MEAN-NORMALIZED STOCHASTIC GRADIENT FOR LARGE-SCALE DEEP LEARNING

Simon Wiesler[1], *Alexander Richard*[1], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
[2]LIMSI CNRS, Spoken Language Processing Group, Paris, France

ABSTRACT

Deep neural networks are typically optimized with stochastic gradient descent (SGD). In this work, we propose a novel second-order stochastic optimization algorithm. The algorithm is based on analytic results showing that a non-zero mean of features is harmful for the optimization. We prove convergence of our algorithm in a convex setting. In our experiments we show that our proposed algorithm converges faster than SGD. Further, in contrast to earlier work, our algorithm allows for training models with a factorized structure from scratch. We found this structure to be very useful not only because it accelerates training and decoding, but also because it is a very effective means against overfitting. Combining our proposed optimization algorithm with this model structure, model size can be reduced by a factor of eight and still improvements in recognition error rate are obtained. Additional gains are obtained by improving the Newbob learning rate strategy.

Index Terms— deep learning, optimization, speech recognition, LVCSR

1. INTRODUCTION

Deep neural networks (DNNs) have become an essential part of state-of-the-art automatic speech recognition systems. In particular, hybrid architectures where a DNN models the tied context-dependent states of an HMM speech recognizer have been shown to work very well [1, 2, 3, 4].

DNN training is a very difficult and highly non-convex optimization problem. The most widely used optimization algorithm used for DNN training is stochastic gradient descent (SGD). Typically, the stochastic gradient is computed on mini-batches. SGD scales well to large databases due to its stochastic nature. Further, the computation within one mini-batch can be parallelized well on GPUs. Nevertheless, improving the optimization is of great interest for two reasons: First, even with the use of GPUs, training time for DNNs is very high and may be reduced with improved optimization algorithms. Second, in regions with pathological curvature, SGD converges very slowly or may even fail to improve the training objective further. Therefore, better error rates may be achieved by improving the optimization. Both of these issues are especially important for large-scale learning tasks as speech recognition.

An alternative to SGD is the use of batch algorithms, *i.e.,* algorithms where the gradient is computed on the full dataset, for example LBFGS [5, 6], Rprop [7], or the Hessian-Free algorithm [8, 9]. These algorithms have in common that they make efficient use of second-order information. Further, they can be parallelized well by distributing the gradient computation across a large number of devices. However, on very large datasets, the computational costs of batch algorithms may become prohibitive. Therefore it is interesting to incorporate second-order techniques into stochastic algorithms as well.

Many approaches to stochastic second-order optimization are based on an approximation to the Hessian matrix on the mini-batch, *e.g.* a diagonal approximation [10] or a quasi-Newton approximation [11, 12]. Most of these approaches have not been very successful, because the stochastic second-order information is very noisy and the algorithms have additional computational costs. In this work, we develop *mean-normalized SGD* (MN-SGD), a new stochastic second-order algorithm which uses analytic results [13, 14] about the structure of the objective function which is optimized in training. We prove convergence of our algorithm in a convex setting.

It has often been observed that improved optimization algorithms may cause overfitting on machine learning tasks. In this work, we investigate a recently proposed model structure [15, 16] that allows for reducing model size and thus improves generalization. Furthermore, these smaller models reduce computational costs in both training and decoding.

Our experiments on an English broadcast conversations recognition task show that our proposed algorithm converges much faster than SGD. With our algorithm, model size can be reduced by a factor of eight and still improvements in recognition accuracy are obtained.

2. MEAN-NORMALIZED STOCHASTIC GRADIENT

In this section, we describe our proposed optimization algorithm, prove its convergence, and describe methods to improve generalization performance.

2.1. General setting and notation

Let $X \subset \mathbb{R}^D$ denote the observation space, $\mathcal{S} = \{1, \dots, S\}$ the classes, and $\{(x_1, s_1), \dots, (x_T, s_T)\} \subset X \times \mathcal{S}$ the training sample. Let

$$q_\Lambda : \mathbb{R}^D \to \mathbb{R}^S, \quad x \mapsto q_\Lambda(x) \tag{1}$$

denote the discriminant function that is defined by a neural network with parameters Λ. The aim of neural network training is the minimization of an objective function

$$F : \mathbb{R}^N \to \mathbb{R},\ \Lambda \mapsto \sum_{t=1}^{T} \ell(q_\Lambda(x_t), s_t) + \alpha\, r(\Lambda)\ , \tag{2}$$

where $\ell(q_\Lambda(x_t), s_t)$ is the loss function, N the number of free parameters, and r is an optional regularization term with regularization constant $\alpha \geq 0$. The most widely used training criterion for

This work was partly realized under the Quaero Programme, funded by OSEO, French State agency for innovation. The research leading to these results has received funding from the European Union Seventh Framework Programme EU-Bridge (FP7/2007-2013) under grant agreement N287658. H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Ile-de-France.

neural networks is the *cross-entropy criterion*, *i.e.,* $\ell(q(x), s) = -\log q(x)[s]$. The update rule of SGD is

$$\Lambda_{t+1} = \Lambda_t - \eta_t \nabla F(\Lambda_t, \mathcal{B}_t) \ , \tag{3}$$

where $\eta_t > 0$ is the learning rate and $\nabla F(\Lambda_t, \mathcal{B}_t)$ is the gradient of F on a mini-batch $\mathcal{B}_t$. The stochastic second-order update rule is

$$\Lambda_{t+1} = \Lambda_t - \eta_t B_t \nabla F(\Lambda_t, \mathcal{B}_t) \ , \tag{4}$$

where B_t is chosen to approximate the inverse Hessian of F.

2.2. Derivation of the algorithm

Although computing the exact Hessian matrix is prohibitive in practice, important conclusions can be drawn from a theoretical analysis. A well-known result is that the eigenvalues of the Hessian matrix depend on the mean and the variance of the input features, see [13] for squared error loss and [14] for the cross-entropy loss and weaker assumptions . It can be concluded, that convergence speed is improved when mean and variance of the input features are normalized. For neural networks, only the input to the lowest layer can be normalized directly, because the input to the other layers changes dynamically during training. The idea of our proposed algorithm is to perform a mean normalization step on model side instead of explicitly normalizing the features. Running averages of the activations are used for the required mean statistics.

For terms of simplicity, we consider only the parameters of one layer of the network in the following. Let $W \in \mathbb{R}^{D_1 \times D_2}$ denote the weight matrix and $a \in \mathbb{R}^{D_2}$ the bias vector. The objective function for a single training sample (x, s) can be written as

$$F : \mathbb{R}^{(D_1+1)\times D_2} \mapsto \mathbb{R}, \quad (W, a) \mapsto \mathcal{G}(W^T z + a) \ , \tag{5}$$

where z is the input to the layer and $\mathcal{G}$ is the composition of the loss $\ell(\cdot, s)$ and the remaining higher layers and non-linearities of the network. The objective function of the mean-normalized feature is

$$\widetilde{F} : \mathbb{R}^{(D_1+1)\times D_2} \mapsto \mathbb{R}, \quad (W, a) \mapsto \mathcal{G}(W^T (z + b) + a) \ , \tag{6}$$

for a vector $b \in \mathbb{R}^{D_1}$. It is possible to map between the parameters corresponding to the original and transformed features. With

$$\phi(W, a) := (W, a - W^T b) \ , \tag{7}$$

we have

$$F(W, a) = \widetilde{F}(\phi(W, a)) \quad \text{and} \tag{8}$$

$$\widetilde{F}(W, a) = F(\phi^{-1}(W, a)) \ . \tag{9}$$

Instead of explicitly normalizing the features and optimizing $\widetilde{F}$, the parameters can be mapped to the normalized parameter space with ϕ, updated with the SGD rule (3), and mapped back to the original parameter space with ϕ^{-1}:

$$(\hat{W}, \hat{a}) := \phi^{-1}\Big(\phi(W, a) - \eta \cdot \nabla \widetilde{F}\big(\phi(W, a)\big)\Big). \tag{10}$$

This modified update rule requires the gradient of $\widetilde{F}$, which can be calculated with the chain rule:

$$\nabla_W \widetilde{F}(W, a) = \nabla_W F\big(\phi^{-1}(W, a)\big) + b \cdot \nabla_a^T F\big(\phi^{-1}(W, a)\big), \tag{11}$$

$$\nabla_a \widetilde{F}(W, a) = \nabla_a F\big(\phi^{-1}(W, a)\big). \tag{12}$$

Inserting into (10), finally leads to the update rule of our proposed mean-normalized SGD:

$$\hat{W} = W - \eta \cdot \big(\nabla_W F(W, a) + b \cdot \nabla_a F(W, a)^T\big) \ , \tag{13}$$

$$\hat{a} = a - \eta \cdot \big(\nabla_W F(W, a)^T \cdot b + (1 + b^T b) \nabla_a F(W, a)\big) \ . \tag{14}$$

Note that this update rule can be written in the form of a general second-order update (4). Writing the parameters as $\Lambda = (\mathrm{vec}(W), \mathrm{vec}(a))$, and omitting the indices t, the matrix B in (4) is of the form

$$B = \begin{pmatrix} I & C^T \\ C & (1 + b^T b) I \end{pmatrix}, \tag{15}$$

where

$$C := \begin{pmatrix} b_1 \dots b_{D_1} & & \\ & \ddots & \\ & & b_1 \dots b_{D_1} \end{pmatrix} \in \mathbb{R}^{D_2 \times D_1 \cdot D_2}. \tag{16}$$

The mean activations of neural networks can be calculated by running averages:

$$b_t = \alpha \mathbb{E}(y|\mathcal{B}_t) + (1 - \alpha) b_{t-1}. \tag{17}$$

Here, y is the activation of the layer, $\mathbb{E}(y|\mathcal{B}_t)$ is the activation mean on mini-batch $\mathcal{B}_t$, and $0 < \alpha < 1$ is a smoothing factor.

Analogous formulas can be derived for an implicit variance normalization . However, in initial experiments, implicit variance normalization did not yield improvements. Therefore, we have not considered this approach further.

2.3. Convergence proof

For the MN-SGD convergence proof, we need to restrict ourselves to the strictly convex case, *e.g.* a single layer network trained according to the ℓ_2-regularized cross-entropy criterion. Note that this assumption is required for almost all results on convergence guarantees. Our convergence proof is an application of a theorem by Sunehag et al. [17, Theorem 3.2], which follows from a very general result by Robbins and Siegmund [18] that makes use of supermartingale theory.

Sunehag's theorem states that a stochastic second-order algorithm converges almost surely if the matrix B in (4) is symmetric and its eigenvalues are bounded below and above by positive numbers m and M. Further, commonly used assumptions on the learning rates and mild assumptions on the objective function are required.

The matrix B in (15) is symmetric. Its block-structure allows for the computation of the extremal eigenvalues:

$$\lambda_{\max,\,\min} = 1 + \frac{1}{2}\|b\|^2 \pm \|b\| \sqrt{1 + \frac{1}{4}\|b\|^2} \ . \tag{18}$$

It can be shown that the eigenvalues are bounded when b is bounded, which proves almost sure convergence of MN-SGD.

2.4. Learning rate strategies

A critical aspect for the performance of DNNs is the choice of a learning rate strategy, see *e.g.* [19]. In our group, we mostly used the popular *Newbob* learning rate strategy as it is implemented in the Quicknet software [20], *i.e.,* the learning rate is kept fixed as long as the frame classification error (FCE) on the cross-validation (CV) set improves by at least 0.5%. In all subsequent epochs, the learning rate is halved. Training is terminated when the improvement is

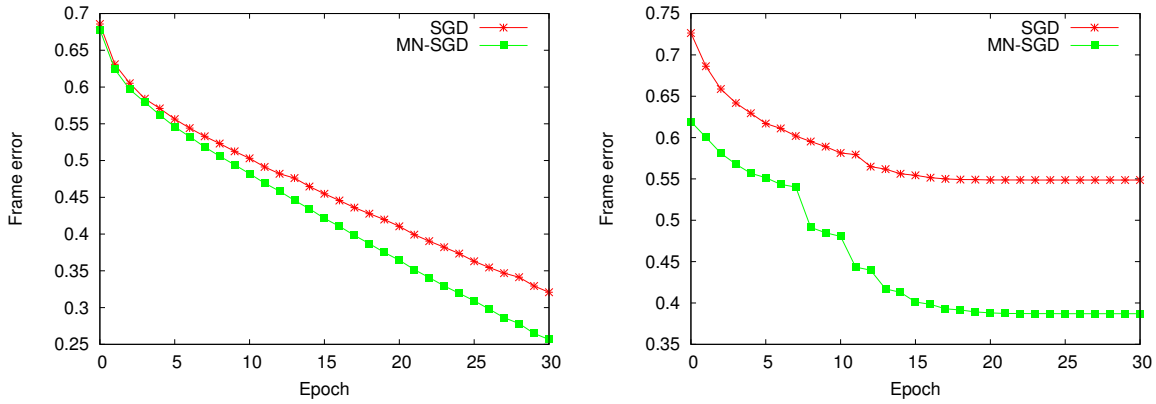

Fig. 1. Training frame classification error of SGD and MN-SGD for full-sized models (left) and models with bottlenecks of size 256 (right). Training has been initialized with pre-trained models. Note that already the pre-training results of SGD and MN-SGD differ strongly.

below 0.1%. This learning rate strategy works reasonably well and generalizes to new datasets without further tuning of hyperparameters. Controlling the learning rate by the CV performance effectively avoids overfitting. However, we noted that Newbob decays the learning rate too quickly. A better strategy is to reduce the learning rate only when there is an additional epoch without sufficient improvement on CV. We refer to this learning rate strategy as Newbob+/CV.

For our purpose, the Newbob(+)/CV strategies are not well suited, because they prevent aggressive optimization on the training data. Therefore, the effect of the optimization algorithm is weakened. Instead, we use Newbob+, but replace the validation set by a representative subset of the training data. Overfitting is avoided by early stopping, *i.e.,* we use the model that performs best on the development data. We refer to this learning rate strategy as Newbob+/Train.

2.5. Improving generalization performance

A general problem of more efficient optimization algorithms on machine learning tasks is that higher training accuracy can cause overfitting. A standard mean against overfitting is the use of regularization, *e.g.* ℓ_2-regularization. However, in our experiments we could not gain improvements in recognition word error rate by using ℓ_2-regularization and therefore were searching for an alternative.

Overfitting can also be avoided by restricting the model size. This has the additional benefit of accelerating both training *and* recognition. The DNN size can be reduced by decreasing the number of layers or hidden nodes per layer. But typically, the best results are achieved with larger models in combination with early stopping or Newbob. Recently, a more sophisticated approach for reducing the size of DNNs has been proposed. Sainath et al. [15] factored the weight matrices into the product of two smaller matrices. This is equivalent with inserting a linear bottleneck between two layers of the network. Sainath et al. found this technique to be effective for the output layer, but not for the hidden layers. They conjectured that these do not have an underlying structure that allows for a low-rank factorization. Xue et al. [16] confirmed that linear bottlenecks in all layers degrade performance when the DNN is trained with SGD from scratch. But they could achieve a factorization of the hidden layers by first training a full-sized model, then factorizing the weight matrices by means of singular value decomposition (SVD), and finally retraining the model. Using this approach, they could compress models by 80% without loss in accuracy.

Sainath et al.'s approach reduces the computational costs of DNNs in training and recognition but is limited to the output layer only. On the other hand, all layers can be compressed using Xue et al.'s approach, which accelerates recognition, but makes training even more expensive. The value of the linear bottleneck structure as a regularization method has not been identified in [15, 16]. Our experiments show that using MN-SGD allows for a factorization of all layers when training from scratch. We even benefit from the factorization as a regularization method.

3. EXPERIMENTAL RESULTS

We validated our proposed approach on the English Quaero corpus, a broadcast conversations recognition task. Our baseline system is a simplified version of our evaluation system [21], which performed best in the Quaero evaluations 2010, 2011, and 2012. All models are trained on a 50-hour subset of the training data. The development and test corpora (quaero-eval10 and quaero-eval11) consist of 3.7 and 3.3 hours of speech respectively.

Our GMM-HMM baseline uses MFCC features with vocal tract length normalization and is trained according to the maximum likelihood criterion. The GMM has a pooled, diagonal covariance matrix and $750k$ densities. 4500 context dependent states are modeled. The recognition lexicon consists of $150k$ words. The language model is a smoothed 4-gram, trained on roughly four billion words. The GMM achieves 24.3% word error rate (WER) on the development data and 31.2% WER on the test data.

For our experiments with neural networks, we used the hybrid approach [22]. All neural networks are trained according to the cross-entropy criterion with the 4500 context-dependent states as outputs. Ten percent of the training data have been removed for cross-validation. The input to the neural networks is a 493-dimensional vector which is built up from the 16-dimensional MFCC vectors in a context window of size 17 and all first and second derivatives which can be computed within this window. The global mean and variance of the input features are normalized before training. The DNN baseline has six hidden layers with sigmoid activation function and 2048 nodes, and a softmax output layer. Training is initialized with a supervised layerwise pre-training as described in [2]. Pre-training has been performed either with SGD or MN-SGD. The mini-batch size for SGD and MN-SGD has been set to 512. All experiments are performed with our open source DNN tool which is part of RASR [23]. The complete training is performed on a GPU. The training

Table 1. Experimental results of SGD and MN-SGD. The first two columns list the size of the bottleneck, the number of parameters of the model, and the reduction in comparison to the full model. The third column specifies whether Newbob+/CV or Newbob+/Train has been used. Columns four to six show the results of SGD: the epoch that has been used in recognition, the frame classification error on the training data at this epoch, and the WER (%) on the development and test data. Analogously, columns seven to eleven show the results of MN-SGD.

			SGD				MN-SGD			
Bottleneck	**Params**	**Newbob**	**Ep.**	**FCE train**	**WER dev**	**WER test**	**Ep.**	**FCE train**	**WER dev**	**WER test**
-	31.2M	CV	10	51.5	19.2	25.4	7	51.9	19.5	25.7
-	31.2M	Train	20	41.1	18.7	24.7	18	38.7	19.0	25.5
512	14.8M (52.3%)	Train	21	49.8	18.9	25.0	9	51.3	18.7	24.7
256	7.9M (74.5%)	Train	19	54.9	19.7	25.7	21	38.7	18.4	24.2
128	4.9M (85.6%)	Train	13	59.9	21.5	27.9	18	47.3	18.6	24.2
64	2.8M (91.2%)	Train	16	61.4	22.4	28.8	15	53.5	19.3	25.3

time per epoch for a full model is roughly one hour.

Our experimental results are summarized in Table 1. The DNN baseline trained with Newbob+/CV achieves 19.2% on the development data, which is a relative improvement of 21.0% in comparison to the GMM baseline. For these experiments, we applied the Newbob termination criterion (see Subsection 2.4) and used the final model for recognition. Training required ten epochs.

Our results show that Newbob+/CV decreases the learning rate too quickly. The Newbob+/Train learning rate strategy not only shows clearer effects on the training accuracy than Newbob+/CV, but also improves recognition performance notably. We also trained a model with SGD and the conventional Newbob/CV learning rate strategy for comparison. This model only achieves 19.8% WER on the development data and 25.7% WER on the test data. We did not observe gains in the CV frame error by using Newbob+/Train instead of Newbob+/CV. Hence, the improvements must be due to a mismatch between frame and word error rate. A possible explanation might be that the frame error of easy classes reaches its optimum at another point than classes which are more discriminative in recognition. In our opinion, this observation requires further analysis.

As can be seen in Figure 1 (left), MN-SGD converges faster than SGD. In addition, Newbob/CV+ terminates the MN-SGD training already after seven epochs instead of the ten epochs of SGD. But since we use very large models, which have more than twice as many parameters as the number of training samples, there is nothing to gain in terms of word error rate. The training error can be reduced arbitrarily by continuing training. Note that the best error rates that we achieved with SGD were obtained with these large networks.

In order to reduce overfitting, we investigated the use of ℓ_2-regularization. But although we could obtain improvements in the CV frame classification error by up to 4.0% absolute, we could not obtain any improvement in word error rate on the development data. This again shows that the frame classification error is not a good measure for the quality of a neural network in a hybrid speech recognition system.

In a second step, we considered models of smaller size with linear bottlenecks, as described in Subsection 2.5. We added a linear bottleneck between all hidden-to-hidden connections and the hidden-to-output connection. As already observed by [15, 16], the performance of such networks degrades when they are trained with SGD, see Table 1. Word error rate already slightly degrades with a reduction of the parameters by a factor of two. Reducing the number of parameters further, increases the word error rate strongly. This behavior is the reason why Xue et al. [16] first trained a full-sized network, then applied SVD to the weight matrices, and finally retrained the reduced model.

Using MN-SGD, training behaves completely differently. Up to a certain point, the error rate decreases with the model size due to less overfitting. The best result of 18.4% on the development data and 24.2% WER on the test data is obtained with a model that is roughly four times smaller than the full model. We still achieve improvements on the development and test data with a parameter reduction of 85.6%.

4. DISCUSSION

We proposed MN-SGD, a new stochastic second-order optimization algorithm. We have proven convergence of MN-SGD in a convex setting. In our experiments, we showed that MN-SGD converges faster than SGD, thus training time is reduced.

An interesting related approach is [24], where dynamically transformed activation functions are proposed. The idea is to achieve zero mean activations too. But in contrast to our work, this method is not applicable to arbitrary network structures. While our proposed approach is a general optimization algorithm which can be applied to any neural network, [24] requires additional skip-connections.

Another contribution of our work is an improvement to the widely used Newbob strategy. We observed that Newbob decays the learning rate too quickly. Furthermore, it is based on the frame classification error which does not correlate well with the word error rate.

We found that a recently proposed model factorization [15] is a very effective means against overfitting. In addition, the reduced size of the factorized models accelerates training as well as recognition. Using SGD, this model factorization can only be applied to the output layer. The training method of [16] allows for a factorization of all layers, but requires to train an unfactorized model first. With our proposed algorithm, we trained models where all layers are factorized from scratch. The number of parameters of the factorized model can be reduced by more than eighty percent and still improvements in word error rate are obtained.

In future work, we want to apply MN-SGD to bottleneck networks used for GMM tandem systems. A comparison of MN-SGD with a full second-order algorithm as the Hessian-Free algorithm would be of interest too.

5. REFERENCES

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

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011, pp. 437–440.

[3] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. Interspeech*, 2012.

[4] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A.-R. Mohamed, "Making deep belief networks effective for large vocabulary continuous speech recognition," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*. IEEE, 2011, pp. 30–35.

[5] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, pp. 503–528, 1989.

[6] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. MIT Press, 2012, pp. 1232–1240.

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

[8] J. Martens, "Deep learning via hessian-free optimization," in *Proc. of the 27th Int. Conf. on Machine Learning*, vol. 951, 2010, p. 2010.

[9] S. Wiesler, J. Li, and J. Xue, "Investigations on hessian-free optimization for cross-entropy training of deep neural networks," in *Proc. Interspeech*, Lyon, France, Aug. 2013, pp. 3317–3321.

[10] Y. LeCun and L. Bottou, "Efficient backprop," in *Neural networks: Tricks of the trade*, G. Orr and K. Müller, Eds. New York, USA: Springer, 1998, pp. 546–546.

[11] N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, March 2007, pp. 436–443.

[12] 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.

[13] Y. LeCun, I. Kanter, and S. A. Solla, "Second order properties of error surfaces: Learning time and generalization," in *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann Publishers Inc., 1990, pp. 918–924.

[14] S. Wiesler and H. Ney, "A convergence analysis of log-linear training," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., Granada, Spain, Dec. 2011, pp. 657–665.

[15] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2013.

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

[17] P. Sunehag, J. Trumpf, S. V. N. Vishwanathan, and N. N. Schraudolph, "Variable metric stochastic approximation theory," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, vol. 5, Clearwater Beach, Florida, 2009, pp. 560–566.

[18] H. E. Robbins and D. O. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Proc. Sympos. Optimizing Methods in Statistics*. Ohio, USA: Academic Press, New York, 1971, pp. 233–257.

[19] A. Senior, G. Heigold, M. Ranzato, and K. Yang, "An empirical study of learning rates in deep neural networks for speech recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, vol. 1, 2013, pp. 6724–6728.

[20] "Quicknet." [Online]. Available: http://www1.icsi.berkeley.edu/Speech/qn.html

[21] M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A. E. Mousa, S. Hahn, D. Nolden, R. Schlüter, and H. Ney, "The RWTH 2010 quaero ASR evaluation system for english, french, and german," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Prague, Czech, May 2011, pp. 2212–2215.

[22] H. A. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*. Springer, 1994, vol. 247.

[23] S. Wiesler, A. Richard, P. Golik, R. Schlüter, and H. Ney, "RASR/ANN: The RWTH Aachen University open source neural network toolkit for speech recognition," in *(submitted to) Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2014.

[24] T. Raiko, H. Valpola, and Y. LeCun, "Deep learning made easier by linear transformations in perceptrons," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, 2012, pp. 924–932.