Hindawi Publishing Corporation
Journal of Robotics
Volume 2015, Article ID 797083, 7 pages
http://dx.doi.org/10.1155/2015/797083

Hindawi

Research Article

A Fast Learning Method for Multilayer Perceptrons in Automatic Speech Recognition Systems

Chenghao Cai,[1] Yanyan Xu,[2] Dengfeng Ke,[3] and Kaile Su[4]

[1]*School of Technology, Beijing Forestry University, No. 35 Qinghua Dong Road, Haidian District, Beijing 100083, China*
[2]*School of Information Science and Technology, Beijing Forestry University, No. 35 Qinghua Dong Road, Haidian District, Beijing 100083, China*
[3]*Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancun Dong Road, Haidian District, Beijing 100190, China*
[4]*Institute for Integrated and Intelligent Systems, Griffith University, 170 Kessels Road, Nathan, Brisbane, QLD 4111, Australia*

Correspondence should be addressed to Yanyan Xu; xuyyxu@gmail.com

Received 21 August 2014; Revised 4 January 2015; Accepted 12 February 2015

Academic Editor: Jorge Dias

We propose a fast learning method for multilayer perceptrons (MLPs) on large vocabulary continuous speech recognition (LVCSR) tasks. A preadjusting strategy based on separation of training data and dynamic learning-rate with a cosine function is used to increase the accuracy of a stochastic initial MLP. Weight matrices of the preadjusted MLP are restructured by a method based on singular value decomposition (SVD), reducing the dimensionality of the MLP. A back propagation (BP) algorithm that fits the unfolded weight matrices is used to train the restructured MLP, reducing the time complexity of the learning process. Experimental results indicate that on LVCSR tasks, in comparison with the conventional learning method, this fast learning method can achieve a speedup of around 2.0 times with improvement on both the cross entropy loss and the frame accuracy. Moreover, it can achieve a speedup of approximately 3.5 times with only a little loss of the cross entropy loss and the frame accuracy. Since this method consumes less time and space than the conventional method, it is more suitable for robots which have limitations on hardware.

1. Introduction

Pattern recognition is one of the most important topics on humanoid robots. To make robots have capabilities of communicating with and learning from the realistic world, recognizing information such as speeches and images is needed. There is much former relevant work. For instance, methods of speech recognition have been used for facilitating interactions between human and humanoid robots for more than ten years [1]. An automated speech recognisor, which has relatively better performance on separating sentences and reducing noises than before, has been then applied to robots [2]. Besides, methods of image recognition have been widely applied to such humanoid robots. A classic example is the use of the robotic vision, such as gesture recognition to realize the direct commanding from humans to robots [3, 4].

However, there are some problems restricting the application of such methods to robots, the chief among which is that the recognising results are not satisfying. Fortunately, deep neural networks (DNNs) can resolve this problem to a great degree. DNNs were first successfully applied to image recognition, bringing evident improvement on the recognition performance [5]. Then they have been used in speech recognition, especially in LVCSR tasks, over the past few years. Former work reveals that automatic speech recognition (ASR) systems based on context dependent Gaussian mixture models (CD-GMMs) and hidden Markov models (HMMs) are improved by replacing GMMs with DNNs [6–10]. Moreover, new usages of DNNs are proposed in recent work [11–18].

An MLP based on a supervised BP learning algorithm is one of the widely used DNNs in ASR systems. However, learning is difficult in the MLP due to the heavy computational burdens of densely connected structures, multilayers, and several epochs of iterations, and thus it requires considerably long time to achieve an essential recognition accuracy.

Another drawback of DNNs is that it is hard to decode them as the decoding processes also entail a large amount of time.

Some methods have been proposed to ameliorate these disadvantages. Since graphics processing units (GPUs) have powerful abilities on parallel computations, they have been used to improve the speed of computing matrix multiplications in regard to the dense weight matrices of MLPs [19]. Meanwhile, asynchronous training algorithms have been applied to the training processes, making several computers or processing units work asynchronously so that the training tasks were allocated to parallel simultaneous jobs [20–22]. Moreover, Hessian-free (HF) optimisation focuses on reducing the number of iterations, which makes parameters converge faster than conventional stochastic gradient descent (SGD) [23–25]. Nevertheless, the heavy computational burdens of learning MLPs still exist, especially on realistic tasks that demand markedly sufficient learning to improve the recognition accuracy. To speed up the decoding processes, SVD is used to restructure the models, but it requires extra time for retraining and once again increases the time consumption [26, 27].

In this paper, we propose a fast learning method, reducing the computational burdens of learning MLPs and decoding them. The basic concept of this method is to preadjust roughly the initial MLP and then train the MLP using an unconventional BP algorithm after restructuring weight matrices via SVD. The preadjusting process alters the distributions of singular values before the MLP is accurately trained. Since SVD reduces the dimensionality of weight matrices, the burdens of computing matrix multiplications are lessened.

The rest of this paper is organized as follows. Section 2 describes the fast learning method. Section 3 shows experimental results and discussions and in Section 4 we draw conclusions.

2. A Fast Learning Method

2.1. A Learning Strategy for the First Epoch. The basic concept of this strategy is to roughly train the MLP before accurate learning. Concretely, it goes through all of the training data only once during the first epoch, using the conventional BP algorithm. During this epoch, the frame accuracy of the MLP is heightened as far as possible.

This strategy first separates averagely the training data into T bunches. When training with the ith bunch data, a dynamically declining learning rate is used, which is

$$\epsilon(i) = \epsilon_0 \alpha^{i-1} \quad (i = 1, 2, 3, \ldots, T), \tag{1}$$

where ϵ_0 denotes the initial learning rate and $0 < \alpha < 1$. The proportions of these bunches are different, observing a rule based on the cosine function. The proportion of the ith bunch is

$$p(i) = \frac{\pi}{2T} \cdot \cos\left(\frac{\pi}{2T} \cdot i\right) \quad (i = 1, 2, 3, \ldots, T-1). \tag{2}$$

Particularly, to ensure that the rest of data are contained in the last bunch, the proportion of the Tth bunch is

$$p(T) = 1 - \sum_{i=1}^{T-1} p(i). \tag{3}$$

In fact, $\sum_{i=1}^{T-1} p(i)$ converges to 1 when T tends to positive infinity, because

$$\lim_{T \to +\infty} \sum_{i=1}^{T-1} \frac{\pi}{2T} \cdot \cos\left(\frac{\pi}{2T} \cdot i\right) = \int_0^{\pi/2} \cos\beta \, d\beta = \left[\sin\beta\right]_0^{\pi/2} = 1. \tag{4}$$

It ensures that all bunches observe the rule of the cosine function and all data are used when T tends to positive infinity. Nonetheless, it is impossible to let T tend to positive infinity in reality, so T is set to a big positive integer practically. We particularly name this strategy as preadjusting (PA), as the learning-rates and data arrangement are different from those conventional training methods.

The dynamic declining learning-rate is used due to the fact that the PA process requires going through the training data once and achieving heightened accuracies as far as possible. Relatively high learning-rates learn models effectively, but low precision exists, whereas relatively low learning-rates learn MLPs slowly but achieve high recognition accuracies. In (1), the initial learning-rate is high, facilitating the learning speed at the beginning, and, then, α^{i-1} decays this rate exponentially, ensuring the precisions of the intermediate and last learning.

2.2. A BP Algorithm Based on Weight Matrix Restructuring

2.2.1. Weight Matrix Restructuring and Training. An MLP consists of an input layer, several hidden layers, and an output layer. Except the input layer that obtains states directly from input vectors, each of the other layers uses a weight matrix, a set of biases, and an activation function to compute states. The computational burdens are mainly due to the weight matrices. Concretely, both forward and backward computations demand the products of weight matrices and various vectors; thus the time complexity of the MLP is determined by the dimensionality of weight matrices.

SVD is one of the basic and important analysis methods in linear algebra [28], which can be used to reduce the dimensionality of matrices and has the following equation [26, 27]:

$$\begin{aligned} \mathrm{SVD}\left(\mathbf{W}_{(\mathbf{m}\times\mathbf{n})}\right) &= \mathbf{U}_{(\mathbf{m}\times\mathbf{m})} \cdot \mathbf{\Sigma}_{(\mathbf{m}\times\mathbf{n})} \cdot \mathbf{V}^{\mathbf{T}}_{(\mathbf{n}\times\mathbf{n})} \\ &\approx \mathbf{U}_{(\mathbf{m}\times\mathbf{l})} \cdot \left(\mathbf{\Sigma}_{(\mathbf{l}\times\mathbf{l})} \cdot \mathbf{V}^{\mathbf{T}}_{(\mathbf{n}\times\mathbf{n})}\right) \\ &= \mathbf{W}_{\mathbf{1}(\mathbf{m}\times\mathbf{l})} \cdot \mathbf{W}_{\mathbf{2}(\mathbf{l}\times\mathbf{n})}, \end{aligned} \tag{5}$$

where the numbers in "( )" stand for dimensions, $\mathbf{W}_{(\mathbf{m}\times\mathbf{n})}$ stands for an $m \times n$ weight matrix, $\mathbf{U}_{(\mathbf{m}\times\mathbf{m})}$, $\mathbf{\Sigma}_{(\mathbf{m}\times\mathbf{n})}$, and $\mathbf{V}^{\mathbf{T}}_{(\mathbf{n}\times\mathbf{n})}$ stand for three matrices generated by SVD, $\mathbf{W}_{\mathbf{1}(\mathbf{m}\times\mathbf{l})}$ and $\mathbf{W}_{\mathbf{2}(\mathbf{l}\times\mathbf{n})}$ stand for two new obtained weight matrices, and

$l < \max(m, n)$ stands for the number of kept singular values. The time complexity of computing a product of $\mathbf{W_{(m\times n)}}$ and a vector $\mathbf{v_{(n)}}$ is originally $O(m \times n)$. By replacing $\mathbf{W_{(m\times n)}} \cdot \mathbf{v_{(n)}}$ with $\mathbf{W_{1(m\times l)}} \cdot (\mathbf{W_{2(l\times n)}} \cdot \mathbf{v_{(n)}})$, the time complexity is reduced to $O((m + n) \times l)$ when $l < m \times n/(m + n)$. Since the effectiveness of SVD, to some extent, depends on the meaningful parameters of weight matrices, the SVD-based method is arranged after preadjusting. In other words, SVD is meaningless to stochastic weight matrices which have not learned anything.

To simplify the discussion, consider a single layer. Let $\mathbf{b_{(m)}}$ denote an m-dimensional set that contains m biases, and $\varphi(x)$ denotes an activation function. The forward computation transforms an n-dimensional input vector $\mathbf{i_{(n)}}$ to an m-dimensional output vector $\mathbf{o_{(m)}}$ by

$$\mathbf{o_{(m)}} = \varphi\left(\mathbf{W_{1(m\times l)}} \cdot \left(\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}}\right) + \mathbf{b_{(m)}}\right). \tag{6}$$

Since the weight matrices are unfolded, the backward computation is required to fit the doubled matrix structure. Let $\mathbf{e_{(m)}}$ stand for a received error signal, $\varphi'(x)$ for the derivative of the activation function, $\boldsymbol{\delta}_{\mathbf{(m)}}$ for a gradient, $\mathbf{e_{(n)}}$ for an error signal that will be transmitted to the beneath layer, $\Delta\mathbf{b_{(m)}}$, $\Delta\mathbf{W_{1(m\times l)}}$, and $\Delta\mathbf{W_{2(l\times n)}}$ for the deltas, and ϵ for a learning-rate. According to the BP theory, the gradient is

$$\boldsymbol{\delta}_{\mathbf{(m)}} = \varphi'\left(\mathbf{W_{1(m\times l)}} \cdot \left(\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}}\right) + \mathbf{b_{(m)}}\right) \cdot \mathbf{e_{(m)}}. \tag{7}$$

The update rule of $\mathbf{b_{(m)}}$ is

$$\Delta\mathbf{b_{(m)}} = \epsilon \cdot \boldsymbol{\delta}_{\mathbf{(m)}}. \tag{8}$$

The update rule of $\mathbf{W_{1(m\times l)}}$ is

$$\Delta\mathbf{W_{1(m\times l)}} = \epsilon \cdot \boldsymbol{\delta}_{\mathbf{(m)}} \cdot \left(\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}}\right)^T. \tag{9}$$

The error signal becomes $\mathbf{W^T_{1(m\times l)}} \cdot \boldsymbol{\delta}_{\mathbf{(m)}}$ through $\mathbf{W_{1(m\times l)}}$; thus, the update rule of $\mathbf{W_{2(l\times n)}}$ is

$$\Delta\mathbf{W_{2(l\times n)}} = \epsilon \cdot \mathbf{W^T_{1(m\times l)}} \cdot \boldsymbol{\delta}_{\mathbf{(m)}} \cdot \mathbf{i^T_{(n)}}. \tag{10}$$

The error $\mathbf{e_{(n)}}$ is

$$\begin{aligned} \mathbf{e_{(n)}} &= \left(\mathbf{W_{1(m\times l)}} \cdot \mathbf{W_{2(l\times n)}}\right)^T \cdot \boldsymbol{\delta}_{\mathbf{(m)}} \\ &= \mathbf{W^T_{2(l\times n)}} \cdot \left(\mathbf{W^T_{1(m\times l)}} \cdot \delta_{\mathbf{(m)}}\right). \end{aligned} \tag{11}$$

Algorithm 1 (the weight-matrix-restructuring-based BP algorithm).

Input. $\mathbf{W_{1(m\times l)}} \in \mathbb{R}^{m\times l}$, $\mathbf{W_{2(l\times n)}} \in \mathbb{R}^{l\times n}$, $\mathbf{b_{(m)}} \in \mathbb{R}^m$, and $\mathbf{i_{(n)}} \in \mathbb{R}^n$, $\varphi(x)$, $\epsilon \in \mathbb{R}$.

Output. $\Delta\mathbf{W_{1(m\times l)}} \in \mathbb{R}^{m\times l}$, $\Delta\mathbf{W_{2(l\times n)}} \in \mathbb{R}^{l\times n}$, $\Delta\mathbf{b_{(m)}} \in \mathbb{R}^m$, and $\mathbf{e_{(n)}} \in \mathbb{R}^n$.

(1) $\mathbf{o_{(m)}} \leftarrow \varphi(\mathbf{W_{1(m\times l)}} \cdot (\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}}) + \mathbf{b_{(m)}})$.

(2) Obtain an error signal $\mathbf{e_{(m)}}$.

(3) $\boldsymbol{\delta}_{\mathbf{(m)}} \leftarrow \varphi'(\mathbf{W_{1(m\times l)}} \cdot (\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}}) + \mathbf{b_{(m)}}) \cdot \mathbf{e_{(m)}}$.

(4) $\Delta\mathbf{W_{1(m\times l)}} \leftarrow \epsilon \cdot \boldsymbol{\delta}_{\mathbf{(m)}} \cdot (\mathbf{W_{2(l\times n)}} \cdot \mathbf{i_{(n)}})^T$.

(5) $\Delta\mathbf{W_{2(l\times n)}} \leftarrow \epsilon \cdot \mathbf{W^T_{1(m\times l)}} \cdot \boldsymbol{\delta}_{\mathbf{(m)}} \cdot \mathbf{i^T_{(n)}}$.

(6) $\Delta\mathbf{b_{(m)}} \leftarrow \epsilon \cdot \boldsymbol{\delta}_{\mathbf{(m)}}$.

(7) $\mathbf{e_{(n)}} \leftarrow \mathbf{W^T_{2(l\times n)}} \cdot (\mathbf{W^T_{1(m\times l)}} \cdot \boldsymbol{\delta}_{\mathbf{(m)}})$.

Algorithm 1 illustrates the training process based on weight matrix restructuring. Step (1) is the forward computation. In step (2), the error signal is obtained. In steps (3), (4), (5), and (6), update $\mathbf{W_{1(m\times l)}}$, $\mathbf{W_{2(l\times n)}}$, and $\mathbf{b_{(m)}}$. In step (7), transmit the error to the beneath layer.

After being trained by this algorithm, the final weight matrices can be inversely converted to the original structure via

$$\mathbf{W_{(m\times n)}} = \mathbf{W_{1(m\times l)}} \cdot \mathbf{W_{2(l\times n)}}. \tag{12}$$

Nonetheless, it is not necessary to convert them to the original structure unless being seriously demanded, because converting inversely does not improve the recognition accuracy but increases the computational burdens of recognition.

2.2.2. The Complexity Reduction Theorem. As previously mentioned, the SVD-based method reduces the time complexities of matrix multiplications, which is summarized by the following theorem.

Theorem 2. *Assume that* $\mathbf{W}$ *is an* $m \times n$ *weight matrix and* $\mathbf{i}$ *is an* n*-dimensional vector. By applying the SVD-based method on* $\mathbf{W}$ *and keeping* l *largest singular values, the time complexity of computing* $\mathbf{W} \cdot \mathbf{i}$ *is reduced from* $O(m \times n)$ *to* $O((m + n) \times l)$*, when* $l < m \times n/(m + n)$.

Proof. Computing $\mathbf{W} \cdot \mathbf{i}$ requires $m \times n$ times of real number multiplications, so the time complexity of computing $\mathbf{W} \cdot \mathbf{i}$ is $O(m \times n)$. Apply the SVD method on $\mathbf{W}$ and obtain $\mathbf{W_1}$ and $\mathbf{W_2}$. After replacing $\mathbf{W}$ by $\mathbf{W_1} \cdot \mathbf{W_2}$, $\mathbf{W} \cdot \mathbf{i}$ is replaced by $(\mathbf{W_1} \cdot \mathbf{W_2}) \cdot \mathbf{i}$. According to the associative law, we obtain

$$\left(\mathbf{W_1} \cdot \mathbf{W_2}\right) \cdot \mathbf{i} = \mathbf{W_1} \cdot \left(\mathbf{W_2} \cdot \mathbf{i}\right). \tag{13}$$

Computing $\mathbf{W_2} \cdot \mathbf{i}$ requires $l \times n$ times of real number multiplications and gets an l-dimensional vector. Computing the product of $\mathbf{W_2}$, the l-dimensional vector requires $m \times l$ times of real number multiplications, so $\mathbf{W_1} \cdot (\mathbf{W_2} \cdot \mathbf{i})$ requires $(m + n) \times l$ times of real number multiplications. The number of real number multiplications is reduced when

$$m \times l + l \times n < m \times n, \tag{14}$$

and we obtain

$$l < \frac{m \times n}{(m + n)}. \tag{15}$$

Therefore, the time complexity is reduced from $O(m \times n)$ to $O((m + n) \times l)$, when $l < m \times n/(m + n)$. □

The time complexities of learning MLPs are reduced to $O((m + n) \times l)$ via (5), (7), (9), (10), and (11), when $l < m \times n/(m + n)$, so the computational burdens are eased in comparison with the conventional training algorithm.

3. Experiments

3.1. Experimental Settings. We conduct experiments of LVCSR tasks on a server with 4 Intel Xeon E5-2620 CPUs and 512 GB memory. The training of MLPs is accelerated by an NVIDIA GeForce GTX TITAN Black graphics card. We use hours (h) of speech databases and their transcriptions to train and test acoustic models. The training data contain a 120 h speech database and the testing data contain a 3 h speech database. The texts of the testing data contain 17,221 words. The language model used is a 5-gram ARPA model.

First, GMMs must be trained before replacing GMMs by MLPs. To obtain GMMs, we use Mel-frequency cepstral coefficients (MFCCs) as the features of speeches and then train monophone, triphone, linear discriminant analysis (LDA), and maximum likelihood linear transformation (MLLT) in turn.

Then, MLPs are trained on the basis of GMMs. Feature-space maximum likelihood linear regression (FMLLR) is used as features of speeches for training MLPs. Alignments from GMMs are used as labels of supervised learning. Each MLP has an input layer, five hidden layers, and an output layer. The input layer has 440 units, corresponding to 440-dimensional input vectors. More specifically, each vector contains 40 real numbers that are the features of the corresponding frame of speeches and $40 \times (5 + 5)$ real numbers that are the features of 5 frames before this frame and 5 frames after this frame. Each hidden layer has 1024 units. Sigmoid is chosen as the activation function of the hidden layers. The output layer has 1952 units. To deal with multiclassification problems in ASR systems, softmax is chosen as the activation function of the output layer. All parameters of these layers, including weight matrices and biases, are stochastically initialized. The conventional method and the PA strategy are used to train this initial stochastic MLP, respectively. The number of bunches (T) is set to 20. For the conventional task, the data are averagely separated into bunches, and the learning-rate is set to 0.032. For the PA task, the data are separated by (2) and (3). The initial learning-rate ϵ_0 is set to 0.032 and α in (1) is set to 0.975.

Next, the SVD-based matrix restructuring method is applied to the basic model, keeping 384, 256, and 128 of the largest singular values, respectively. Since the input layer has 440 units, applying the SVD-based method to the first weight matrix will not evidently decrease the time complexity. Therefore, the SVD-based method will not be applied to the first weight matrix, but to all of the other matrices, including the one of the output layer. The structure of the model which keeps 256 singular values is shown in Figure 1 as an example, where the bottleneck means the linear transform. The reason of the numbers of kept largest singular values being set to 384($1024 \times 3/8$), 256($1024 \times 1/4$), and 128($1024 \times 1/8$), respectively, is that the time complexity is reduced when $l < m \times n/(m + n)$, and therefore $l < 512$ if $m = n = 1024$. After that, the BP algorithm illustrated in Section 2.2 is used to train the restructured models. The learning-rates of iterations are decayed from an initial value: when the increment of the frame accuracy on cross validation (The frame accuracy is equal to $(N_{\text{correct}}/N_{\text{total}}) \times 100$, where N_{correct} denotes the number of correct recognized states on softmax and N_{total} denotes the total number of states.) is not smaller than 0.5, the learning-rate does not change, but when the increment of the frame accuracy on cross validation is smaller than 0.5, the learning-rate is halved. The initial learning-rate is set to 1×10^{-5}.

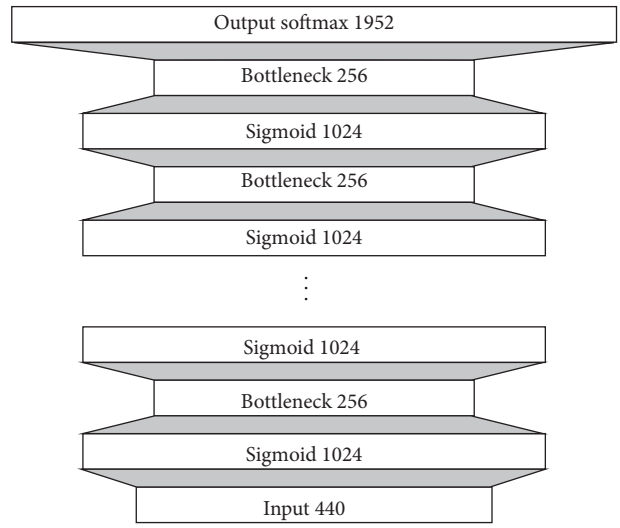

Figure 1: A model restructured by the SVD method.

In these experiments, the cross entropy losses and the frame accuracies on cross validation are used to appraise the performance of MLPs. The word error rate (WER) is used to assess the performance of final CD-MLP-HMMs, which is equal to the number of misrecognized words divided by the total number of words.

3.2. Results and Discussions. Figure 2 shows the changes of the cross entropy loss during the first epoch. The curves of the PA task and the conventional task are provided. Both of them first drop sharply, followed by slight decreases after training by 8 bunches. However, the PA task drops more significantly when training by the first 8 bunches, after which it remains stable. By contrast, the cross entropy loss of the conventional task keeps decreasing when training, but finally it is still higher than that of the PA task, which is because the first 8 bunches on the PA task contain more data due to the fact that they are based on the cosine function. Another further contributing factor is that the dynamic learning-rate facilitates the training, which is also the reason why the PA task has a considerable drop when training by the 3rd–7th bunches.

Figure 3 reveals the changes of frame accuracies on cross validation during the first epoch. Combining with Figure 2, we can see that the frame accuracy increases when the cross entropy loss decreases. However, the changes of frame accuracies are more evident. After training by the first 5 bunches, the frame accuracy of the PA task reaches a very high point, whereas the low point of the cross entropy loss occurs after 8 bunches. A similar phenomenon also occurs on the conventional task. More importantly, the final frame

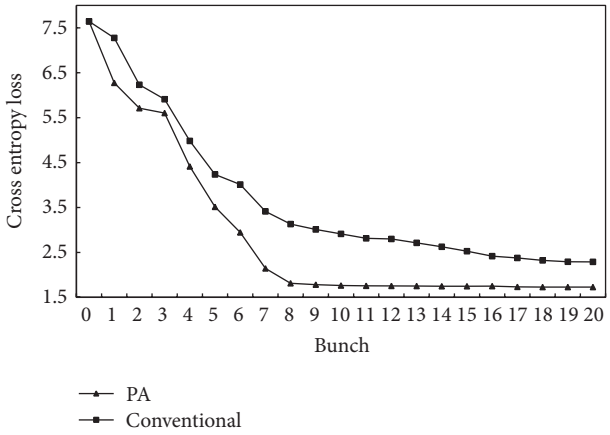

Figure 2: Cross entropy losses during the first epoch.

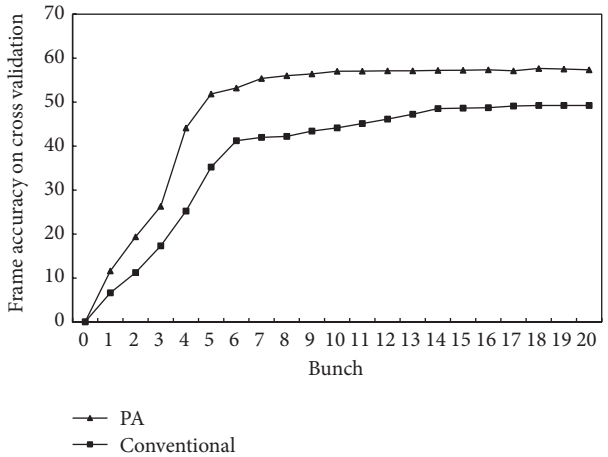

Figure 3: Frame accuracies on cross validation during the first epoch.

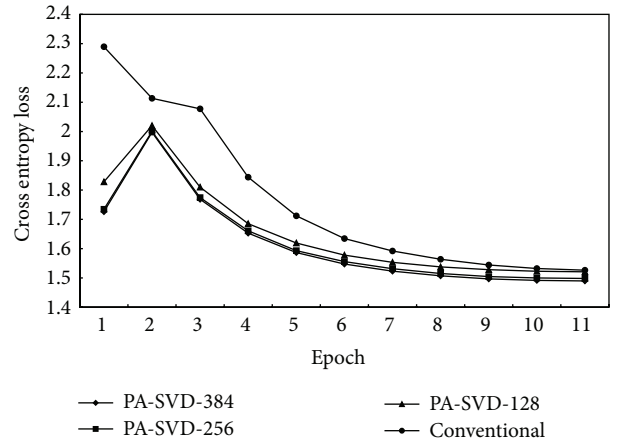

Figure 4: Changes of cross entropy losses.

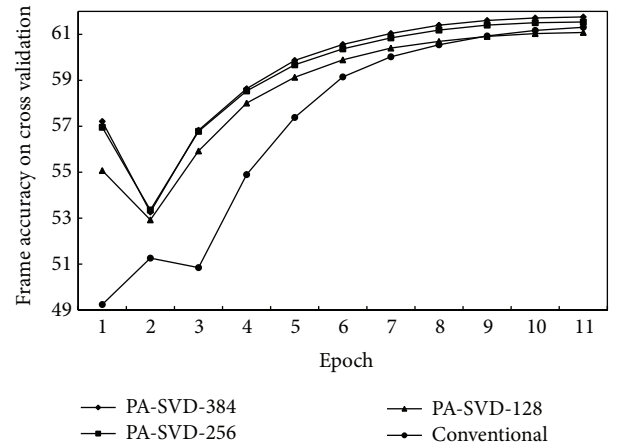

Figure 5: Changes of frame accuracies on cross validation.

accuracy of the PA task is higher than that of the conventional one. Such a high accuracy facilitates the subsequent training, and it is the reason why we use the PA strategy.

A glance at Figure 4 shows some differences on cross entropy losses between the PA-SVD training method and the conventional method. The initial cross entropy loss of the conventional task is significantly higher than those of the PA-SVD tasks due to the fact that the PA strategy has better performance on reducing the cross entropy loss during the first epoch. With regard to the PA-SVD tasks, the initial cross entropy loss is low, and the more bottlenecks mean the lower value. However, the cross entropy losses of the PA-SVD tasks increase during the second epoch, achieving peaks which are dramatically higher than before, which is attributed to the fact that the structures of these models have been altered by the SVD method, and the training algorithm is different from the conventional BP method. After the peaks, marked declines of the cross entropy losses occur to these tasks, followed by sustained decreases. Finally, all of these cross entropy losses become more and more similar to each other. More importantly, the final cross entropy losses of the PA-tasks (PA-SVD-384 and PA-SVD-256) are still slightly lower than that of the conventional task, indicating that the former models have better performance than the latter one.

In fact, on LVCSR tasks, the frame accuracy is more practical, because it directly indicates the proportion of correct recognition results of MLPs. Figure 5 provides the changes of frame accuracies on cross validation. It is easy to note that the initial frame accuracies of PA-SVD tasks are evidently higher than that of the conventional one, which means that the PA strategy improves not only the cross entropy loss (see Figure 4) but also the frame accuracy. Meanwhile, small gaps occur among the three PA-SVD tasks. This phenomenon is attributed to the fact that the SVD method brings loss of information to models, particularly when the number of bottlenecks is small. Then the frame accuracies of the PA-SVD tasks reach minima after the second epoch, and the reason is the same as that of the increasing of the cross entropy loss. After that, the frame accuracies keep increasing till the end of training. With regard to the conventional task, the frame accuracy has a slight decrease during the third epoch, which is because the learning-rate is high during this epoch, and from this point it is halved. Finally, the frame accuracy of the PA-SVD-384 task as well as that of the PA-SVD-256 task

Table 1: WERs and model scales.

Task	NUM-PARA (N_{para})	WER
Conventional	≈6.64M	16.32%
PA-SVD-384	≈4.74M	16.48%
PA-SVD-256	≈3.31M	16.55%
PA-SVD-128	≈1.88M	16.71%

is slightly higher than that of the conventional task, whereas the frame accuracy of the PA-SVD-128 task is a little lower. These results again indicate that the PA-SVD-384 model and the PA-SVD-256 model perform better than the conventional model.

Table 1 provides the final results of the overall LVSCR tasks, including the WERs and the numbers of parameters. It is easy to note that the bigger number of parameters means the lower WER, but the gaps among them are very small. In comparison with the previous results, although the PA-SVD-256 task and the PA-SVD-384 task have higher WERs than the conventional task, they have better cross entropy losses and frame accuracies, which is because WERs not only depend on the performance of MLPs but also are affected by the ARPA models above them. For the same MLP, using different ARPA models will bring different results.

With regard to the complexities, the number of computations (including real number multiplications and additions) for both a forward pass and a backward pass is approximately equal to the number of parameters. During training, the computing is on GPUs, and a forward pass and a backward pass are required. Thus, the time complexity for training is

$$O_{\text{tr}} = \frac{2 \times N_{\text{para}}}{N_{\text{GPU}}}, \tag{16}$$

where N_{GPU} denotes the number of GPU cores which can realistically run parallel (In reality, for some tasks, not all of the GPU cores can work simultaneously, but it is difficult to discuss in this work, as parallel computing is very complicated.). During decoding, only a forward pass is required. The time complexity for decoding is

$$O_{\text{de}} = \frac{N_{\text{para}}}{N_{\text{GPU}}}. \tag{17}$$

In our experiments, the NVIDIA GeForce GTX TITAN Black graphics card, including 2880 GPU cores, is used. Since N_{GPU} is large, the experiments run relatively fast and are finished in a few days. However, the volume of this graphics card is big (26.67 cm × 11.12 cm × 7.44 cm), so it is hard to embed it into a humanoid robot for decoding. If smaller graphics cards or CPUs are used in the robot, it will take considerable longer time for training and decoding. Thus, it is important to reduce the time complexities.

Equations (16) and (17) reveal that the time cost depends on the number of parameters. Revisiting Table 1, we notice that the PA-SVD tasks have significant less time cost than the conventional task, whereas the WERs are almost the same. Particularlly, the PA-SVD-256 task achieves a 2.0 times speedup and the PA-SVD-128 task achieves a 3.5 times speedup, which provides a way for humanoid robots to learn and recognize speech much more efficiently and effectively. Besides, the memories of robots are much smaller than servers, as robots have restrictions on sizes, weights, and powers. It is easy to note that the final models of the PA-SVD tasks have markedly lower numbers of parameters than the conventional model, which consequently also provides a way for robots to reduce their sizes, weights, and consumptions of energy.

4. Conclusions

We propose a fast learning method for MLPs in ASR systems in this paper, which is suitable for humanoid robots whose CPU/GPUs and memories are limited, as its time complexities are low, and the final model sizes are small. First, the PA strategy improves the frame accuracies and the cross entropy losses of the MLP during the first training epoch, based on the cosine function separation of training data and the dynamic learning-rate. The SVD-based method then restructures the weight matrices of the preadjusted MLPs and reduces their dimensionality. After that, the BP algorithm that fits the unfolded weight matrices is used to train the MLP obtained by the SVD restructuring. In the experiments, this method accelerates the training processes to around 2.0 times faster than before with improvements on the cross entropy loss and the frame accuracy, and moreover it accelerates the training processes to around 3.5 times faster than before with just a negligible increase of the cross entropy loss as well as a tiny loss of the frame accuracy.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (YX2014-18), the Beijing Higher Education Young Elite Teacher Project (YETP0768), and the National Natural Science Foundation of China (61472369 and 61103152).

References

[1] C. Breazeal and L. Aryananda, "Recognition of affective communicative intent in robot-directed speech," *Autonomous Robots*, vol. 12, no. 1, pp. 83–104, 2002.

[2] S. Yamamoto, J.-M. Valin, K. Nakadai et al., "Enhanced robot speech recognition based on microphone array source separation and missing feature theory," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '05)*, pp. 1477–1482, April 2005.

[3] K. Nickel and R. Stiefelhagen, "Visual recognition of pointing gestures for human-robot interaction," *Image and Vision Computing*, vol. 25, no. 12, pp. 1875–1884, 2007.

[4] R. Stiefelhagen, C. Fügen, P. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel, "Natural human-robot interaction using

speech, head pose and gestures," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, pp. 2422–2427, October 2004.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] G. E. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 1, no. 6, pp. 82–97, 2012.

[7] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8609–8613, May 2013.

[8] A.-R. Mohamed, G. E. Dahl, and G. E. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[9] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[10] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proceedings of the 13th Annual Conference of the International Speech Communication Association (INTERSPEECH '12)*, pp. 2577–2580, September 2012.

[11] M. D. Zeiler, M. Ranzato, R. Monga et al., "On rectified linear units for speech processing," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 3517–3521, May 2013.

[12] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6645–6649, May 2013.

[13] L. Deng, G. E. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8599–8603, IEEE, Vancouver, Canada, May 2013.

[14] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML '13)*, Atlanta, Ga, USA, June 2013.

[15] D. Yu, L. Deng, and F. Seide, "The deep tensor neural network with applications to large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 2, pp. 388–396, 2013.

[16] H. Su, G. Li, D. Yu, and F. Seide, "Error back propagation for sequence training of Context-Dependent Deep NetworkS for conversational speech transcription," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6664–6668, May 2013.

[17] D. Yu, G. E. Hinton, N. Morgan, J.-T. Chien, and S. Sagayama, "Introduction to the special section on deep learning for speech and language processing," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 4–6, 2012.

[18] N. Morgan, "Deep and wide: multiple layers in automatic speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 7–13, 2012.

[19] Z. Luo, H. Liu, and X. Wu, "Artificial neural network computation on graphic process unit," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '05)*, vol. 1, pp. 622–626, July–August 2005.

[20] J. Dean, G. S. Corrado, R. Monga et al., "Large scale distributed deep networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1232–1240, Lake Tahoe, Nev, USA, December 2012.

[21] G. Heigold, V. Vanhoucke, A. W. Senior et al., "Multilingual acoustic models using distributed deep neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 8619–8623, May 2013.

[22] S. Zhang, C. Zhang, Z. You, R. Zheng, and B. Xu, "Asynchronous stochastic gradient descent for DNN training," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6660–6663, May 2013.

[23] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 735–742, June 2010.

[24] P. L. Dognin and V. Goel, "Combining stochastic average gradient and Hessian-free optimization for sequence training of deep neural networks," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU '13)*, pp. 321–325, December 2013.

[25] J. Martens and I. Sutskever, "Training deep and recurrent networks with Hessian-free optimization," in *Neural Networks: Tricks of the Trade*, vol. 7700 of *Lecture Notes in Computer Science*, pp. 479–535, Springer, Berlin, Germany, 2nd edition, 2012.

[26] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '14)*, pp. 6359–6363, Florence, Italy, May 2014.

[27] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH '13)*, pp. 2365–2369, Lyon, France, 2013.

[28] V. C. Klema and A. J. Laub, "The singular value decomposition: its computation and some applications," *IEEE Transactions on Automatic Control*, vol. 25, no. 2, pp. 164–176, 1980.