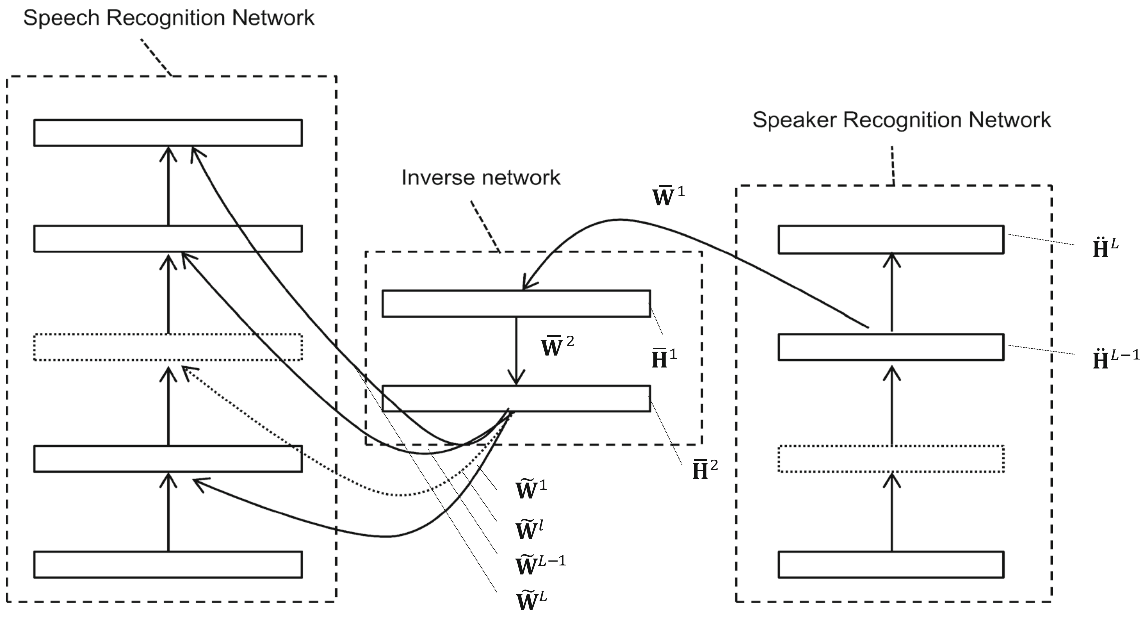

Speech Recognition Network
Inverse network
Speaker Recognition Network
$\bar{\mathbf{W}}^1$
$\bar{\mathbf{W}}^2$
$\bar{\mathbf{H}}^1$
$\bar{\mathbf{H}}^2$
$\ddot{\mathbf{H}}^L$
$\ddot{\mathbf{H}}^{L-1}$
$\widetilde{\mathbf{W}}^1$
$\widetilde{\mathbf{W}}^l$
$\widetilde{\mathbf{W}}^{L-1}$
$\widetilde{\mathbf{W}}^L$

Fig. 9 Block diagram of the proposed model-based adaptation method

In the proposed adaptation method, as it is shown in Fig. 9, speaker and speech recognition networks are implemented separately on the same sequence of input representations. The speaker recognition network extracts speaker information of its input central frame from the context. Then, an inverse network processes the extracted information and feeds them to the speech recognition network. In this way, speaker information is used to adapt the speech recognition network toward any new speaker. The scheme of connecting the inverse network output to all of the hidden and output layers is similar to the results achieved in [33, 46]. In [33], it has been shown that connecting the speaker code to one of the hidden layers is not as efficient as connecting it to all the layers.

Due to the modular structure of the proposed speech recognition network, the procedure of applying speaker information to different modules is important. The simplest way is to compute speaker information of the central input frame of the speech recognition network and feed it to all the modules in different layers. However, each module processes the speech information of different local receptive fields in the time axis. Moreover, speaker information is time variant. Therefore, the speaker information imported to each module must be extracted from the central input frame of that module. Figure 10 illustrates how this adaptation scheme is implemented to the proposed MDNN.

As it was indicated, in this adaptation method, only the connection weights and the weights of the inverse network are trained from the training data to adapt SI-MDNN. Therefore, both speech recognition and speaker recognition networks are trained independently before adaptation. Then, the connection between two networks through the inverse network connection weights which are initialized randomly is established. In the training phase, several epochs of the stochastic gradient decent using the training data are run to train the inverse network weights. Assume that the connection weights from the inverse network output to each layer of the basic MDNN be denoted by $\widetilde{\mathbf{W}}^l$. Therefore, each feature detector in each layer gets input from both the lower layer nodes in its receptive field and the inverse network. In this case, the output values of each module m in each layer l are computed as follows:

$$\mathbf{H}_m^l = f\left(\left(\sum_{j=m}^{m+s^l-1}\left[\mathbf{H}_j^{l-1},1\right]\mathbf{W}_{j-m+1}^l\right) + \left(\bar{\mathbf{H}}^L_{m+\sum_{r=1}^{l}\frac{s^r-1}{2}}\right)\widetilde{\mathbf{W}}^l\right)$$
$$\text{for}\ \ m = \left\{1,2,\ldots,n^l\right\}. \tag{18}$$

where $\bar{\mathbf{H}}^L_{m+\sum_{r=1}^{l}\frac{s^r-1}{2}}$ is the output vector of the inverse network. This vector is extracted from the speaker recognition network when it is located on the input sequence, the central frame of which is the same as that of the mth module. The central frame of the mth module of the MDNN is the $(m + \sum_{r=1}^{l}\frac{s^r-1}{2}$th frame of the network input.

Using the back-propagation algorithm, the error gradient with respect to $\widetilde{\mathbf{W}}^l$ is calculated as follows:

$$\frac{\partial E}{\partial \widetilde{\mathbf{W}}^l} = \frac{1}{n^l}\sum_{m=1}^{n^l}\left(\bar{\mathbf{H}}^L_{m+\sum_{r=1}^{l}\frac{s^r-1}{2}}\right)'\mathbf{ER}_m^l \tag{19}$$

where $\mathbf{ER}_m^l$ stands for the error vector back-propagated to the mth feature detector in the lth layer of the MDNN

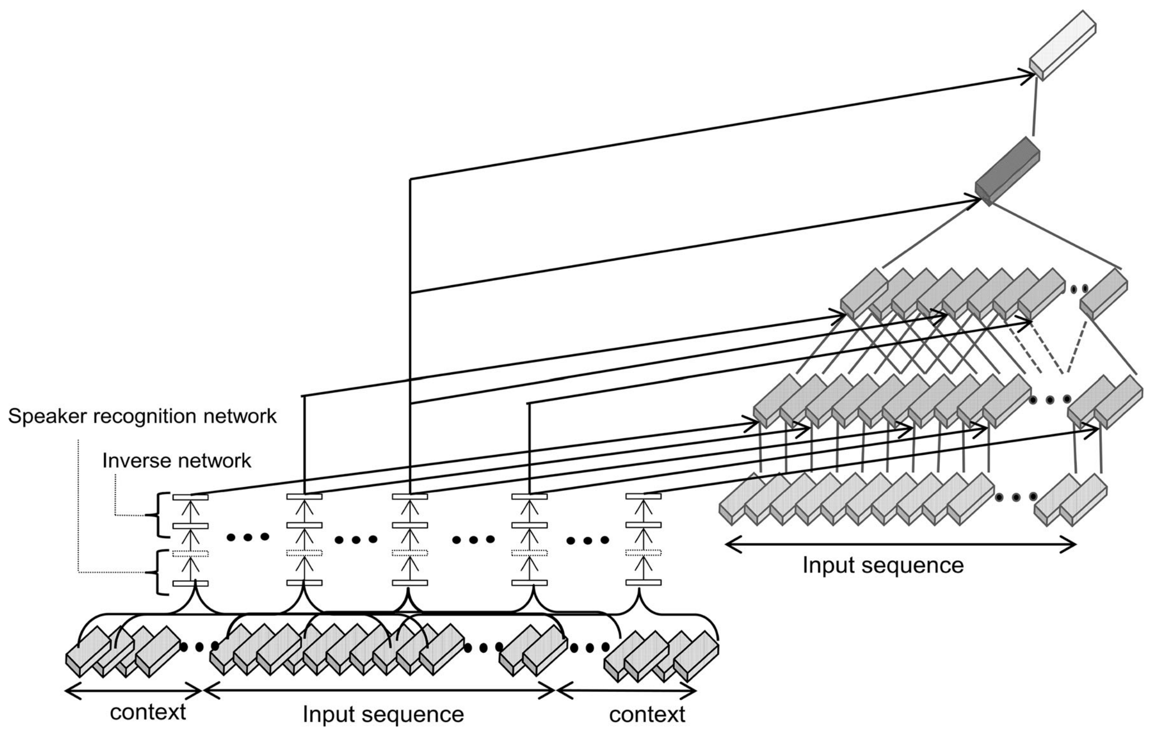


Fig. 10 Assigning the proposed model adaptation method for the introduced MDNN. The speaker recognition network extracts speaker information of each frame as a central frame of a context. The inverse network processes the extracted speaker information and applies the relevant information to each module

calculated as it is described in Eq. (7). In the same way, the gradient with respect to the inverse network weights denoted by $\widetilde{\mathbf{W}}^l$ is computed as follows:

$$\frac{\partial E}{\partial \overline{\mathbf{W}}^l} = \frac{\sum_{i=1}^{n^1} \left(\overline{\mathbf{H}}_i^{l-1}\right)' \overline{\mathbf{ER}}_i^l}{n^1} \quad \text{for} \quad 1 < l \leq L \tag{20}$$

$$\frac{\partial E}{\partial \overline{\mathbf{W}}^1} = \frac{\sum_{i=1}^{n^1} \left(\ddot{\mathbf{H}}_i^{L-1}\right)' \overline{\mathbf{ER}}_i^1}{n^1} \tag{21}$$

where $\ddot{\mathbf{H}}_i^{L-1}$ is the output vector of the top most hidden layer of the speaker recognition network for the input sequence for which the central frame is the *i*th frame of the MDNN input. $\overline{\mathbf{ER}}_i^l$ is calculated as follows:

$$\overline{\mathbf{ER}}_i^L = \overline{\mathbf{H}}_i^L \times \left(1 - \overline{\mathbf{H}}_i^L\right) \quad \times \left(\sum_{l=1}^{L} \sum_{j=1}^{n^l} \left(\mathbf{ER}_j^l \left(\widetilde{\mathbf{W}}^l\right)' \right) \times \delta\left(i - \left(j + \sum_{r=1}^{l} \frac{s^r - 1}{2} \right) \right) \right) \tag{22}$$

$$\overline{\mathbf{ER}}_i^l = \overline{\mathbf{H}}_i^l \times \left(1 - \overline{\mathbf{H}}_i^l\right) \cdot \left(\overline{\mathbf{ER}}_i^{l+1} \left(\overline{\mathbf{W}}^{l+1}\right)' \right) \quad \text{for } 1 \leq l < L \tag{23}$$

where $\delta(.)$ equals 1 when the central frame of the speaker recognition network is identical with the central input frame of the *j*th module in the *l*th layer of the MDNN. The gradients computed in Eqs. (20) and (21) are used to update the weights associated with the inverse network.

In the test phase, the speaker recognition network extracts speaker information of each input sequence of frames. The extracted information shows the similarities between the acoustic characteristics of the central frame of this sequence with those of the training speakers. This information is processed by the inverse network and fed to the relevant modules of the MDNN. Consequently, MDNN displaces its decision regions for each phone based on the applied information.

6 Experimental results

To evaluate the proposed methods for developing DNN-based speech recognition systems, the experiments were conducted on two Persian speech datasets: FARSDAT [47] and Large FARSDAT [48].

The speech data were analyzed using a 23-ms window with a fixed frame rate of 12.5 ms. In the following experiments, the speech signal was represented using Logarithm of Hanning Critical Band filter Bank (LHCB) parameters with 18 coefficients distributed on a bark-scale. Different researches have shown that these representations are suitable for ANNs used in speech recognition systems [37, 46, 49]. However, typical speech recognition systems use Mel-Frequency Cepstral Coefficient (MFCC) representations. One of the main assumptions in the process of forming MFCCs is using discrete cosine transform (DCT) to decorrelate Mel-spectral vectors. This property of MFCCs is suitable for HMMs. Since, in the independence assumption made by HMM models, the output observation in time t depends only on the current state and is independent of the previous observations and states. However, as ANNs are capable of making use of less pre-processed representations [15], the DCT transformation eliminates useful information for ANNs. So, LHCB parameters were preferred in the following experiments. The input representations were normalized using longitudinal normalization of norm 1 as it is depicted in Eq. (11).

In all of the experiments, the Viterbi algorithm was used to decode the sequence of phones as it had been implemented in [50], where each phone was assumed to be represented by a multistate HMM. The emission likelihoods of the HMM states were assumed equal and were derived from the associated output of the MDNN. In this algorithm, bi-phone language model (LM) scores were used as transition parameters between phones. In the following experiments, these scores were estimated from the training set. In the first proposed development strategy for the basic MDNN, output sequence information was implemented to fine-tune the MDNN enlarged to DST structure. Consequently, the network was expected to contain information about phonetic sequences. Hence, to evaluate how much phonetic sequence information this network had learned, at first the recognition results were calculated by applying the Viterbi decoder without using LM scores. Then, in order to compare the best recognition results of the MDNN network trained with DST structure with basic MDNN and GMM-HMM, bi-phone LM was used in Viterbi computations.

The decoding results of the Viterbi decoder may be improved by using the insertion penalty parameter. This parameter is implemented into the decoder through modifying transition parameters as presented in Eq. (24).

$$a_{ij} = GSF \times a_{ij} + IP \tag{24}$$

where a_{ij} is the transition parameter between phone i and phone j, which is replaced with the bi-phone language model. *GSF* is the grammar scale factor and regulates the influences of bi-phone LM information on the decoder. In this equation, *IP* is the insertion penalty.

In this section, at first the proposed MDNN structure and its development approaches are evaluated on the FARSDAT dataset. Afterward, Large FARSDAT is used to evaluate the efficiency of the proposed methods on a large dataset.

6.1 Experiments on FARSDAT

FARSDAT [47] is a well-known Persian speech dataset. It includes recorded Persian speech data collected from 304 speakers with gender, age, dialect and academic degree differences. In this dataset, each speaker speaks 20 sentences in two sessions. The sentences are phonetically balanced and provide allophones in all possible phonetic contexts. This dataset has been registered in European Language Resource Association (ELRA) with the ID: ELRA-S0112. Moreover, FARSDAT is nearly equal to the TIMIT dataset for English [37]. This dataset has been implemented extensively in researches such as: [37, 51–54]. In this work, the speech data of 297 speakers were selected for the training set and the rest for the test set. In total, 5940 sentences are allocated for the training phase and 140 sentences for the test phase. This dataset includes both phonetic and word transcriptions.

6.2 Performance of MDNN on FARSDAT

The basic MDNN structure that was used in this article is shown in Table 1.

The motivation of selecting such structure is as follows. In [55], it has been shown that in order to provide a phone recognition network with some information about the acoustic content of a frame, a context window of 7–25 frames must be considered in its input. This length of context enables the phone recognition network to consider the information of both short and long phones. Mohamed has achieved similar results in [15]. In [15], it has been indicated

Table 1 Implemented structure of the basic MDNN in this article

Layer	Number of input vectors/ feature detectors in each layer	Number of nodes in each input vector/ feature detector	Each feature detector receptive field
Input	23	18	–
First hidden layer	23	64	1
Second hidden layer	15	512	9
Third hidden layer	1	62	15
Output	1	36	1

that the context window of 11–27 frames, i.e., the average size of phones or syllables, is suitable for a phone recognition network. Thus, the proposed MDNN structure considers a context window of 23 frames as its input. By comparing the phone recognition results that have been achieved in [37] with those in [46], it can be concluded that a TDNN with a semi-connected first hidden layer converges much faster than its full-connected version without losing its performance much. The connections in the semi-connected first hidden layer are such that each frame is processed independently. Since the extracted feature vectors from the speech signal have a dimension of 18 for each frame, much more neurons are considered in the first hidden layer to be able to detect multiple features from each speech frame. In [37, 44, 53], 32 neurons were considered for processing of each frame. However, since the implemented training data in this article are larger, the number of neurons can be extended to 64. Increasing the number of neurons in this layer improves the discrimination of inputs. Thus, the first hidden layer includes 23 sequences of 64 neurons or feature detectors. Moreover, as it was discussed in Sect. 2, the MDNN is designed such that lower layers explore smaller dynamic features and higher layers detect larger dynamic information. Thus, the information extracted from nine consecutive frames in the first hidden layer is processed by a series of neurons (512 neurons) in the second hidden layer. The processing of these small dynamics (with the width of 9) is performed to detect distinguishing sounds in phone recognition. Thus, 15 sequences of feature detectors in the second hidden layer are considered to extract the information from short sequences of the first hidden layer neurons. Afterward, the extracted features in this layer are combined and processed by a series of neurons (62 neuron) in the third hidden layer. In this layer, large dynamics (with the width of 15) of its input is processed. Thus, the feature detectors in this layer have a receptive field with size 15. Finally, phonetic recognition is achieved in the output layer. Thus, 36 neurons in the output layer correspond to the 36 Persian phones.

This network was pre-trained layer-wisely with the recipe indicated in Sect. 4 using the stochastic gradient descent algorithm. For the auto-associative neural network used to train first hidden layer weights, 1000 epochs of training with learning rate annealing started with 0.1 were run. While, for training second and third hidden layer weights, two single-hidden-layer hetero-associative networks were used as illustrated in Fig. 5. The second and third single-hidden-layer networks were trained with learning rate annealing started with 0.1. Figure 11 shows the MSE of the training data for the single-hidden-layer networks used to train second and third hidden layer weights for the same number of epochs. As it is noticeable in this figure, the second network yields less training error.

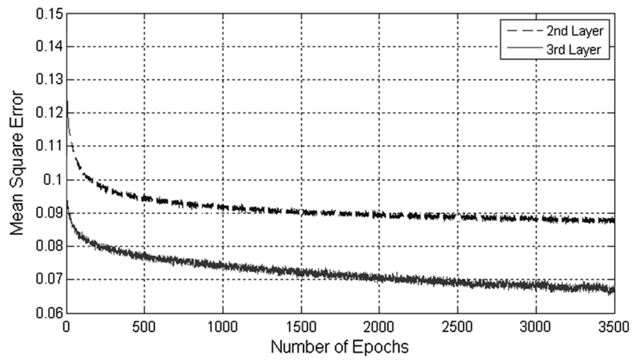


Fig. 11 Mean square error (MSE) of training two single-hidden-layer networks used to pre-train second and third hidden layer weights of MDNN on FARSDAT

Table 2 Comparing phone recognition rate of different structures of MDNN with GMM-HMM on FARSDAT (bi-phone LM is used)

Phone recognition model	Phone recognition rate (%)
MDNN with three hidden layers	76.5
MDNN with two hidden layers	69
GMM-HMM	73.17

This result indicates that the information extracted by the third hidden layer weights is more abstract and efficient for the final recognition purpose.

The trained weights were substituted into the MDNN. For fine-tuning of this network, the stochastic gradient descent algorithm with the learning rate of 0.1 was used. The momentum was always kept fixed at 0.7. In order to compare the performance of the proposed MDNN with graphical models, a monophone GMM-HMM model that includes a set of multistate GMM-HMMs for each of the Persian phones was trained using HTK toolbox version 3.4 [56]. In order to achieve the best recognition results of the monophone GMM-HMM model, eight mixture Gaussians for each state of the HMM models were used. Moreover, according to the conditional independence assumption in HMM models, MFCC features were implemented to be trained to these models. Besides, the first- and second-order time derivations of MFCCs were used to consider dynamic information in training the GMM-HMMs.

In Table 2, the best recognition performance of the basic MDNN is compared with that of the GMM-HMM model. Furthermore, in order to investigate the effect of the fully connected output layer, a two-hidden-layer neural network with a similar structure as the MDNN but without the top layer was trained separately. The results in Table 2 show that post-processing of the combined outputs of different modules achieves slightly better recognition results. Moreover, the results confirm that the basic MDNN performs 3.3 % better than the GMM-HMM.

Table 3 Structure of the expanded MDNN into DST-MDNN

Layer	Number of input vectors/ feature detectors in each layer	Number of nodes in each input vector/ feature detector	Each feature detector receptive field
Input	37	18	–
First hidden layer	37	64	1
Second hidden layer	29	512	9
Third hidden layer	15	62	15
Output	15	36	1

6.2.1 *Performance of MDNN trained with DST structure on FARSDAT*

After the MDNN was trained based on the frame-level MSE training criterion, its structure was expanded to DST architecture to be trained on phonetic sequences. Therefore, its structure characteristics were changed as it is shown in Table 3. In this experiment, the MDNN was grown to DST structure to extract phonetic sequence information of 15-frame sequences. This information was not so abstract to require a deeper neural network. Therefore, the number of MDNN hidden layers was not changed. However, if it was required to extract complex information of long phonetic sequences, the MDNN had to be grown in its hidden layers as well as each layer nodes.

Training of the MDNN with DST structure was conducted as discussed in Sect. 4. Training started with an initial learning rate of 0.1. As the MDNN was grown to DST structure to become able to be trained on phonetic sequence information, in the test phase there was no need for the added modules. Therefore, by removing added modules to the network after training it, the DST-MDNN structure shrank to the basic MDNN structure. However, the remained MDNN included phonetic sequence information. Experimental results shown in Fig. 12 demonstrate that the MDNN trained with DST structure outperforms both MDNN trained based on frame-level information by 4.5 % and GMM-HMM model by 7.9 %. Phone recognition results achieved without using bi-phone LM scores in the Viterbi decoder display that the DST structure has accomplished to model phonetic sequences well. The significant improvement achieved by learning 15 sequences of phones to the model shows that the network becomes capable of filtering invalid phonetic sequences in its own structure and only yields valid ones.

6.2.2 *Performance of the speaker adaptation methods on FARSDAT*

In this set of experiments, the proposed methods to improve the performance of the MDNN trained with DST structure under the mismatches between training and test conditions are examined.

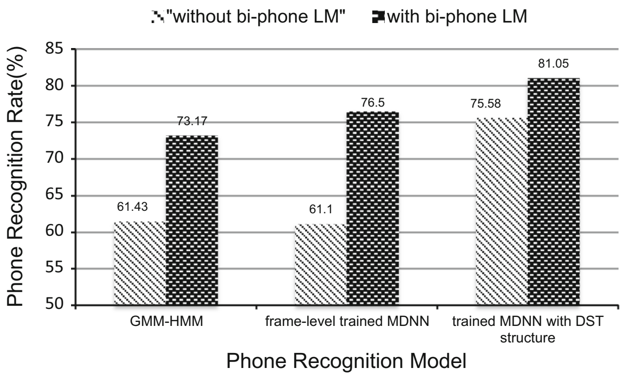


Fig. 12 Phone recognition rate of basic MDNN, MDNN trained with DST structure and GMM-HMM in two conditions of using bi-phone LM in decoding and without it on FARSDAT

a. Performance of nonlinear normalization method

In this experiment, the MDNN was first trained with DST structure. Then, as it was indicated in Sect. 5, the network was trained adaptively with the nonlinearly normalized input representations. In this training procedure, the stochastic gradient descent algorithm with a fixed learning rate of 0.1 was used. After training convergence, nonlinear normalization of test representations was carried out. Two iterations of nonlinear normalization were found sufficient to achieve the recognition convergence. Table 4 shows the nonlinear normalization performance. The results indicate that the proposed normalization method improves the phone accuracy rate of the MDNN trained speaker independently with DST structure by 2.8 % when bi-phone LM is implemented.

Different situations were also investigated in training and test phases. In one situation, the SI network was not trained speaker adaptively before being used in the test phase. However, nonlinear normalization was conducted in the test phase. In another situation, the network was trained speaker adaptively in the training phase, but only the linear normalization was applied in the test phase. The results show that the network performance improves when training and test conditions are consistent with each other. Moreover, the speaker adaptively trained network in test phase is crucial in this adaptation method. This can be explained because by speaker adaptive training, the network can learn

Table 4 Phone recognition rate of the speaker-independent MDNN trained with DST structure (SI MDNN), adapted using nonlinear normalization method in different conditions: when the network is trained speaker adaptively (SA) or not in training phase and when nonlinear normalization is applied in test phase or not (bi-phone LM is used)

Phone recognition network + test frames	Phone recognition rate (%)
SI MDNN + linearly normalized frames	81.05
SA trained MDNN + nonlinearly normalized frames	**83.83**
SI MDNN + nonlinearly normalized frames	80.53
SA trained MDNN + linearly normalized frames	80.26

Bold value indicates the nonlinear normalization adaptation method is effective when the MDNN has been trained speaker adaptively before

Table 5 Implemented structure of the speaker recognition network

Layer	Number of nodes in each layer
Input	23 × 18
First hidden layer	736
Second hidden layer	512
Output	298

nonlinear paths to move the distorted inputs to the optimum locations and use the trained paths to normalize test representations.

b. Performance of model adaptation method

In this subsection, the proposed model adaptation method for adapting speaker independent MDNN to speaker variability is examined. To do this, a speaker recognition network was used. This network structure as shown in Table 5 is a TDNN with a semi-connected first hidden layer as the MDNN. The initial structure of this neural network was used in [34]. Each time, a window of 23 consecutive frames is imported to this network. The information of each frame is processed by a series of neurons (32 neuron) in the first hidden layer, so 23 × 32 neurons are in the first hidden layer. In this layer, different acoustic representations of each speaker are extracted from his speech frames. Subsequently, the extracted information is processed through a hidden layer of 512 neurons and an output layer of 298 neurons, respectively. The output layer has 298 neurons to indicate the training speaker codes. For each of the 297 training speakers, one neuron takes the value of 1 and the others take zero. The 298th neuron is used to consider silence. Since silence frames have no speaker information, this neuron is activated when no speaker information is fed to the network input. After training the network, it was applied by the proposed model adaptation method as shown in Fig. 9. Then, the inverse network and the connection weights between MDNN and speaker recognition networks were trained.

The outcomes in Table 6 verify that the proposed model adaptation method improves the recognition performance of the MDNN trained with DST structure by 1.6 %. Moreover, Table 6 shows that the information extracted from the top most hidden layer of the speaker recognition network is more effective than what is obtained from its output layer in adapting MDNN.

Table 6 Phone recognition rate of MDNN adapted with the proposed model adaptation method on FARSDAT in two different choices of extracting the speaker information from the output layer (case 1) or the top most hidden layer (case 2) of speaker recognition network (bi-phone LM is used)

Phone Recognition network	Phone recognition rate (%)
SI MDNN	81.05
SI MDNN + case 1	82.32
SI MDNN + case 2	**82.66**

The result in bold represents that when the speaker information is extracted from the top most hidden layer of the speaker recognition network, the performance of the adapted model is improved

c. Comparing with another speaker adaptation method

In this experiment, the proposed adaptation methods are compared with a well-performed speaker-code-based adaptation method suggested in [33]. This method has shown better results in adapting DNNs with a small amount of adaptation data in comparison with many adaptation methods, such as those presented in [31, 32, 36]. In the conducted experiment in this work, cross-validation was used to implement this adaptation method and to test it for each speaker. Each speaker in FARSDAT dataset has spoken in two sessions. Therefore, in one run of the adaptation, the first session of each test speaker was used for adaptation and another session was used for test. In the other run, adaptation and test sections were alternated. The reported phone recognition result is the average of the results achieved in both runs. The results displayed in Fig. 13 demonstrate that this method improves the recognition accuracy of the speaker independent MDNN model trained with DST structure by 0.7 %. However, the proposed adaptation methods in this article, not only achieve more significant recognition improvement, but also do not require any adaptation data and adapt the MDNN online.

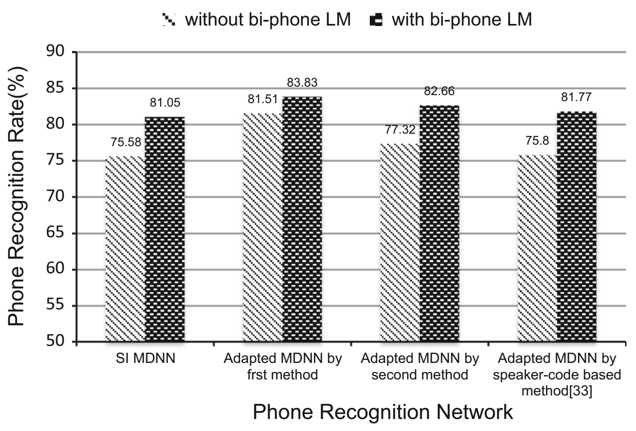


Fig. 13 Comparison of the proposed adaptation methods in this article with a well-performed adaptation method proposed in [33]

6.3 Experiments on large FARSDAT

Large FARSDAT is a large Persian speech dataset. It includes the speech data of 100 speakers with age, gender, dialect and educational level differences. Each speaker has read about 20 pages of various texts of newspapers in an office room. The texts cover a variety of fields such as politics, culture, economics and sports. This dataset contains a lexicon with phonemic transcription. In the following experiments, the speech signal of 95 speakers is dedicated for the training set and the remained data for the test.

6.3.1 Semi-supervised training of MDNN with DST structure on Large FARSDAT

In order to fine-tune the MDNN with DST structure on Large FARSDAT, the same procedure indicated in Fig. 7 was adopted. Sequentially trained MDNN on FARSDAT as shown in Fig. 12 has a highly acceptable performance. Therefore, this network was used to decode each word and obtain the phonetic frame-level transcription $\boldsymbol{\beta}_{u_i}$. Besides, the other transcription $\boldsymbol{\alpha}_{u_i}$ was achieved based on the average length of each phone on FARSDAT. Then, $\boldsymbol{\alpha}_{u_i}$ was aligned with $\boldsymbol{\beta}_{u_i}$ by using DP algorithm. Consequently, $\boldsymbol{\alpha}_{u_i}$ was modified based on the information attained from $\boldsymbol{\beta}_{u_i}$. As insertion and deletion costs obtained by applying DP were dependent on the phone transition times, they were compensated in $\boldsymbol{\alpha}_{u_i}$. However, substitution cost was mostly dependent on the differences between the phonemic transcription ($\boldsymbol{\alpha}_{u_i}$) and the phonetic one ($\boldsymbol{\beta}_{u_i}$), so it could not be made up. Consequently, the network was trained using the modified transcription $\boldsymbol{\alpha}_{u_i}$. When its training was converged, the above procedure was conducted once more based on the recognition results of the trained network till then. Table 7 displays phoneme recognition rate of the MDNN with DST structure trained on Large FARSDAT using the semi-supervised training procedure. To illustrate the effects of using two information resources to obtain the frame-level transcription, in the other experiment the transcription $\boldsymbol{\alpha}_{u_i}$ was used to train the network individually. Experimental results demonstrate that the proposed semi-supervised training strategy enables the network to be trained on Large FARSDAT.

Table 7 Comparing the MDNN performance on Large FARSDAT when it is trained only on FARSDAT dataset with when it is fine-tuned on Large FARSDAT by using two semi-supervised learning algorithms

Semi-supervised learning algorithm	Phoneme recognition rate
Not using	53
Based on phone length averages	64
Based on combining the information	69

Since Large FARSDAT has only phonemic transcription for each word, in order to evaluate the network performance, the recognition results of the network for each utterance were given to the Viterbi decoder to yield the phonetic sequence of its input. Therefore, the obtained phonetic sequence of that utterance was compared with its phonemic sequence. In this way, phoneme recognition rate of the network was calculated.

6.3.2 Performance of the speaker adaptation methods on Large FARSDAT

In this section, the proposed adaptation methods are implemented to adapt the trained MDNN in the previous subsection on Large FARSDAT. Performing the nonlinear normalization method required frame-level transcription. Thus, the optimal phonetic transcription that was achieved in the previous section was implemented. Then, nonlinear normalization was performed as it was indicated in Sect. 5. In order to perform the model-based adaptation method, since the training speakers in this dataset are fewer than those of FARSDAT, a speaker recognition network with fewer neurons was used. The output layer of this network had 96 neurons, and its top most hidden layer had 256 neurons. Table 8 shows the results obtained by performing these adaptation methods on Large FARSDAT. As it is illustrated in this table, the nonlinear normalization method can not greatly increase the phoneme recognition rate of the MDNN. This may be due to the sensibility of this method to recognition results of the speech recognition network. Therefore, the method becomes inappropriate for not accurately phonetic transcribed datasets. However, the model adaptation method improves MDNN performance by 1 %.

Table 8 Performance of the proposed speaker adaptation methods implemented on trained MDNN with DST structure on Large FARSDAT

Phoneme recognition network	Phoneme recognition rate (%)
MDNN	69
SA MDNN with nonlinear normalization method	69.4
SA MDNN with model adaptation method	70

7 Conclusions

In this article, a growing modular deep neural network for continuous speech recognition is proposed. The special structure of this network and the suggested pre-training procedure for that, make it efficient for both aspects of speech recognition accuracy and training cost. By expanding this network into a double spatiotemporal architecture, it can be trained sequentially to learn the phonetic sequences subspace. Therefore, this sequentially trained network improves the recognition accuracy. As far as we know, it is the first time that both sequential and acoustic information are trained to an integrated network. Afterward, to improve the recognition performance of the proposed network, two speaker adaptation methods motivated by a special mode of signal processing in the human brain are proposed. The first method relies on nonlinear normalization of speech representations iteratively. In the second method, the phone recognition model adapts its decision regions based on the information extracted from a separate speaker recognition model. The proposed MDNN structure and the strategies proposed for developing it are examined on two Persian speech datasets: FARSDAT with frame-level phonetic transcription and Large FARSDAT without that. Since there is no frame-level phonetic transcription for Large FARSDAT, a semi-supervised learning algorithm is suggested to train the MDNN and its extended structure on this dataset. Experimental results have shown that all of the developmental strategies are effective for a continuous speech recognition task on both FARSDAT and Large FARSDAT.

In future works, we would like to develop the deep modular neural network such that it can recognize syllables and words in an integrative manner when it moves on the speech signal. Due to the complexities that these speech units have, besides growing the network, some additive modules that feed the grown network with useful information seems to be required. Moreover, it appears that in the second adaptation method, speech recognition network also has an influence on the speaker recognition results. So, for future works, there can be a modification in this method that strongly will have a significant effect on speech recognition accuracy.

Acknowledgments The research was supported by the Iranian Cognitive Sciences and Technologies Council with the grant code 209.

References

1. Baker J, Deng L, Glass J, Khudanpur S, Lee C, Morgan N et al (2009) Research developments and directions in speech recognition and understanding, Part 1. IEEE Signal Process Mag 26(3):75–80
2. Baker J, Deng L, Khudanpur S, Lee C, Glass J, Morgan N et al (2009) Updated MINDS report on speech recognition and understanding, Part 2. IEEE Signal Process Mag 26(4):78–85
3. Dahl G, Yu D, Deng L, Acero A (2012) Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. IEEE Trans Audio Speech Lang Process 20(1):30–42
4. Kapadia S, Valtchev V, Young S J (1993) MMI training for continuous phoneme recognition on the TIMIT database. In: Proceedings of ICASSP
5. Juang B-H, Hou W, Lee C-H (1997) Minimum classification error rate methods for speech recognition. IEEE Trans Audio Speech Lang Process 5(3):257–265
6. McDermott E, Hazen TJ, Le Roux J, Nakamura A, Katagiri S (2007) Discriminative training for large-vocabulary speech recognition using minimum classification error. IEEE Trans Audio Speech Lang Process 15(1):203–223
7. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Audio Speech Lang Process 37(3):328–339
8. Morgan N, Bourlard H (1990) Continuous speech recognition using multilayer perceptrons with hidden Markov models. In: Proceedings of ICASSP, pp 413–416
9. Seyyedsalehi SA (1995) Continuous Persian speech recognition using functional model of human brain in speech perception. PhD dissertation. Technical Faculty. Tarbiyat Modarres University (in Persian)
10. Bourlard H, Morgan N (1993) Continuous speech recognition by connectionist statistical methods. IEEE Trans Neural Netw 4(6):893–909
11. Bourlard HA, Morgan N (1994) Connectionist speech recognition: a hybrid approach. Springer 247
12. Trentin E, Gori M (2001) A survey of hybrid ANN/HMM models for automatic speech recognition. Neurocomputing 37(1):91–126
13. Seide F, Li G, Yu D (2011) Conversational speech transcription using context-dependent deep neural networks. In: Proceedings of INTERSPEECH, pp 437–440
14. Jaitly N, Nguyen, P, Senior A.W, Vanhoucke V (2012) Application of pretrained deep neural networks to large vocabulary speech recognition. In: INTERSPEECH
15. Mohamed AR, Dahl GE, Hinton G (2012) Acoustic modeling using deep belief networks. IEEE Trans Audio Speech Lang Process 20(1):14–22
16. Yu D, Deng L, Seide F (2013) The deep tensor neural network with applications to large vocabulary speech recognition. IEEE Trans Audio Speech Lang Process 21(2):388–396
17. Hinton G, Deng L, Yu D, Dahl GE, Mohamed AR, Jaitly N et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97

18. Deng L, Yu D, Platt J (2012) Scalable stacking and learning for building deep architectures. In: Proceedings of ICASSP, pp 2133–2136
19. Abdel-Hamid O, Mohamed A.R, Jiang H, Penn G (2012) Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition. In: Proceedings of ICASSP, pp 4277–4280
20. Andrew G, Bilmes J (2012) Sequential deep belief networks. In: Proceedings of ICASSP, pp 4265–4268
21. Hinton G, Osindero S, Teh Y (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554
22. Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layer-wise training of deep networks. In: Proceedings of NIPS, pp 153–160
23. Abdel-Hamid O, Deng L, Yu D (2013) Exploring convolutional neural network structures and optimization techniques for speech recogntiion. In: Proceedings of INTERSPEECH
24. Sainath T, Kingsbury B, Soltau H, Ramabhadran B (2013) Optimization techniques to improve training speed of deep neural networks for large speech tasks. IEEE Trans Audio Speech Lang Process 21(11):2267–2276
25. Mohamed AR, Yu D, DengL (2010) Investigation of full-sequence training of deep belief networks for speech recognition. In: INTERSPEECH, pp 2846–2849
26. He X, Deng L, Chou W (2008) Discriminative learning in sequential pattern recognition. IEEE Signal Process Mag 25(5):14–36
27. Kingsbury B (2009) Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In: Proceedings of ICASSP, pp 3761–3764
28. Prabhavalkar R, Fosler-Lussier E (2010, March) Backpropagation training for multilayer conditional random field based phone recognition. In: Proceedings of ICASSP, pp 5534–5537
29. Lewandowski N, Droppo J, Seltzer M, Yu D (2014) Phone sequence modeling with recurrent neural networks. In: Proceedings of ICASSP
30. Trmal I (2011) Spatio-temporal structure of feature vectors in neural network adaptation. PhD dissertation, Faculty of applied sciences, University of West Bohemia
31. Gemello R, Mana F, Scanzio S, Laface P, De Mori R (2007) Linear hidden transformations for adaptation of hybrid ANN/HMM models. Speech Commun 49(10):827–835
32. Li B, Sim, KC (2010) Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems. In: INTERSPEECH, pp 526–529
33. Shaofei X, Abdel-Hamid O, Jiang H, Dai L, Liu Q (2012) Fast adaptation of deep neural networks based on discriminant codes for speech recognition. IEEE Trans Audio Speech Lang Process 22(12):1713–1725
34. Karimi K (2003) Implementing the speaker features for quality improvement of the speech recognition models. MS thesis. Faculty of biomedical engineering. Amirkabir University of Technology
35. Pan J, Liu C, Wang Z, Hu Y, Jiang H (2012) Investigation of deep neural networks (DNN) for large vocabulary continuous speech recognition: why DNN surpasses GMMS in acoustic modeling. In: ISCSLP, pp 301–305
36. Seide F, Li G, Chen X, Yu D (2011) Feature engineering in context-dependent deep neural networks for conversational speech transcription. In: IEEE workshop on proceedings of ASRU, pp 24–29
37. Nejadgholi I, Seyyedsalehi SA (2009) Nonlinear normalization of input patterns to speaker variability in speech recognition neural networks. Neural Comput Appl 18(1):45–55
38. Seyyedsalehi SA (2003) Designing a neural network based speech recognition system for Persian LVCSR task. Technical Report. Research Center of Intelligent Signal Processing (RCISP) (in Persian)
39. Sainath TN, Kingsbury B, Ramabhadran B, Fousek P, Novak P, Mohamed AR (2011) Making deep belief networks effective for large vocabulary continuous speech recognition. In: IEEE workshop on proceedings of automatic speech recognition and understanding (ASRU), pp 30–35
40. Koerner E, Gewaltig M, Koerner U, Richter A, Rodemann T (1999) A model of computation in neocortical architecture. Neural Netw 12:989–1005
41. Koerner E, Tsujino H, Masutani T (1997) A cortical type modular neural network for hypothetical reasoning. Neural Netw 10:791–814
42. Lang KJ, Waibel AH, Hinton GE (1990) A time-delay neural network architecture for isolated word recognition. Neural Netw 3(1):23–43
43. Chen B, Zhu Q, Morgan N (2005) Tonotopic multilayer perceptron a neural network for learning long term temporal features for speech recognition. In: Proceedings of ICASSP, pp 945–948
44. Behbood H, Seyyedsalehi SA, Tohidypour HR, Najafi M, Gharibzadeh sh (2012) A novel neural-based model for acoustic-articulatory inversion mapping. Neural Comput Appl 21(5):935–943
45. Seyyedsalehi SZ, Seyyedsalehi SA (2014) Simultaneous learning of nonlinear manifolds based on the bottleneck neural network. Neural Process Lett 40(2):191–209
46. Ansary L, Seyyedsalehi SA (2004) Modeling phones coarticulation effects in a neural network based speaker recognition system. In: Proceedings of Interspeech
47. Bijankhan M, Sheikhzadegan J, Roohani MR, Samareh Y, Lucas C, Tebyani M (1994) FARSDAT-the speech database of Farsi spoken language. In: Proceedings of the Australian conference on speech science and technology, vol 2, pp 826–830
48. Ghayoomi M, Momtazi S, Bijankhan M (2010) A study of corpus development for Persian. Int J Asian Lang Process 20(1):17–33
49. Rahiminejad M (2002) Development and enhancement of current feature extraction methods in speech recognition systems. MS Thesis. Faculty of Biomedical Engineering. Amirkabir University of Technology (in Persian)
50. Pinto J, Garimella S, Hermansky H, Bourlard H (2011) Analysis of MLP-based hierarchical phoneme posterior probability estimator. IEEE Trans Audio Speech Lang Process 19(2):225–241
51. Makki B, Noori Hosseini M, Seyyedsalehi SA (2010) An evolving neural network to perform dynamic principal component analysis. Neural Comput Appl 19(3):459–463
52. Makki B, Noori Hosseini M, Seyyedsalehi SA, Sadati N (2010) Unaligned training for voice conversion based on a local nonlinear principal component analysis approach. Neural Comput Appl 19(3):437–444
53. Tohidypour HR, Seyyedsalehi SA, Behbood H, Roshandel H (2012) A new representation for speech frame recognition based on redundant wavelet filter banks. Speech Commun 54(2):256–271
54. Sameti H, Veisi H, Bahrani M, Babaali B, Hosseinzadeh Kh (2009) Nevisa, a persian continuous speech recognition system. Advances in Computer Science and Engineering. Springer, Berlin, pp 485–492
55. Karami Sh (2000) Implementing information existing in transition regions of phoneme borders to improve capability of speaker independent neural network based speech recognition systems. MS Thesis. Faculty of biomedical engineering. Amirkabir University of Technology (in Persian)
56. HTK (v.3.4), Hidden Markov Model Toolkit: http://htk.eng.cam.ac.uk