Speeding up deep neural network based speech recognition systems

Yeming Xiao, Yujing Si, Ji Xu, Jielin Pan, Yonghong Yan
Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics,
Chinese Academy of Sciences, Beijing, China 100190
Email: {xiaoyeming, siyujing, xuji, jpan, yyan}@hccl.ioa.ac.cn

Abstract—**Recently, deep neural network (DNN) based acoustic modeling has been successfully applied to large vocabulary continuous speech recognition (LVCSR) tasks. A relative word error reduction around 20% can be achieved compared to a state-of-the-art discriminatively trained Gaussian Mixture Model (GMM). However, due to the huge number of parameters in the DNN, real-time decoding is a bottleneck for the DNN based speech recognition systems. In this paper, we adopt several techniques for the speed optimization of the DNN-based system. Specifically, we use singular value decomposition (SVD) to reduce the model parameters, use the SSE instruction sets for the parallel calculation in the data space, and quantize the model parameters reasonably to convert the floating-point arithmetic into fixed-point arithmetic. Besides, taking the characteristics of speech signal into account, we use a frame-skipping method when evaluating the posterior probabilities. Finally, compared to the un-optimized baseline system, with negligible recognition performance loss, the decoding real-time factor of the optimized one is significantly reduced, from 6.1 to 0.31. And this response speed can basically meet the requirement of our real applications.**

Index Terms—**Large Vocabulary Continuous Speech Recognition, Acoustic Modeling, Deep Neural Network, SSE Instructions.**

I. Introduction

RECENTLY, the Deep Neural Network-Hidden Markov Model (DNN-HMM) based acoustic modeling has achieved great success on Large Vocabulary Continuous Speech Recognition (LVCSR) tasks [1], [2]. A relative word error reduction around 20% can be achieved compared to a state-of-the-art discriminatively trained Gaussian Mixture Models (GMM) [3], [4]. Under the DNN-HMM framework, unlike the traditional GMM-HMM framework where GMM is used to model the probability distribution of acoustic features associated with states of an HMM, DNN is used to produce posterior probabilities over HMM states directly. In fact, DNN is a special Artificial Neural Network with many hidden layers, which offer several potential advantages over GMM [5]:

1) Its estimation of the posterior probabilities of HMM states does not require detailed assumptions about the data distribution.
2) It allow an easy way of combining diverse features, including both discrete and continuous features.
3) It use far more data to constrain each parameter because the output of each training case is sensitive to a large fraction of the weights.

Although DNN has been shown superior modeling capability over GMM, the outstanding performance accompanies with huge computation cost due to that DNN has much more parameters than the traditional GMM [6]. Usually, a typical DNN used in ASR has many hidden layers (5 to 9), each equipped with about 2000 $\sim$ 3000 neurons, and a much larger output layer designed to model senones (tri-phone states) directly [7], which results that DNN has 2 to 10 times more than the GMM counterpart. Thanks to the development of Graphics Processing Unit (GPU), the training of DNN has been speeded up significantly [8]–[10]. Nevertheless, due to a variety of factors, GPU is not always available on all types of hardware, which limits the application of DNN in a lot of scenarios. Especially at the decoding stage, where a faster response is demanded under real applications for ASR systems [11].

In this paper, we investigate the optimization of improving the decoding speed under the DNN-HMM based ASR systems. In order to better identify those factors which affect the system's decoding speed, we divide the decoder into three modules: 1) The feature extraction module; 2) The DNN-based posterior probabilities evaluation module; 3) The Viterbi search module. The function of the feature extraction module is to get a compact representation of the original time-domain signal. The calculation in this module is very fast so we skip it; The posterior probabilities evaluation module uses DNN to evaluate the posterior probabilities over HMM states and involves lots of matrix-vector production. According to the analysis of our experiments, the time consumption in this module attributes to 70% of the total decoding time. Therefore, the optimization of the DNN-based posterior probabilities calculation is the core of this paper. The Viterbi searching module tries to find the optimal path in the searching space. Due to the large vocabulary used in most the-state-of-the-art ASR systems, the searching space is also very

Manuscript received August 10, 2011; revised January 2, 2012; accepted April 16, 2012. © 2005 IEEE.

This work is partially supported by the National Natural Science Foundation of China (Nos. 10925419, 90920302, 61072124, 11074275, 11161140319, 91120001), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant Nos. XDA06030100, XDA06030500), the National 863 Program (No. 2012AA012503) and the CAS Priority Deployment Project (No. KGZD-EW-103-2).

large. And this module occupies $20\% \sim 30\%$ of time in total, there is also some room for improvement.

Based on the analysis above, the main contents of this paper are divided into two parts: the acceleration of the DNN-based posterior probability calculation; and the acceleration of the searching process with the combination of the inherent characteristics of the speech recognition task. The starting point of all of the acceleration techniques is to improve the decoding speed of the ASR system without much recognition performance degradation [6].

The rest of this paper is organized as follows. In section II, we review the framework of the DNN-HMM based ASR system. In section III, we describe the acceleration techniques for the DNN-based posterior probability evaluation. Section IV presents the optimization method during Viterbi searching. Experiments and analysis are given in section V. Finally, we draw the conclusions in section VI.

II. The Framework of the DNN-HMM based ASR system

The DNN-HMM [3] [12] is a special ANN-HMM hybrid system in which the DNN is in place of the the shallow ANN and is used to model the posterior probabilities of the HMM states directly. By combining the discriminative modeling power of DNN with the sequential modeling capability of HMM, DNN-HMM outperforms the traditional GMM-HMM significantly.

The basic architecture of the DNN-HMM based ASR system can be illustrated as in Figure 1: The original speech signals are converted into acoustic features by a feature extraction module. The DNN accepts an input pattern x, which typically consists of 7-13 frames of those acoustic features, and passes it through many layers of nonlinear transformation [13]

$$y_i^l = \sigma((w_{i,*}^l)^T y^{l-1} + b_i^l) \quad (1)$$

where w^l and b^l are the weight matrix and bias at layer l, $w_{i,*}^l$ is the i-th row of w and b_i^l is the i-th elements of b, y_i^l denote and the activation of the i-th neuron at layer l. When the l-th layer is an intermediate layer, $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function. and when it is the output layer, then $\sigma(x)$ is the softmax function. The outputs of DNN are the estimates of state posterior probabilities $p(s|x)$, which is converted into a scaled state emission probability as

$$p(x|s) = \frac{p(s|x)}{p(s)} p(x) \quad (2)$$

where $s \in \{1, 2, \ldots, S\}$ is the state id of HMM, S is the total number of states, $p(s)$ is the prior probability of state s, and $p(x)$ is independent of the model. These emission probabilities are used in the Viterbi searching process to find the optimum path in the searching space [?], [14].

From the above description, the mathematic operations of the DNN-based state posterior probabilities evaluation

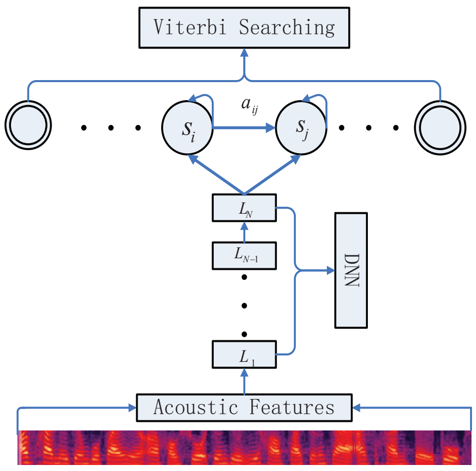

Figure 1. The framework of DNN-HMM based ASR system

are lots of matrix-vector production and nonlinear transformation. For the optimization of the DNN processing speed, one way is to reduce the DNN parameters so as to reduce the mathematic operations, the other way is to speed up the mathematic computation.

III. Acceleration of the DNN-based posterior probabilities evaluation

As described in section II, the mathematic operations of the DNN-based state posterior probabilities evaluation are lots of matrix-vector production and nonlinear transformation. To accelerate the DNN processing speed, one way is to reduce the parameter number of DNN so as to reduce the number of the mathematic operations. Another way is to speed up the mathematic computation itself.

A. Singular Value Decomposition Based Restructuring

The number of parameters of DNN in state-of-the-art ASR systems is often above 40M, which is 2 to 10 times more than the traditional GMM counterparts. Although experiments demonstrate that the recognition accuracy of DNN typically increases with the number of hidden units and layers [12], inspection of the well trained DNN has shown that a large portion of all connections have very small weights. As an specific example, we draw the accumulative distribution of our experimental DNN's weight magnitudes in Figure 2. As we can see, except the first layer, the magnitude of 70% of the weights is below 0.5 on each layer. And for the last two layers, the proportion of the magnitude of weights below 0.2 is over 90%. Motivated by this, [15] try to reduce the model size by removing those connections with small weight magnitude. However, since the location of the remained weights are not continuous, so some extra memories are needed to record their positions. More importantly, under this strategy, we can't take the advantage of using SSE instructions. Yu et al. [16] decompose the original weight matrix into two matrices with smaller size, and the total parameters can be significantly reduced. Similar

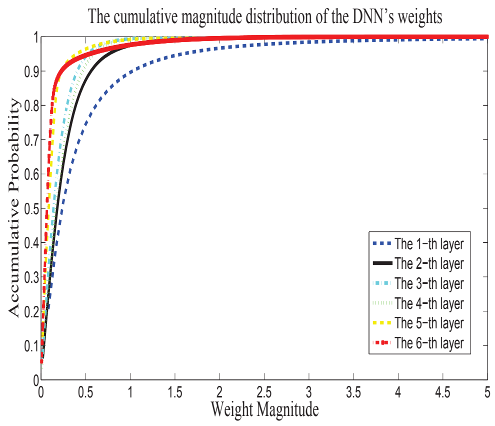

Figure 2. The accumulative distribution of the magnitude of weights on different layers

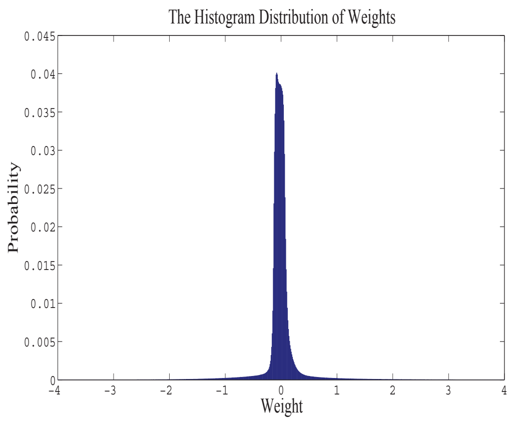

Figure 3. The Histogram Distribution of Weights

to [16], we use Singular Value Decomposition (SVD) to approximate the weight matrices.

For any $m \times n$ matrix A, it can be decomposed as

$$A_{m\times n} = U_{m\times n}\Sigma_{n\times n}V_{n\times n}^{T} \tag{3}$$

where Σ is a diagonal matrix whose elements are A's singular values in a decreasing order. The columns of U and V are the left-singular vectors and right-singular vectors of A corresponding to singular values Σ, respectively. Typically, especially for a sparse matrix, the singular values decrease heavily. Around 10% of singular values contribute to 80% of total values. This means that the top most singular values and the corresponding vectors characterize the most important content of the matrix. If only the top k biggest singular values of A are kept, then A can be approximated as

$$\begin{aligned} A_{m\times n} &\approx U_{m\times k}\Sigma_{k\times k}V_{n\times k}^{T} = \tilde{U}_{m\times k}\tilde{V}_{n\times k}^{T} \\ \tilde{U}_{m\times k} &= U_{m\times k}\Sigma_{k\times k}{}^{\frac{1}{2}}, \tilde{V}_{n\times k}^{T} = \Sigma_{k\times k}{}^{\frac{1}{2}}V_{n\times k}^{T} \end{aligned} \tag{4}$$

The number of parameters is changed from $m \times n$ to $(m+n) \times k$ after approximation.

In implementation, we choose the value of k so that a majority of the weight matrices's singular values are kept, and restructure the DNN with those approximated matrices.

B. *Using SSE Instructions*

The basic idea of Streaming SIMD Extensions (SSE) instruction sets is to perform multiple operations in parallel on contiguous data [17]. With a 128 bits register, the CPU can manipulate 16 bytes worth of data with a single instruction. The basic data types and storage format in the register are illustrated in TABLE I. Fundamental operations on these datatypes use assembly instructions. Fortunately, for C/C++ programmers, the intrinsics [18] provides simple wrapper functions for using the SSE instructions. Take the additional arithmetic of floating-point for example, we can fulfill 4 additions in a single instruction by simply calling the function $\_mm\_add\_ps()$.

For using SSE instructions efficiently, there are two points should be paid special attention to. Firstly, the SSE instructions operate faster when the address of the first byte of data in memory is a multiple of 16, so the memory should be allocated by forcing 16-byte alignment; Secondly, since the SSE instructions operate on a block of 16 bytes, the edge effects should be dealt carefully if the number of elements in a data vector is not a multiple of 16.

C. *Fixed-pointed Arithmetic*

To illustrate the rationality of applying fixed-point arithmetic to DNN, we plot the histogram distribution of weights and biases in Figure 3 and Figure 4 respectively. As we can see, the dynamic range of weights is very small, most of the weights lie in the range of [-2,2], and the dynamic range is [-8,2] for the biases. Besides, the activations of each layer are probabilities in the [0,1] interval. Based on this observation, we can use fixed-point arithmetic to replace the floating-point operation. Specifically, we apply a linear quantization to all of them, the weights are quantized to 8-bit char, the activations are converted into 8-bit unsigned char and the biases are encoded as 32-bit int. Then with the SSE instructions applied to these fixed-pointed data, we can operate 16 elements within a single function call. As can be seen from our experimental results, the quantization error is minimal and can be ignored.

IV. Acceleration of the Decoding Process

Though the time of the DNN-based posterior probability evaluation can be significantly reduced with the techniques describe above as shown by our experiments, those optimizations are independent of the LVCSR task. In this section, we will accelerate the decoding process by taking advantage of the inherent acoustic characteristics of the speech signals.

As it is well known, the speech signal can be treated as piece-wise stationary, and there is a tight correlation between adjacent acoustic frames. Considering this, a

TABLE I.
THE DATA TYPE AND STORAGE FORMAT USED IN SSE INSTRUCTIONS

Data Type	Storage Format							
__128i	$\cdots 16chars \cdots$							
__128i	short	short	short	short	short	short	short	short
__128i	int		int		int		int	
__128	float		float		float		float	
__128d	double				double			

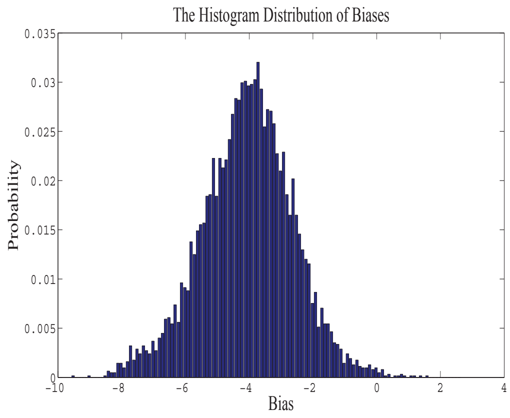

Figure 4. The Histogram Distribution of Biases

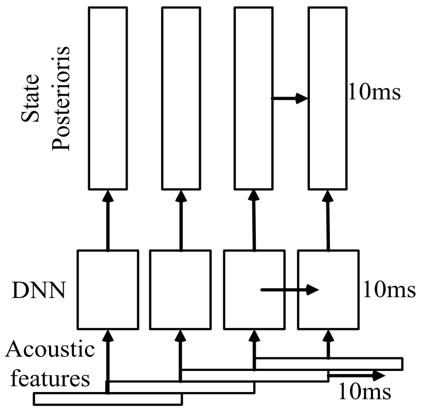

Figure 5. frame synchronous predition [20]

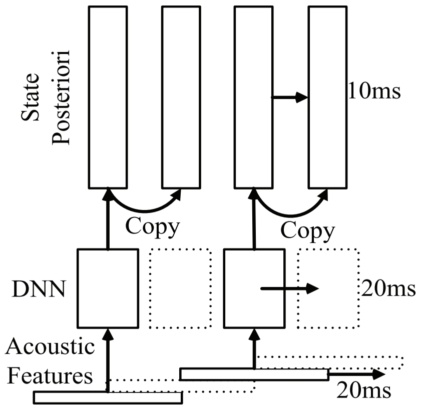

Figure 6. frame asynchronous predition [20]

variable frame rate approach has been successfully applied to GMM-HMM systems in [19]. As for the case of DNN-HMM, a much wider window of time is used as inputs to make a frame classification decision, and the correlation of the adjacent acoustic frames is even bigger than that in the GMM-HMM framework. V. and H. [20] proposed a stronger multi-frame acoustic model for the prediction of states posterior probabilities. Here we just use a simplified version of this approach. Traditionally, the stacked frames are passed to the DNN to generate the predictions for the HMM states synchronously with the Viterbi decoder as shown in Figure 5. Under the multi-frame rate strategy as shown in Figure 6, the predictions are evaluated every 20ms, but the decoder still runs at 10ms rate. The posterior probabilities of those frames without predictions are just copied from previous one. Specifically, the predictions for frame $t+1$ can be just copied from frame t. This means that we can predict the posterior probabilities every $k+1$ frames while skipping k frames without propagating the acoustic features in the DNN. In this sense, we call this as a frame-skipping method.

V. EXPERIMENTS

This section presents the experimental results of the acceleration techniques for our DNN-based ASR systems on a call center speech recognition task.

A. Experimental Setup

The DNN is trained on a data set of 800 hour conversation telephone speech, which contains an input layer, five hidden layers and an output layer. The input layer has 616 inputs consisting of 11 consecutive, overlapping 25 ms frames of speech, sampled every 10 ms. Each frame consists of 13-dimension Perceptual Linear Predictive (PLP) and 1-dimension pitch, appended with Δ, $\Delta\Delta$ and $\Delta\Delta\Delta$ coefficients to form a 56 dimension vector. Each intermediate layer consists of 2048 nodes and uses a sigmoid as non-linearity. The final layer has 6245 outputs corresponding to the context dependent acoustic states (Senones) of HMMs, and uses a softmax function to generate the posterior probabilities for these states.

The test set consists of 96 utterances of telephone dialog, has 14505 Chinese characters and lasts for 2885 seconds. The dictionary contains about 50K words and the recognition is performed using a Weighted Finite-State Transducer (WFST) decoder [21] [22]. All of our

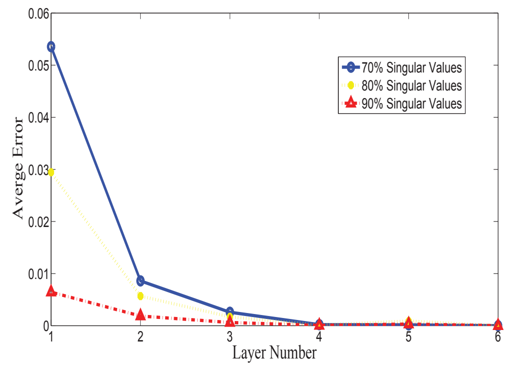

Figure 7. the average error of the output of each layer on the test set with different singular value proportion

experiments are performed on a single CPU on an Intel Xeon Dual Core E5-2620 machine with Windows Server 2008 OS. The implementation is coded in C++ and compiled with Visual studio 2008. The real-time (RT) factor and character error rate (CER) are used as the evaluation metrics for our ASR system.

B. Experimental results

1) Acceleration of calculation on DNN: We first e-valuate the acceleration techniques for the DNN-based posterior probability calculation. The real-time factor of the baseline is 6.1 and is shown in the second row in TABLE III.

- SVD-Based Restructuring. The key factor for the SVD-based model restructuring is to determine how many singular values are kept to approximate the weight matrix. We simply keep k singular values which account of a fixed proportion of the total singular values of the weight matrix in each layer. We plot the average error of the outputs of each layer on the test set with different singular value proportion in Figure 7 (The error means the difference of the outputs on each layer between the original DNN and the restructured DNN). As we can see, the average error is very low even only keep 70% of the singular values, especially at the output layer. The number of parameters of the restructured DNN is 15.3M (22.7M, 32M) when the proportion of the kept singular values is 70% (80%, 90%). To get an optimum choice of k, we evaluate the recognition performance and the corresponding real-time factor of posterior probabilities calculation and list the results in TABLE II. From which we can see, when only keeping 70% of the singular values we can get the best overall performance, the number of parameters of the restructured DNN is halved as compared to the baseline DNN, and the RT factor of posterior computation is reduced from 6.1 to 3.64 with the CER slightly degraded only by 0.7%.
- Using SSE Instructions. With the SSE instructions applied to the floating-point operations of the re-structured DNN based on SVD, the RT factor of the posterior probabilities evaluation is reduced from 3.64 to 1.18 as shown by the third and the 4-th rows in TABLE III. As discussed in section III-B, with the SSE instructions we can implement four floating arithmetic with one instruction. Experimental results demonstrate that using the SSE instruction can real-ize approximate linear speeding up.
- Fixed-pointed Arithmetic. We perform a linear quan-tization to the parameters and the activations of the restructured DNN as described in section III-C. With SSE3 and SSE4 instructions we can perform parallel multiply and add operations on these 8-bit signed/unsigned char weights/activations, this results in a 2.19 incremental speedup of the computation over the floating-point arithmetic as shown at the last line in TABLE III. It is noteworthy that the error caused by quantization is negligible. In fact, the recognition CER of the quantized DNN is 29.5%, as opposed to 29.4% for the floating-point one, the degradation is only 0.1%. But the RT factor is reduced significantly from 1.18 to 0.37.

TABLE II.

	baseline	70% SVs	80% SVs	90% SVs
CER	28.7%	29.4%	29.0%	28.8%
RT	6.1	3.64	4.71	6.23

TABLE III.
EXPERIMENTAL RESULTS OF DNN-BASED POSTERIOR PROBABILITY EVALUATION

	Real Time	Incremental speed-up
Baseline	6.1	–
SVD	3.64	67.6%
SVD+SSE	1.18	208%
SVD+SSE+FP	0.37	219%

For comparison, we summarize the experimental results of the speed of the DNN-based posterior probabilities evaluation in TABLE III. From which we can see, the RT factor of the posterior probabilities calculation module is reduced significantly, from 6.1 to 0.37. And this result demonstrates that the accelerating techniques are very effective.

2) Acceleration of the decoding process: In this section, we present the experiments of the frame-skipping approach. As for our DNN-based ASR sytem, the RT factor is 6.3 without using these techniques as mentioned in the above section. For efficiency reason, all of the results in this section are based on the techniques tested in the above section, which means that the RT factor of the posterior probabilities evaluation module is 0.37. The RT factor is 0.54 and the CER is 29.5% of the non-skipping ASR system as shown in TABLE IV. As discussed above, the time accounted for the posterior probabilities evaluation attributes a large proportion of the entire decoding time, then the speed-up by frame-skipping is obvious. The key factor is to balance the relationship between decoding speed and recognition performance. Therefore, we try different skipping strategies: skipping

TABLE IV.
THE DATA TYPE AND STORAGE FORMAT USED IN SSE INSTRUCTIONS

	no skip	$k = 1$	$k = 2$	$k = 3$
CER	29.5%	29.8%	30.4%	32.1%
RT	0.54	0.35	0.28	0.26

$k = 1, 2, 3$ frames. The results are shown also in TABLE IV. As we can see, when skipping only 1 frame, the CER increased to 29.8% from 29.5% with a slightly performance degradation, but the RT factor is reduced significantly, from 0.54 to 0.35, a 35.2% relative speed-up. When we continue to increase the skipping interval, the recognition performance degrades relatively fast, as shown by the 4-th and 5-th columns in TABLE IV. So in our final system, we only skip one frame when evaluating the posterior probabilities for decoding.

These experimental results show that the acoustic characteristics of the speech signal in the time domain can be utilized to reduce the posterior probabilities calculations during the decoding process. By skipping an interval of frame, we can effectively accelerate the decoding speed yet with negligible recognition performance loss.

VI. CONCLUSIONS

This paper focused on a series of techniques for the speed-up of the deep neural network based speech recognition system. We separate the decoding process into two individual modules and accelerate them respectively. For the posterior probabilities evaluation module, we restructure the original DNN via applying a SVD to the weights matrices in each layer, and then quantize the parameters of the restructured DNN and its activations to convert the floating-point arithmetic into the fixed-pointed arithmetic. Finally, the SSE instructions are used to accelerate these fixed-point arithmetic. With these methods, the RT factor for the posterior probability evaluation module is reduced from 6.1 to 0.37. For the search module (decoding module), we take advantage of the time-domain co-relationship of the adjacent acoustic features, and use a frame-skipping strategy to speed the decoding process further. After all of these optimizations, the total RT factor of the DNN-based ASR system is 0.35 as opposed to 6.3 of the baseline system.

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

REFERENCES

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

[2] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

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

[4] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 24–29.

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

[6] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequencediscriminative training of deep neural networks," in *Proc. INTERSPEECH*, 2013, pp. 2345–2349.

[7] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition." ICASSP, 2013.

[8] K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. W. Senior, P. A. Tucker, *et al.*, "Large scale distributed deep networks." in *NIPS*, 2012, pp. 1232–1240.

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

[11] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, *et al.*, "Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 451–460.

[12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[13] A.-r. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4273–4276.

[14] R. Kiros, "Training neural networks with stochastic hessian-free optimization," *arXiv preprint arXiv:1301.3641*, 2013.

[15] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.

[16] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," *Proc. Interspeech, Lyon, France*, 2013.

[17] "Streaming simd extensions," in $http : //en.wikipedia.org/wiki/Streaming_SIMD_Extensions$.

[18] "Compiler intrinsics," in $http : //msdn.microsoft.com/zh - cn/site/26td21ds$.

[19] Q. Zhu and A. Alwan, "On the use of variable frame rate analysis in speech recognition," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1783–1786.

[20] M. D. Vincent Vanhoucke and G. Heigold, "Multiframe deep neural networks for acoustiv modeling." ICASSP, 2013.

[21] Y. GUO, T. Li, X. ZHAO, J. PAN, and Y. YAN, "Fast speech recognition system using weighted finite-state transducers," *Journal of Information Computational Science*, vol. 9, no. 18, pp. 5807–5814, 2012.

[22] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

Yeming Xiao received his B.E. degree in school of Electronic and Information Engineering from Beihang University in 2008. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interests include Large Vocabulary Continuous Speech Recognition, Neural Networks, Deep Learning and Machine Learning.

Yujing Si received his B.E. degree in Information Engineering from Jilin University in 2009. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interests include Large Vocabulary Continuous Speech Recognition, Neural Networks, Deep Learning and Machine Learning.

Ji Xu received his Bachelor and Masters degrees in 2008 and 2011 respectively, from the Department of Electronic Engineering, Tsinghua University. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interests include Large Vocabulary Continuous Speech Recognition, Deep Learning and Speech synthesis.

Jielin Pan received his B.S. from Peking University in 1986 and his Master degree in Electronic Engineering from Tsinghua University in 1989. From Jan. 2000 to July 2001 he worked at Intel China Research Center as a senior researcher and speech recognition group manager. In Dec. 2002 he joined the Key Laboratory of Speech Acoustics and Content Understanding as a professor. His current research includes: LVCSR, speech analysis, acoustic model and search algorithm.

Yonghong Yan Yonghong Yan received his B.E. from Tsinghua University in 1990, and his PhD from Oregon Graduate Institute (OGI). He worked in OGI as Assistant Professor (1995), Associate Professor (1998) and Associate Director (1997) of Center for Spoken Language Understanding. He worked at Intel from 1998-2001, chaired Human Computer Interface Research Council, worked as Principal Engineer of Microprocessor Research Lab and Director of Intel China Research Center. Currently he is a professor and director of the Key Laboratory of Speech Acoustics and Content Understanding. His research interests include speech processing and recognition, language/speaker recognition, and human computer interface. He has published more than 300 papers and holds 40 patents.