FAST DNN TRAINING BASED ON AUXILIARY FUNCTION TECHNIQUE

Dung T. Tran[1], *Nobutaka Ono*[2,3], *Emmanuel Vincent*[1]

[1]Inria, Villers-lès-Nancy, F-54600, France
[2]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
[3]The Graduate University for Advanced Studies (SOKENDAI), Japan
dung.tran@inria.fr

ABSTRACT

Deep neural networks (DNN) are typically optimized with stochastic gradient descent (SGD) using a fixed learning rate or an adaptive learning rate approach (ADAGRAD). In this paper, we introduce a new learning rule for neural networks that is based on an auxiliary function technique without parameter tuning. Instead of minimizing the objective function, a quadratic auxiliary function is recursively introduced layer by layer which has a closed-form optimum. We prove the monotonic decrease of the new learning rule. Our experiments show that the proposed algorithm converges faster and to a better local minimum than SGD. In addition, we propose a combination of the proposed learning rule and ADAGRAD which further accelerates convergence. Experimental evaluation on the MNIST database shows the benefit of the proposed approach in terms of digit recognition accuracy.

Index Terms— DNN, back-propagation, auxiliary function technique, gradient descent, adaptive learning rate.

1. INTRODUCTION

Deep neural networks have been become a hot topic and have been successfully applied for many classification problems such as speech recognition [1–3], speech separation [4–6], robust speech recognition [7–9], language modeling [10,11], and image classification [12]. DNN training is a highly non-linear optimization problem. The most widely used optimization algorithm for DNN training is stochastic gradient descent (SGD) [13]. Using SGD requires tuning of parameters such as the learning rate. When the rate is too small, it leads to slow convergence. In contrast, a too large learning rate causes instability or divergence. There are many approaches to overcome the sensitivity to learning rate tuning. Many such approaches are semi-Newton approaches based on an approximation of the Hessian matrix [14–16]. An adaptive learning rate method was proposed by Duchi et al [17] where the learning rate is computed by dividing the global learning rate by the square of the accumulated gradients of all past iterations. Mean-normalized stochastic gradient descent [18] was proposed recently and shows significantly faster convergence than conventional SGD. In this paper we introduce a new learning rule for neural networks based on an auxiliary function technique without parameter tuning and we analyze its effectiveness by comparing it with existing methods such as gradient-based back-propagation.

2. BACKGROUND

2.1. NN training

Let us consider an N-layer neural network (NN). Let k_n be the number of elements (neurons) in the n-th layer, p the data index, and $z_{j,p}^{(n)}$ the output from the j-th element at the n-th layer for the p-th data. P is number of data samples. Here we define $x_{i,p}^{(n)}$ as the input to the i-th element. Let $w_{ij}^{(n)}$ be the weight from the j-th element to the i-th element and $u_i^{(n)}$ be the i-th element of the bias term between the n-th and the $(n+1)$-th layer. The neural network can be defined as

$$x_{i,p}^{(n+1)} = \sum_{j=1}^{k_n} w_{ij}^{(n)} z_{j,p}^{(n)} + u_i^{(n)} \tag{1}$$

$$z_{i,p}^{(n+1)} = f(x_{i,p}^{(n+1)}) \tag{2}$$

where f represents a nonlinear function. Possible activation functions include sigmoid, tangent hyperbolic, rectified linear unit [19] and maxout [20] functions. The network is trained by minimizing a certain loss function, such as the squared Euclidean distance or the cross-entropy [21].

In the following, we consider the tangent hyperbolic function and the squared Euclidean loss. The objective function can be expressed as

$$\mathbf{E} = \frac{1}{2}\sum_{p=1}^{P}\sum_{i=1}^{k_N}(z_{i,p}^{(N)} - y_{i,p})^2 + \frac{\lambda}{2}\sum_{n=1}^{N}\sum_{i}\sum_{j}(w_{i,j}^{(n)})^2 \tag{3}$$

where the first term is the squared Euclidean distance between the NN output and the target and the second term is a regularization term that avoids over-fitting. The problem here is to find a set of $w_{ij}^{(n)}$ and $u_i^{(n)}$ that minimize (3). The most widely used optimization algorithm used for NN training is SGD:

$$w_{ij,t+1}^{(n)} = w_{ij,t}^{(n)} + \alpha\, \partial\mathbf{E}_t/\partial w_{ij,t}^{(n)}. \tag{4}$$

The adaptive learning rate method ADAGRAD [17] is another popular algorithm whose learning rule is given by

$$w_{ij,t+1}^{(n)} = w_{ij,t}^{(n)} + \alpha\frac{\partial\mathbf{E}_t/\partial w_{ij,t}^{(n)}}{\sqrt{\sum_t(\partial\mathbf{E}_t/\partial w_{ij,t}^{(n)})^2}} \tag{5}$$

where α is a fixed learning rate which is set manually and t is the iteration index. Note that these gradient based learning rules can also be applied for $u_i^{(n)}$.

2.2. Auxiliary function technique

Auxiliary function based optimization [22–26] has recently become popular in other fields as exemplified by, e.g., the audio source separation techniques HPSS [27] and AuxIVA [28]. Following that, to avoid learning rate tuning and derive an effective learning rule, we introduce an auxiliary function technique for NN training. Instead of minimizing the objective function, an auxiliary function is introduced and the minimization procedure is applied to that auxiliary function. Let us express the general optimization problem as:

$$w^{(n)} = \underset{w^{(n)}}{\operatorname{argmin}} \mathbf{E}(w^{(n)}). \tag{6}$$

In the auxiliary function technique, an auxiliary function $\mathbf{Q}$ is designed that satisfies

$$\mathbf{E}(w^{(n)}) \leq \mathbf{Q}(w^{(n)}, w_0^{(n)}) \tag{7}$$

for all $w^{(n)}$ and all values of the auxiliary variable $w_0^{(n)}$. The equality is satisfied if and only if $w^{(n)} = w_0^{(n)}$. Now, starting from an initial parameter value $w_0^{(n)}$, we can find the optimal value of $w^{(n)}$ that minimizes $\mathbf{Q}(w^{(n)}, w_0^{(n)})$:

$$w_1^{(n)} = \underset{w^{(n)}}{\operatorname{argmin}} \mathbf{Q}(w^{(n)}, w_0^{(n)}). \tag{8}$$

As a result

$$\mathbf{E}(w_1^{(n)}) \leq \mathbf{Q}(w_1^{(n)}, w_0^{(n)}) \leq \mathbf{Q}(w_0^{(n)}, w_0^{(n)}) = \mathbf{E}(w_0^{(n)}). \tag{9}$$

The procedure can be applied iteratively. The inequality in (9) guarantees the monotonic decrease of the objective function. When the auxiliary function is quadratic, this algorithm converges linearly but at a typically faster rate than SGD [29]. Also, it does not require any parameter tuning provided that (8) can be written in closed form.

3. QUADRATIC AUXILIARY FUNCTION FOR NEURAL NETWORK

We derive two auxiliary functions at each layer: one relating to the nonlinear activation function (2) and one relating to the linear combination (1). We then combine these two auxiliary functions into a single minimization scheme.

3.1. First quadratic auxiliary function

For simplicity, let us first omit the indices i, p, and n, and derive an auxiliary function for

$$\begin{aligned}\mathbf{E} &= (z-y)^2 \\ &= \tanh^2(x) - 2y\tanh(x) + y^2.\end{aligned} \tag{10}$$

The regularization term in (3) will be discussed later on. We derive a quadratic auxiliary function using the following lemma.

Lemma 3.1 *For any positive real numbers x and x_0 and any real number y, the following inequality is satisfied:*

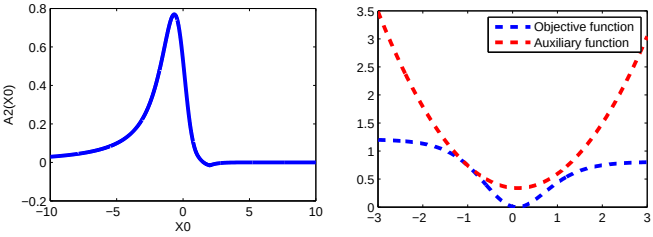
$$(\tanh(x) - y)^2 \leq ax^2 - 2bx + c = \boldsymbol{Q} \tag{11}$$

where

$$a = A_1(x_0) + |y|A_2(-\sigma x_0) \tag{12}$$

$$b = y[\sigma x_0 A_2(-\sigma x_0) + \operatorname{sech}^2(x_0)] \tag{13}$$

$$\begin{aligned}c = &- A_1(x_0)x_0^2 + \tanh^2(x_0) + |y|A_2(-\sigma x_0)x_0^2 \\ &+ 2y\operatorname{sech}^2(x_0)x_0 - 2y\tanh(x_0) + y^2\end{aligned} \tag{14}$$



(a) (b)

Fig. 1. a) The shape of A_2; b) Illustration of generation of auxiliary function from $x_0 = -1, y = 0.1$.

and

$$\sigma = \operatorname{sign}(y) \tag{15}$$

$$A_1(x_0) = \frac{\operatorname{sech}^2(x_0)\tanh(x_0)}{x_0} \tag{16}$$

$$A_2(x_0) = \sup_x \frac{\tanh(x) - \tanh(x_0) - \operatorname{sech}^2(x_0)(x - x_0)}{(1/2)(x - x_0)^2} \tag{17}$$

The equality sign is satisfied if and only if $x = x_0$.

Proof The objective function (10) includes two x terms: $\tanh^2(x)$ and $2y\tanh(x)$. According to [30, Theorem 4.5], when $f(x)$ is an even, differentiable function on $\mathbb{R}$ such that the ratio $f'(x)/x$ is decreasing on $(0, \infty)$, the inequality

$$f(x) \leq g(x) = \frac{f'(x_0)}{2x_0}(x^2 - x_0^2) + f(x_0) \tag{18}$$

is satisfied.

Also, according to [30], if a function $f(x)$ is differentiable in x, and

$$A(x_0) = \sup_z \frac{f(x) - f(x_0) - f'(x_0)(x - x_0)}{\frac{1}{2}(x - x_0)^2} \tag{19}$$

has a finite positive value, then

$$f(x) \leq f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}A(x_0)(x - x_0)^2 \tag{20}$$

is satisfied for all x and x_0. By substituting $f(x) = \tanh^2(x)$ into (18), and $f(x) = y\tanh(x)$ where y can be positive or negative into (20), we have (11). ∎

Note that $A_2(x_0)$ can not computed in closed form. But we can prepare a table of $A_2(x_0)$ in advance. Fig. 1 shows the shape of $A_2(x_0)$ and an example of the auxiliary function.

3.2. Second auxiliary function for separating variables

Now that we have derived an auxiliary function as a function of the inputs $x_{i,p}^{(n)}$ in one layer, we need to propagate it down to the outputs $z_{j,p}^{(n-1)}$ of the previous layer. Once again, let us omit the indices i, p, and n, and consider

$$x = \sum_j w_j z_j + u. \tag{21}$$

We wish the auxiliary function to decompose as a sum of terms, each relating to one neuron z_j, such that Lemma 3.1 can be applied again at the lower layer. Note that plugging (21) into (11) induces some cross-terms of the form $z_j z_{j'}$. In order to separate the contribution of each z_j additively, we apply the following lemma.

Lemma 3.2 *For $x = \sum_j w_j z_j + u$, the inequality*

$$ax^2 + bx + c \leq \sum_{j=1}^{J} [aJw_j^2(z_j - y_j)^2 + aJ(\beta_j^2 - y_j^2)] + au^2 + bu + c = \mathbf{R} \quad (22)$$

is satisfied for any β_j such that $\sum_{j=1}^{J} \beta_j = 0$ where

$$y_j = \frac{(2aJ\beta_j - 2au - b)w_j}{2aJw_j^2}. \quad (23)$$

The equality is satisfied if and only if

$$\beta_j = w_j z_j - \frac{1}{J}\sum_{j=1}^{J} w_j z_j. \quad (24)$$

Proof Generally for any s_j and β_j, minimizing $\sum_{j=1}^{J}(s_j - \beta_j)^2$ under the constraint that $\sum_{j=1}^{J} \beta_j = 0$, we have the inequality

$$\left(\sum_{j=1}^{J} s_j\right)^2 \leq J\sum_{j=1}^{J}(s_j - \beta_j)^2. \quad (25)$$

Applying this inequality to the case where $s_j = w_j z_j$, we obtain inequality (22). ∎

3.3. Recursively deriving auxiliary functions

Based on Lemmas 3.1 and 3.2, we now have two kinds of auxiliary functions for the first term of **E** in (3) with the following forms:

$$\mathbf{Q}^{(N)} = \sum_p \sum_i a_{i,p}^{(N)} (x_{i,p}^{(N)})^2 + b_{i,p}^{(N)} x_{i,p}^{(N)} + c_{i,p}^{(N)} \quad (26)$$

$$\mathbf{R}^{(N)} = \sum_p \sum_i \sum_j a_{i,p}^{(N)} J^{(N-1)} (w_{ij}^{(N-1)})^2 (z_{j,p}^{(N-1)} - y_{j,p}^{(N-1)})^2 + a_{i,p}^{(N)} (u_i^{(N-1)})^2 + b_{i,p}^{(N)} u_i^{(N-1)} + c_{i,p}^{(N)} + const \quad (27)$$

where $a_{i,p}^{(N)}$, $b_{i,p}^{(N)}$, $c_{i,p}^{(N)}$, and $y_{j,p}^{(N-1)}$ are defined in (12), (13), (14), and (23), respectively, $J^{(N-1)}$ is the number of neurons in the $(N-1)$-th layer, and *const* represents a term unrelated to optimization.

The expression of $\mathbf{R}^{(N)}$ is similar to that of the original objective function in that it is a sum of squared error terms of the form $(z - y)^2$. Therefore, we can recursively apply the above two lemmas in decreasing layer order n in a similar fashion as conventional back-propagation and obtain a sequence of auxiliary functions such that

$$\mathbf{E} \leq \mathbf{Q}^{(N)} \leq \mathbf{R}^{(N)} \leq \mathbf{Q}^{(N-1)} \leq \mathbf{R}^{(N-1)} \cdots \quad (28)$$

which guarantee the monotonic decrease of the objective function overall.

The optimal values of $w_{ij}^{(n-1)}$ and $u_i^{(n-1)}$ can be obtained by minimizing the sum of $\mathbf{Q}^{(n)}$ and the quadratic regularization term in (3). This minimization is costly as it involves some quadratic cross-terms. Noticing that the role of w_j and z_j in (21) is symmetric, we can derive a separable majorizing function for $\mathbf{Q}^{(n)}$ which has the same expression as (22) where the variables w_j and z_j are switched in (22) and (23). Each $w_{ij}^{(n-1)}$ and $u_i^{(n-1)}$ can then be separately computed by minimizing the sum of this majorizing function and the regularization term instead.

4. ALGORITHMS

4.1. Auxiliary function based NN training

In summary, each iteration of the auxiliary function based NN training (AuxNNT) algorithm is described in Algorithm 1.

Algorithm 1 Auxiliary function based method (AuxNNT)

Require: Initial parameters $w_{ij}^{(n)}$, $u_i^{(n)}$ for all i, j, n

Compute forward pass using (1) and (2).

for $n = N$ to 2

1. Compute auxiliary function coefficients as follows:

$$\sigma_{i,p}^{(n)} = sign(y_{i,p}^{(n)})$$
$$a_{i,p}^{(n)} = A_1(x_{i,p}^{(n)}) + |y_{i,p}^{(n)}| A_2(-\sigma_{i,p}^{(n)} x_{i,p}^{(n)})$$
$$b_{i,p}^{(n)} = y_{i,p}^{(n)} [\sigma_{i,p}^{(n)} x_{i,p}^{(n)} A_2(-\sigma_{i,p}^{(n)} x_{i,p}^{(n)}) + \mathrm{sech}^2(x_{i,p}^{(n)})]$$
$$\beta_{i,j,p}^{(n)} = w_{ij}^{(n-1)} z_j^{(n-1)} - \frac{1}{J^{(n-1)}} \sum_{j=1}^{J} w_{ij}^{(n-1)} z_j^{(n-1)}$$
$$y_{j,p}^{(n-1)} = \frac{\sum_i \left(2a_{i,p}^{(n)} J^{(n-1)} \beta_{i,j,p} - 2a_{i,p}^{(n)} u_i^{(n-1)} - b_{i,p}^{(n)}\right) w_{ij}^{(n-1)}}{\sum_i 2a_{i,p}^{(n)} J^{(n-1)} (w_{ij}^{(n-1)})^2}$$

2. Update the parameters in $(n-1)$-th layer as follows:

$$w_{ij}^{(n-1)} = \frac{\sum_p \left(2a_{i,p}^{(n)} J^{(n-1)} \beta_{i,j,p} - 2a_{i,p}^{(n)} u_i^{(n-1)} - b_{i,p}^{(n)}\right) z_{j,p}^{(n-1)}}{\sum_p 2a_{i,p}^{(n)} J^{(n-1)} (z_{j,p}^{(n-1)})^2 + \frac{\lambda}{PI^{(n)}}}$$
$$u_i^{(n-1)} = \frac{\sum_p \left(-2a_{i,p}^{(n)} J^{(n-1)} \sum_j \left(w_{ij}^{(n-1)} z_{j,p}^{(n-1)}\right) - b_{i,p}^{(n)}\right)}{\sum_p 2a_{i,p}^{(n)} J^{(n-1)}}$$

endfor

4.2. Hybrid algorithm

One benefit of the proposed AuxNNT method is that it can be combined with any gradient based method such as ADAGRAD [17]. The gradient can be computed at any point based on the parameters of the auxiliary function with lower computational effort. We observed in preliminary experiments that, when the change in the parameter values from the previous to the current iteration is small, ADAGRAD results in a greater decrease of the objective function than AuxNNT because the learning rate at the current iteration increases.

We propose an hybrid approach called Hybrid auxNNT that takes advantage of both methods. Specifically, when the change in the parameter values is small, several iterations of ADAGRAD are performed. We then select the iteration number for which the gradient is largest and continue with AuxNNT onwards, until the change in the parameter values becomes small again. This hybrid method relies on two tuning parameters: a parameter change threshold ϵ and the number t_{eval} of ADAGRAD iterations. The details of each

iteration of this hybrid algorithm are described in Algorithm 2. Note that ∇**E2** is initialized by 0 at first iteration.

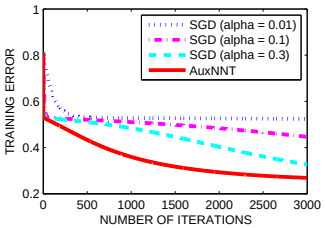
Algorithm 2 Hybrid method (Hybrid AuxNNT)

Require: Initial parameters $w_{ij}^{(n)}$, $u_i^{(n)}$ for all i, j, n

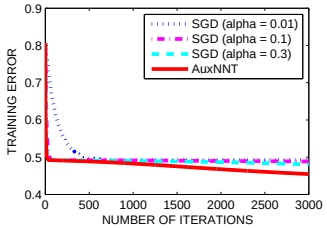
Require: global learning rate α, threshold ϵ, number of gradient evaluations t_{eval}.

1. Compute forward pass.
2. Compute auxiliary function coefficients using Algorithm 1.
3. Update the parameters for all layers using Algorithm 1.
4. Fold w_{ij} and u_i into a vector θ.
5. Compute gradient $\partial \mathbf{E}/\partial\theta_k$.
6. Accumulate square of gradient $\Delta \leftarrow \Delta + (\partial \mathbf{E}/\partial\theta_k)^2$.
7. Compute $\delta_{\theta_k} = \theta_{k,\text{previous}} - \theta_{k,\text{current}}$

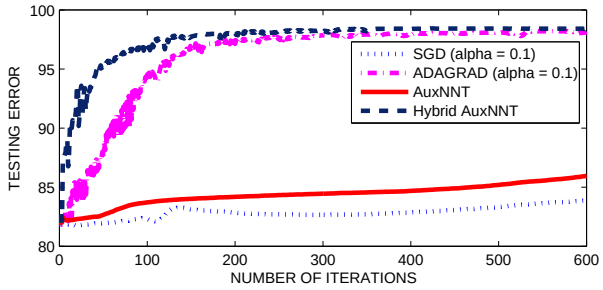
if $\sum_k (\delta_{\theta_k})^2 < \epsilon$ **then**
  for $t = 1$ **to** t_{eval} **do**
    Compute gradient $\partial \mathbf{E}/\partial\theta_{k,t}$.
    $\theta_{k,t+1} := \theta_{k,t} + \alpha \frac{\partial \mathbf{E}_t/\partial\theta_{k,t}}{\sqrt{\Delta + \sum_t (\partial \mathbf{E}_t/\partial\theta_{k,t})^2}}$
  end for
  $t_{\max} = \arg\max_{t \in \{1 \ldots t_{\text{eval}}\}} \sum_k (\partial \mathbf{E}/\partial\theta_{k,t})^2$
  $\theta_k = \theta_{k,t_{\max}}$.
end if



(a) 1 hidden layer autoencoder



(b) 2 hidden layers autoencoder



(c) 2 hidden layers neural network

Fig. 2. Training and testing progress on MNIST database.

5. EXPERIMENTAL EVALUATION

To analyze the effectiveness of the proposed methods, we conducted two experiments on the MNIST handwritten digits database [12]. In both experiments, all parameters were initialized to random numbers drawn uniformly from the interval $-\sqrt{6/(n_{\text{in}} + n_{\text{out}} + 1)}$, $\sqrt{6/(n_{\text{in}} + n_{\text{out}} + 1)}$ where n_{in} is the the number of inputs feeding into a neuron and n_{out} is the the number of units that a neuron feeds into.

In the first experiment, we learned an auto-encoder and analyzed the value of the objective function, a.k.a the training error, over the iterations. Note that the objective function for the auto-encoder classically includes a sparsity term which we did not include here. Two auto-encoders were built: the first one has one hidden layer with 25 neurons and the second one has two hidden layers with 25 neurons in each hidden layer. The input and output layers have 64 neurons. To generate a training set, we sample 10000 8×8 image patches and concatenate them into a 64×10000 matrix. Fig. 2 (a) and (b) show that the AuxNNT method results in monotonic decrease of the training error and converges faster and to a better solution than SGD. The computational cost of one iteration of SGD and AuxNNT is equal to 0.02 s and 0.047 s for 100 samples, respectively, for both three-layer and four-layer networks.

In the second experiment, we analyse the results in terms of classification accuracy over test data. A simple neural network was designed where the input is a 28×28 image folded into a 784 dimensional vector and the output is the 10 dimensional posterior probability vector over the 10 digit classes. For example if the target is the digit "2", then the second element of the output vector is equal to 1 and the 9 remaining elements are equal to 0. There are two hidden layers with 25 neurons for each layer. When decoding, the recognized digit corresponds to the biggest element in the output vector. The training data contains 10000 image samples. The optimal learning rate was set for ADAGRAD and SGD. Fig. 2 (c) shows that AuxNNT outperforms SGD and Hybrid AuxNNT outperforms all other techniques, including ADAGRAD. Using the Hybrid AuxNNT method, we achieved 98.4% accuracy while with ADAGRAD the accuracy was 98.1%.

To reduce the computation cost of the Hybrid AuxNNT method, we used 1000 samples only to compute the gradient since we found in preliminary experiments that using all data did not significantly affect performance. All data were used to compute the gradient for ADAGRAD, however, since using only 1000 samples was found to degrade ADAGRAD's performance. The computation cost of one iteration of the Hybrid AuxNNT method is equal to 0.052 s for 100 samples.

6. CONCLUSION

A new learning rule was proposed for neural networks based on an auxiliary function technique without parameter tuning. Instead of minimizing the objective function, a quadratic auxiliary function is recursively introduced layer by layer which has a closed form optimum. We also proved the monotonic decrease of the new update rule. Experimental results on the MNIST database showed that the proposed algorithm converges faster and to a better solution than SGD. In addition, we found the combination of ADAGRAD and the proposed method to accelerate convergence and to achieve a better performance than ADAGRAD alone. In the future, we will seek to improve the proposed AuxNNT method by using information from previous iterations as well as applying it to robust speech recognition and speech separation tasks.

7. ACKNOWLEDGMENT

This research was partially supported by Grant-in-Aid for Challenging Exploratory Research (26540090) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan. The authors acknowledge Dr. Hubert Soyer and Dr. Trung-Kien Le for useful discussion.

8. REFERENCES

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

[2] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, pp. 2345–2349.

[4] Y. Wang and D. L. Wang, "Towards scaling up classification-based speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1381–1390, 2013.

[5] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. ICASSP*, 2014, pp. 1562–1566.

[6] F. Weninger, J. Le Roux, J. R. Hershey, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *Proc. GlobalSIP*, 2014.

[7] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of noise robustness of deep neural networks," in *Proc. ICASSP*, 2013, pp. 7398–7402.

[8] S. Renals and P. Swietojanski, "Neural networks for distant speech recognition," in *Proc. HSCMA*, 2014, pp. 172–176.

[9] C. Weng, D. Yu, M. L. Seltzer, and J. Droppo, "Single-channel mixed speech recognition using deep neural networks," in *Proc. ICASSP*, 2014, pp. 5632–5636.

[10] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, 2010, pp. 1045–1048.

[11] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *Proc. NAACL-HLT Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, 2012, pp. 20–28.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[13] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, pp. 9–42. Cambridge University Press, 1998.

[14] S. Becker and Y. LeCun, "Improving the convergence of back-propagation learning with second order methods," Tech. Rep., Department of Computer Science, University of Toronto, 1988.

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

[16] 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.

[17] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," in *Conference on Learning Theory (COLT 2010)*, 2010.

[18] S. Wiesler, A. Richard, R. Schluter, and H. Ney, "Mean-normalized stochastic gradient for large-scale deep learning," in *Proc. ICASSP*, 2014, pp. 180–184.

[19] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On rectified linear units for speech processing," in *Proc. ICASSP*, 2013, pp. 3517–3521.

[20] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013.

[21] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. ICML*, 2008.

[22] J. de Leeuw, "Block relaxation algorithms in statistics," in *Information Systems and Data Analysis*, pp. 308–325. Springer, 1994.

[23] W. J. Heiser, "Convergent computing by iterative majorization: theory and applications in multidimensional data analysis," in *Recent Advances in Descriptive Multivariate Analysis*, pp. 157–189. Clarendon Press, 1995.

[24] M. P. Becker, I. Yang, and K. Lange, "EM algorithms without missing data," *Stat. Methods Med. Res.*, vol. 6, pp. 38–54, 1997.

[25] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions (with discussion)," *J. Comput. Graphical Stat.*, vol. 9, pp. 1–20, 2000.

[26] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *Amer. Statist.*, vol. 58, pp. 30–37, 2004.

[27] N. Ono, K. Miyamoto, J. Le Roux, H. Kameoka, and S. Sagayama, "Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram," in *Proc. EUSIPCO*, 2008.

[28] N. Ono, "Stable and fast update rules for independent vector analysis based on auxiliary function technique," in *Proc. WASPAA*, 2011, pp. 189–192.

[29] D. Böhning and B. G. Lindsay, "Monotonicity of quadratic-approximation algorithms," *Ann. Inst. Statist. Math.*, vol. 40, no. 4, pp. 641–663, 1988.

[30] J. de Leeuw and K. Lange, "Sharp quadratic majorization in one dimension," *Comput. Stat. Data Anal.*, vol. 53, pp. 2471–2484, May 2009.