INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Hybrid Accelerated Optimization for Speech Recognition

Jen-Tzung Chien[1], *Pei-Wen Huang*[1], *Tan Lee*[2]

[1]Department of Electrical and Computer Engineering, National Chiao Tung University, Taiwan
[2]Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong

Abstract

Optimization procedure is crucial to achieve desirable performance for speech recognition based on deep neural networks (DNNs). Conventionally, DNNs are trained by using mini-batch stochastic gradient descent (SGD) which is stable but prone to be trapped into local optimum. A recent work based on Nesterov's accelerated gradient descent (NAG) algorithm is developed by merging the current momentum information into correction of SGD updating. NAG less likely jumps into local minimum so that convergence rate is improved. In general, optimization based on SGD is more stable while that based on NAG is faster and more accurate. This study aims to boost the performance of speech recognition by combining complimentary SGD and NAG. A new hybrid optimization is proposed by integrating the SGD with momentum and the NAG by using an interpolation scheme which is continuously run in each mini-batch according to the change rate of cost function in consecutive two learning epochs. Tradeoff between two algorithms can be balanced for mini-batch optimization. Experiments on speech recognition using CUSENT and Aurora-4 show the effectiveness of the hybrid accelerated optimization in DNN acoustic model.

Index Terms: hybrid optimization, stochastic gradient descent, deep neural network, speech recognition

1. Introduction

Deep learning has become a popular research topic for speech recognition in recent years. Many researchers have demonstrated that deep neural network (DNN) based speech recognition can significantly reduce word error rate over the traditional system using Gaussian mixture model based hidden Markov model (GMM-HMM) [1, 2]. However, training such a complex DNN model with tens of thousands of parameters is a big challenge to build a desirable speech recognition system. Challenge is twofold. First, the computation cost is demanding because iterative training procedure is run for individual parameters. Second, the convergence of training procedure is not always guaranteed due to the non-convex DNN model. To deal with these issues, the parallel training algorithm [3] was developed to speed up training procedure for abundant model parameters. In addition, the convergence of training procedure could be improved by introducing acceleration scheme in optimization algorithm. This paper presents a hybrid accelerated optimization procedure for DNN speech recognition.

In the literature, stochastic gradient descent (SGD) [4, 5] has been known as a very popular optimization algorithm for DNN speech recognition where the cross-entropy error function is minimized from mini-batches of speech data. This method requires an adaptive learning rate to assure convergence in learning process. The drawback of SGD is that the training procedure is prone to be slow due to the flatness in the gradients of model parameters [6, 7]. Therefore, the momentum method was implemented to catch the velocity information through accelerating the descent when the gradients in successive two learning epochs keep the same direction and slowing down the updating rule when the direction of gradients is changed [8, 9]. More recently, the second-order optimization methods were proposed to improve the convergence of learning [10, 11, 12]. In [13, 14], the Hessian-free optimization was developed by simulating and calculating Hessian matrix indirectly. The second-order information as the curvature near the parameters was used to speed up the updating rule. However, this method is demanding on the memory and computation costs.

This paper focuses on the first-order optimization and proposes a training algorithm which captures the complimentary capabilities of SGD and Nesterov's accelerated gradient descent (NAG) [15] algorithms. Basically, SGD is capable of conducting the stable learning while NAG is seen as a solution to speed up learning procedure. We integrate these two algorithms into a unified training framework. This hybrid algorithm continuously interpolates both optimization methods based on the convergence rate of error function from consecutive two learning epochs. Accordingly, an integrated optimization is developed to effectively balance the tradeoff between SGD with momentum and NAG. We therefore accelerate the parameter updating and reach a stable optimum in learning schedule. The experiments on speech recognition are conducted to illustrate the merit of this hybrid learning algorithm for deep acoustic modeling. This paper is organized as follows. Section 2 surveys and evaluates the optimization algorithms for DNN and simulation function. Section 3 proposes the hybrid accelerated optimization which balances the tradoff between two methods. Experiments on CUSENT and Aurora-4 tasks are reported in Section 4. Conclusions from this study are drawn in Section 5.

2. Background survey

2.1. Optimization in deep neural network

The traditional speech recognition system was built by using acoustic model based on GMM-HMMs. In recent years, DNNs have been popularly developed. Basically, DNN can replace GMM for calculating the posterior probabilities of individual HMM states given a phonetic unit. Such an acoustic model is known as the DNN-HMM. DNN optimization is a non-convex problem. No closed-form solution exists. There are two passes in DNN iterative training procedure based on the error back-propagation algorithm. In forward pass, the input speech observations are propagated layer by layer towards softmax outputs for class posteriors. An affine transformation and a non-linear activation are computed for individual units in each layer. In backward pass, we calculate the gradients of cross entropy error function with respect to DNN parameters and update the model parameters according to the stochastic gradient descent (SGD)

http://dx.doi.org/10.21437/Interspeech.2016-192

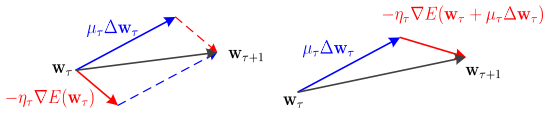


Figure 1: *Geometrical interpretation for SGD with momentum (left) and NAG (right).*

algorithm by using mini-batch data

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau - \eta_\tau \nabla E(\mathbf{w}_\tau) \tag{1}$$

where $\mathbf{w}_\tau$ and η_τ denote the parameter vector and learning rate using a mini-batch indexed by τ, respectively, and ∇E denotes the gradient with respect to $\mathbf{w}_\tau$. To obtain $\nabla E(\mathbf{w}_\tau)$, local gradients of individual weights are calculated from mini-batch data and then propagated from output layer back to input layer.

2.2. Accelerated optimizations

In general, SGD learning is stable but suffers from the circumstance that the learning procedure gets slower in later training stage. To deal with this issue, it is popular to incorporate the momentum mechanism into SGD learning. Learning speed is increased by taking into account the learning velocity. Basically, this mechanism is performed by increasing the step size when the gradient keeps the same learning direction but decreasing the step size when the gradient varies the learning direction. Therefore, SGD with momentum can correct the gradient by using the velocity information of model parameters. The updating rule of SGD with momentum is run by

$$\begin{aligned} \mathbf{w}_{\tau+1} &= \mathbf{w}_\tau + \Delta \mathbf{w}_{\tau+1} \\ \text{where} \quad \Delta \mathbf{w}_{\tau+1} &= \mu_\tau \Delta \mathbf{w}_\tau - \eta_\tau \nabla E(\mathbf{w}_\tau). \end{aligned} \tag{2}$$

In addition to the term $-\eta_\tau \nabla E(\mathbf{w}_\tau)$, the velocity or the adjustment of parameters $\Delta \mathbf{w}_{\tau+1}$ at new iteration using current mini-batch is formed by further incorporating the velocity $\Delta \mathbf{w}_\tau$ at old iteration using previous mini-batch. μ_τ is merged as the momentum parameter. The learning speed is increased when consecutive two iterations hold the same learning direction.

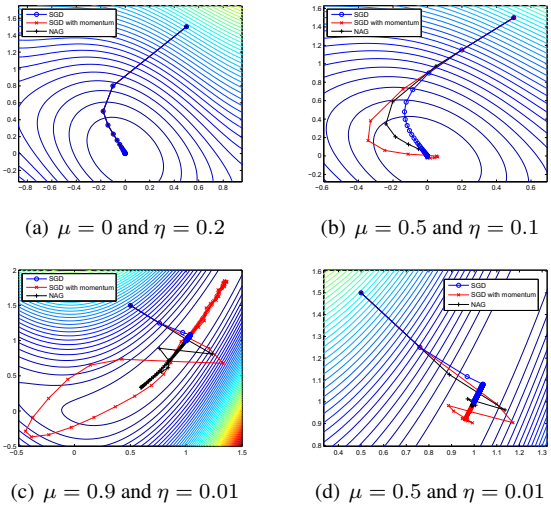
In [15], Nesterov's accelerated gradient descent (NAG) was proposed as an alternative to accelerate the SGD with momentum. The idea of NAG was designed by calculating the gradient by using the information from model parameters as well as momentum term. Therefore, NAG may accelerate the updating rule and early correct the updating which is not so reliable. Accordingly, NAG more likely skips local minimum by calculating the velocity at new iteration or mini-batch by

$$\Delta \mathbf{w}_{\tau+1} = \mu_\tau \Delta \mathbf{w}_\tau - \eta_\tau \nabla E(\mathbf{w}_\tau + \mu_\tau \Delta \mathbf{w}_\tau) \tag{3}$$

where $\mathbf{w}_\tau$ is now replaced by $\mathbf{w}_\tau + \mu_\tau \Delta \mathbf{w}_\tau$ when calculating the gradient term ∇E.

2.3. Comparison for optimization methods

The key difference between SGD with momentum and NAG is the calculation of gradient term. NAG merges the current momentum $\mu_\tau \Delta \mathbf{w}_\tau$ into calculation of gradient for parameter updating at new iteration $\tau + 1$. Different from SGD with momentum, NAG conducts the scheme of looking ahead so that the resulting gradient may speed up learning and prevent the learning procedure trapping into local minimum. Figure 1 illustrates that if the momentum term and the gradient term point



(a) $\mu = 0$ and $\eta = 0.2$ (b) $\mu = 0.5$ and $\eta = 0.1$

(c) $\mu = 0.9$ and $\eta = 0.01$ (d) $\mu = 0.5$ and $\eta = 0.01$

Figure 2: *Learning curves for simulation functions.*

to the same direction, the updating will be much larger than the updating which considers the gradient term but *without* momentum term. NAG shows that the weight jumps a step size $\mu_\tau \Delta \mathbf{w}_\tau$ for calculation of gradient at $\mathbf{w}_\tau + \mu_\tau \Delta \mathbf{w}_\tau$ so that NAG can correct the updating faster than SGD with momentum.

Table 1: *Number of iterations for finding optimum of simulation functions in Figure 2(a), 2(b), 2(c) and 2(d).*

Simulated Function	SGD	SGD (m)	NAG
three-hump camel (a)	72	72	72
three-hump camel (b)	161	81	63
Rosenbrock (c)	4627	2293	2115
Rosenbrock (d)	4627	386	340

We conduct a case study of optimization learning for finding the optimum of a simulation function by using SGD, SGD with momentum (denoted by SGD (m)) and NAG. The simulation function is a three-hump camel function $f(w_x, w_y) = 2w_x^2 + 1.05w_x^4 + \frac{w_x^6}{6} + w_x w_y + w_y^2$ in Figures 2(a) and 2(b) where the optimum is located at $(0, 0)$ but with different momentum parameter μ and learning rate η. The other simulation function is the Rosenbrock function $f(w_x, w_y) = (1 - w_x)^2 + 10(w_y - w_x^2)^2$ in Figures 2(c) and 2(d) where the optimum is located at $(1, 1)$ with different μ and η. We set the same initial position at $(0.5, 1.5)$. Two-dimensional parameters w_x and w_y are updated iteratively using the optimization algorithm until the parameters reach to the optimum. Figure 2(a) shows that the learning curves of different methods are the same because μ is 0. In Figures 2(b), 2(c) and 2(d), SGD uses many more iterations than the other methods while NAG corrects the updating direction faster than the others. The number of iterations for arriving at optimum is compared in Table 1. NAG is faster and SGD is slower in terms of iteration number for convergence.

3. Hybrid optimization

This paper presents a hybrid accelerated optimization for DNN parameters and applies it for speech recognition. Our idea is to boost speech recognition performance by combining the complimentary SGD and NAG optimizations. In general, learning procedure using SGD is more stable while NAG usually runs

faster in convergence and less likely traps into local minimum. We would like to capture the stability and acceleration in optimization by combining SGD with momentum and NAG. This hybrid optimization is implemented by two realizations. One is hard combination and the other is soft combination.

3.1. Hard hybrid optimization

Hybrid optimization using hard combination is realized through a switching scheme between SGD with momentum and NAG. Either SGD with momentum or NAG is active in each learning epoch. In this implementation, we use the approximate solution to NAG algorithm. Starting from Eq. (3), NAG algorithm is realized by $\Delta \mathbf{w}_{\tau+1} = \mu_\tau \Delta \mathbf{w}_\tau - \eta_\tau \nabla E(\hat{\mathbf{w}}_\tau)$ where the gradient ∇E is calculated by using the predictor

$$\hat{\mathbf{w}}_\tau = \mathbf{w}_\tau + \mu_\tau \Delta \mathbf{w}_\tau \tag{4}$$

which can be manipulated to construct the updating rule [16]

$$\begin{aligned} \hat{\mathbf{w}}_{\tau+1} &= \hat{\mathbf{w}}_\tau - \mu_\tau \Delta \mathbf{w}_\tau + \Delta \mathbf{w}_{\tau+1} + \mu_{\tau+1} \Delta \mathbf{w}_{\tau+1} \\ &= \hat{\mathbf{w}}_\tau + \mu_\tau \mu_{\tau+1} \Delta \mathbf{w}_\tau - \eta_\tau (1 + \mu_{\tau+1}) \nabla E(\hat{\mathbf{w}}_\tau). \end{aligned} \tag{5}$$

This updating is iteratively performed to find the parameters $\hat{\mathbf{w}}_{\tau+1}$ which will approximate and converge to $\mathbf{w}_{\tau+1}$. Interestingly, the fashion of updating is similar to that in Eqs. (2) and (3). However, this NAG algorithm adopts two iterations of momentum parameters μ_τ and $\mu_{\tau+1}$ in the updating rule.

In general, NAG is unstable in early training stage. When implementing the hard hybrid optimization, we first apply SGD with momentum in Eq. (2) for parameter updating. After updating by using T mini-batches in a learning epoch, we examine the convergence rate of cross-entropy error functions by using $\mathbf{w}_T$ in current epoch and the best parameters $\mathbf{w}_b$ in previous epoch

$$\gamma = \frac{E(\mathbf{w}_b) - E(\mathbf{w}_T)}{E(\mathbf{w}_b)}, \qquad 0 \le \gamma \le 1. \tag{6}$$

To assure model generalization, we use a small set of *validation data* to calculate this measurement. If this value is smaller than a threshold θ, the optimization is switched to NAG algorithm in Eq. (5) so as to accelerate the updating procedure in next learning epoch. Otherwise, this system will keep running SGD with momentum in next epoch. The best parameters are subsequently refreshed by $\mathbf{w}_b \leftarrow \mathbf{w}_T$ after each learning epoch. By using this hybrid optimization, we can automatically adapt the optimization procedure in different epochs in accordance with this convergence rate.

3.2. Soft hybrid optimization

A weakness of hard hybrid optimization is the determination of threshold θ for judging either SGD or NAG which is generally sensitive in system performance and should be empirically determined in different tasks. To tackle this problem, we propose a *soft* hybrid optimization to balance the tradeoff between the stability using SGD and the acceleration using NAG without a switching threshold θ. Our idea is to conduct a tight fusion of SGD with momentum and NAG in each min-batch and learning epoch based on a *single* updating rule

$$\begin{aligned} \mathbf{w}_{\tau+1} = \mathbf{w}_\tau &+ \mu_\tau \left[\gamma + (1-\gamma)\mu_{\tau+1}\right] \Delta \mathbf{w}_\tau \\ &- \eta_\tau \left[\gamma + (1-\gamma)(1+\mu_{\tau+1})\right] \nabla E(\mathbf{w}_\tau) \end{aligned} \tag{7}$$

where the second term and the third term in right-hand-side are obtained by linearly interpolating the corresponding terms in Eqs. (2) and (5) through an interpolation parameter γ. In Eq. (7), the velocity $\Delta \mathbf{w}_\tau$ is updated by following the style of momentum method in Eq. (2). Basically, parameter γ balances the tradeoff between SGD and NAG which is measured by the convergence rate of error function as given in Eq. (6). Similar to hard hybrid optimization, this parameter is determined in each learning epoch by using validation data. The updating rule adopts the same γ for all mini-batches in next learning epoch. In an extreme case, if parameter $\gamma = 1$ is calculated, we equivalently perform SGD with momentum to slow down learning step. If $\gamma = 0$ happens, this optimization is reduced to NAG optimization to speed up learning. However, a real-valued parameter γ paves a solution to soft hybrid optimization which simultaneously and consistently captures the stability and acceleration at each mini-batch updating for DNN optimization. The mini-batch optimization algorithm using soft combination is constructed in Algorithm 1.

Algorithm 1 *Soft hybrid optimization using T mini-batches*

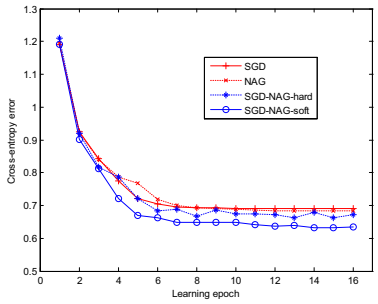
Given $\mathbf{w}_0 \in \mathbb{R}^N$, $\eta_0 > 0$ and $\mu_0 > 0$
$\mathbf{w}_b = \mathbf{w}_0$
for epoch $= 1, 2, 3, \cdots$ **do**
  for $\tau = 0, 1, 2, \cdots, T$ **do**
    Compute $\nabla E(\mathbf{w}_\tau)$
    $\mathbf{w}_{\tau+1} \leftarrow \mathbf{w}_\tau + \mu_\tau \left[\gamma + (1-\gamma)\mu_{\tau+1}\right] \Delta \mathbf{w}_\tau - \eta_\tau \left[\gamma + (1-\gamma)(1+\mu_{\tau+1})\right] \nabla E(\mathbf{w}_\tau)$
    $\Delta \mathbf{w}_{\tau+1} \leftarrow \mu_\tau \Delta \mathbf{w}_\tau - \eta_\tau \nabla E(\mathbf{w}_\tau)$
  end for
  Compute $\gamma \leftarrow \frac{E(\mathbf{w}_b) - E(\mathbf{w}_T)}{E(\mathbf{w}_b)}$
  $\mathbf{w}_b \leftarrow \mathbf{w}_T$
end for

4. Experiments

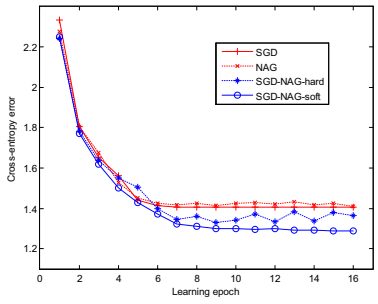
We compare different optimization methods for DNN speech recognition in terms of learning curve and word error rate (WER) by using CUSENT and Aurora-4 tasks. SGD with momentum (or simply denoted by SGD), NAG, SGD-NAG-hard (hard hybrid optimization) and SGD-NAG-soft (soft hybrid optimization) are implemented. CUSENT is a read speech corpus consisting of continuous Cantonese utterances collected in a quiet room [17]. There are 20,400 training utterances from 34 males and 34 females and 1,200 test utterances from the other 6 males and 6 females. Aurora-4 is a noisy English speech database. Training set contains 7,137 utterances from 83 speakers totally 14 hours. The evaluation set has 4,620 utterances from 8 speakers. There are 14 evaluation sets which are grouped into subsets A (clean data), B (noisy data), C (clean data with channel distortion), and D (noisy data with channel distortion). A small set of training data is held out for validation in CUSENT and Aurora-4 tasks.

4.1. Experimental setup

In the experiments, Kaldi toolkit [18] was used for DNN speech recognition. The input feature vector consisted of 40 FMLLR features in CUSENT task and 40 FBNK features in Aurora-4 task. Each frame contained features from current frame and ± 5 contextual frames. The input dimension for DNN was 440 (40×11). Bi-phone models in CUSENT and triphone models in Aurora-4 were constructed by using decision tree state tying. In CUSENT, DNN architecture was constructed by 6 hidden layers where each layer had 1024 hidden neurons. In Aurora-4, there were 7 hidden layers with 2048 neurons in each layer



(a)



(b)

Figure 3: *Learning curves by using different optimization methods under (a) CUSENT task and (b) Aurora-4 task.*

[19, 20, 21]. Language model (LM) for decoding was based on bigrams for CUSENT and trigrams for Aurora-4.

In the implementation, GMM-HMMs were trained to find the state alignment of training data for initialization of a DNN-HMM system. During DNN training, the weight parameters were pre-trained by using the restricted Boltzmann machine. The training features were then propagated through the network. After the forward computation, the error back-propagation was run to calculate the gradients for updating individual weights. The initial learning rate was $\eta_0 = 0.008$, the momentum value was $\mu_0 = 0.3$, and the threshold was $\theta = 0.01$ in CUSENT task. In Aurora-4, we used $\eta_0 = 0.008$, $\mu_0 = 0.3$ and $\theta = 0.005$. These parameters were empirically selected from validation data. Based on the convergence rate of validation data, the learning rate was exponentially decayed. The momentum value was fixed in training procedure. The mini-batch size of 256 frames was adopted. After all mini-batches run in a learning epoch, we compute the convergence rate γ using validation data and conduct the updating for the next epoch. DNN acoustic model is iteratively trained through learning epochs until convergence. During decoding phase, LM is applied to decode the most likely word sequence.

4.2. Experimental results

First of all, we illustrate the learning curves of training data and WERs of test data by using GMM-HMMs and DNNs where four optimization algorithms are compared under CUSENT task. Figure 3(a) displays the cross-entropy error function versus the learning epoch. Basically, four optimization methods converge in learning procedure although NAG and SGD-NAG-hard oscillate a bit by learning epochs. SGD-NAG-soft runs a smooth learning curve with the lowest error among four methods. Table 2 reports WERs of using different models and different optimization methods. DNN-HMMs perform significantly better than GMM-HMMs. NAG is slightly better than SGD but worse than hard and soft combinations of SGD and NAG. SGD-NAG-soft achieves lower WER than the other methods.

Table 2: *WERs (%) of GMM-HMMs and DNNs by using different optimization methods. CUSENT task is evaluated.*

Model	WER (%)
GMM-HMM	9.77
DNN (SGD)	7.34
DNN (NAG)	7.29
DNN (SGD-NAG-hard)	7.23
DNN (SGD-NAG-soft)	7.09

Table 3: *WERs (%) of GMM-HMMs and DNNs by using different optimization methods. Aurora-4 task is evaluated.*

Model	A	B	C	D	Avg.
GMM-HMM	7.29	12.97	12.61	27.66	18.83
DNN (SGD)	3.47	7.72	10.59	22.12	13.79
DNN (NAG)	3.40	7.74	10.18	22.33	13.88
DNN (SGD-NAG-hard)	3.29	7.60	10.14	22.00	13.65
DNN (SGD-NAG-soft)	3.19	7.43	9.98	20.91	13.09

Figure 3(b) and Table 3 display the learning curves and WERs by using different optimization algorithms under Aurora-4 task, respectively. WERs are averaged over four conditions including one subset of A, six subsets of B, one subset of C and six subsets of D. Basically, the cross entropy error functions using four methods converge in their learning procedure. Among these methods, the lowest training error is achieved by using SGD-NAG-soft. SGD-NAG-hard algorithm is relatively unstable. Hybrid optimization obtains lower training error than individual optimization. In this task, SGD with momentum has lower error than NAG. In terms of WER, DNN outperforms GMM-HMM. Hybrid optimizations SGD-NAG-hard and SGD-NAG-soft perform better than individual methods SGD and NAG. SGD obtains lower WER than NAG. In this comparison, the lowest averaged WER 13.09% is achieved by the proposed SGD-NAG-soft which runs a stable and accelerated learning procedure in presence of multi-condition training data.

5. Conclusions

We have presented a joint optimization algorithm which balanced the tradeoff between the stability of using SGD with momentum and the acceleration of using NAG. The hybrid SGD and NAG could be realized by hard combination as well as soft combination. Hard combination employed a switching scheme in learning procedure and ran either SGD with momentum or NAG based on the convergence rate of new epoch relative to current epoch by using validation data. A threshold was required in a hard decision. Alternatively, we fused SGD with momentum and NAG through an interpolation mechanism where the convergence rate was used as the interpolation weight. It became a natural fusion of SGD with momentum and NAG which relied more on NAG when convergence rate was low and on SGD when convergence rate was high. Such a single updating rule was run in each mini-batch of a learning epoch. The proposed algorithm was evaluated for DNN acoustic modeling based on learning curve and WER. Experimental results show the effectiveness of soft integration of SGD with momentum and NAG for clean speech recognition using CUSENT as well as noisy speech recognition using Aurora-4.

6. References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, pp. 82–97, 2012.

[2] G. Saon and J.-T. Chien, "Large-vocabulary continuous speech recognition systems v a look at some recent advances," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 18–33, 2012.

[3] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of deep neural networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5587–5591.

[4] L. Bottou, "Stochastic gradient learning in neural networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, 1991.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[7] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *IEEE International Conference on Neural Networks*, vol. 1, 1993, pp. 586–591.

[8] N. S. Keskar and G. Saon, "A nonmonotone learning rate strategy for SGD training of deep neural networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4974–4978.

[9] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. of International Conference on Machine Learning (ICML)*, 2013, pp. 1139–1147.

[10] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proc. of International Conference on Machine Learning (ICML)*, 2011, pp. 265–272.

[11] A. Bordes, L. Bottou, and P. Gallinari, "SGD-QN: careful quasi-Newton stochastic gradient descent," *Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[12] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[13] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. of International Conference on Machine Learning (ICML)*, 2010, pp. 735–742.

[14] J. Martens and I. Sutskever, "Learning recurrent neural networks with Hessian-free optimization," in *Proc. of International Conference on Machine Learning (ICML)*, 2011, pp. 1033–1040.

[15] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k2)," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[16] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8624–8628.

[17] T. Lee, W. K. Lo, P. Ching, and H. Meng, "Spoken language resources for Cantonese speech processing," *Speech Communication*, vol. 36, no. 3, pp. 327–342, 2002.

[18] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[19] J.-T. Chien and T.-W. Lu, "Tikhonov regularization for deep neural network acoustic modeling," in *Proc. of IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 147–152.

[20] ——, "Deep recurrent regularization neural network for speech recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4560–4564.

[21] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network for language modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 361–374, 2016.