DEEP RECURRENT REGULARIZATION NEURAL NETWORK FOR SPEECH RECOGNITION

Jen-Tzung Chien and Tsai-Wei Lu

Department of Electrical and Computer Engineering
National Chiao Tung University, Hsinchu, Taiwan 30010, ROC

ABSTRACT

This paper presents a deep recurrent regularization neural network (DRRNN) for speech recognition. Our idea is to build a regularization neural network acoustic model by conducting the hybrid Tikhonov and weight-decay regularization which compensates the variations due to the input speech as well as the model parameters in the restricted Boltzmann machine as a pre-training stage for feature learning and structural modeling. In addition, a new backpropagation through time (BPTT) algorithm is developed by extending the truncated minibatch training for recurrent neural network where the minibatch BPTT is not only performed in recurrent layer but also in feedforward layer. The DRRNN acoustic model is accordingly established to capture the *temporal correlation* in a *regularization neural network*. Experimental results on the tasks of RM and Aurora4 show the effectiveness and robustness of using DRRNN for speech recognition.

Index Terms— Recurrent neural network, model regularization, deep learning, acoustic model

1. INTRODUCTION

Deep neural network (DNN) has been widely demonstrated to achieve high performance in large vocabulary continuous speech recognition [1, 2]. Such an DNN system is typically trained from a large collection of speech utterances which cover a variety of phonetics, speakers, environments, communication channels, etc. The trained DNN based on the maximum likelihood (ML) estimation or the cross-entropy error minimization may be overtrained or mismatched with the test conditions. Also, the temporal correlation in deep representation is not well characterized in standard DNN acoustic model. This paper aims to tackle the *model regularization* and capture the *temporal correlation* for DNN speech recognition. We accordingly develop the deep recurrent regularization neural network (DRRNN) for robust speech recognition.

To deal with the regularization issue, the weight-decay regularization or Gaussian prior [3] was introduced in the cross-entropy error minimization which performed the maximum *a posteriori* training. Model complexity was controlled to avoid the over-trained system. In [4], a dropout scheme was developed to resolve the regularization issue for DNN acoustic modeling. Dropout was performed by randomly pruning a subset of hidden neurons in each layer during training procedure. Instead of pruning hidden neurons, the dropconnect randomly pruned a subset of weights between network layers [5]. To characterize the temporal dependency in sequential generation of speech samples, the recurrent neural network (RNN) was developed for acoustic modeling [6]. The RNN language model was proposed to capture the temporal dependency in lexicon generation [7]. More recently, the deep RNN was proposed for robust speech recognition [8, 9]. In [10], the truncated minibatch backpropagation through time (BPTT) was introduced to tackle the issue of non-uniform contribution when calculating the minibatch gradient.

This paper presents the Tikhonov regularization for deep RNN acoustic modeling. Our motivation is to compensate the perturbations over training samples and come out with a robust latent variable model which holds the property of invariance due to the transformation of input data. This Tikhonov regularization is combined with the weight-decay regularization in a pre-training procedure to increase the rate of convergence when estimating the DNN-hidden Markov model (HMM) system. RNN is then introduced to improve deep representation of temporal information in speech samples and their contexts. We present a new truncated minibatch training to elevate the performance of BPTT algorithm for DRRNN acoustic modeling by taking into account error backpropagation for recurrent layer and feedforward layer.

2. REGULARIZATION NEURAL NETWORK

In the training procedure of DNN, we start from a Gaussian mixture model (GMM)-HMM system. This GMM-HMM is applied to find the label of a tied state or senone $s_t = k$ corresponding to each observation frame $\mathbf{o}_t$.

2.1. DNN training

We can reliably train a DNN acoustic model based on a pre-training by using the restricted Boltzmann machines (RBMs). An efficient learning procedure is to construct a DNN structure in an unsupervised layer-by-layer manner where the lay-

978-1-4673-6997-8/15/$31.00 ©2015 IEEE

ers are made of the stacked RBMs. This pre-training stage is applied to initialize the weights of a DNN. The *contrastive divergence* (CD) algorithm was developed to train each two-layered RBM. Given the trained model structure and initial weights, the error backpropagation algorithm is applied to conduct the supervised training and fine-tune the DNN parameters. In forward pass, we calculate the outputs of hidden neurons $\mathbf{z}_t^{(l)} = \{z_{tk}^{(l)}\}$ at each time t and layer $1 \leq l < L$ where

$$z_{tk}^{(l)} = f(a_{tk}^{(l)}) = f((\mathbf{w}_k^{(l)})^T \mathbf{z}_t^{(l-1)}) \tag{1}$$

and $\mathbf{w}_k^{(l)}$ denotes the weight parameters which connect all neurons at layer $l-1$ to the neuron k at layer l. Here, the observation $\mathbf{o}_t$ is treated as the inputs at initial layer, $\mathbf{z}_t^{(0)} = \mathbf{o}_t$, and the output vector $\mathbf{y}_t$ is seen as the outputs of layer L, $\mathbf{z}_t^{(L)} = \mathbf{y}_t$. The activation function $f(\cdot)$ is assigned by the *sigmoid* function $\sigma(\cdot)$ for those neurons in all layers l except output layer L while $f(\cdot)$ is given by the *softmax* function $s(\cdot)$ for those neurons in output layer L. Each output neuron in $\mathbf{y}_t$ produces the posterior probability for a senone $p(s_t = k|\mathbf{o}_t)$. DNN training aims to minimize the cross-entropy error function $E(\mathbf{w})$ between the DNN outputs $\{y_{tk}\}$ and the target values $\{r_{tk}\}$ from N training samples $\{\mathbf{o}_t\}$

$$E_{1:N}(\mathbf{w}) = \sum_{t=1}^{N} E_t(\mathbf{w}) = -\sum_{t=1}^{N}\sum_{k=1}^{K} r_{tk} \log y_{tk} \tag{2}$$

where $\mathbf{w} = \{\mathbf{w}_k^{(l)}\}$ and target value r_{tk} of an observation $\mathbf{o}_t$ is assigned by 1 if $\mathbf{o}_t$ corresponds to state k and 0 if $\mathbf{o}_t$ is associated with the other states.

2.2. Hybrid regularization

To deal with the over-fitting problem in DNN pre-training, we modify the objective function in the stochastic gradient descent (SGD) algorithm by simultaneously considering the effects of the over-trained model parameters $\mathbf{w}$ and the variations of input data $\mathbf{o}_t$ in the regularized error function

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \lambda_1 \Omega_1(\mathbf{w}) + \lambda_2 \Omega_2(\mathbf{w}) \tag{3}$$

where λ_1 and λ_2 are the regularization parameters, $\Omega_1(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the ℓ_2 or weight-decay regularization which prevents the over-trained parameters, and $\Omega_2(\mathbf{w})$ is the Tikhonov regularization [11, 12, 13] which compensates for the variations of input data due to a shift parameter $\boldsymbol{\xi}$ via $\mathbf{o}_t \to \mathbf{o}_t + \boldsymbol{\xi}$

$$\begin{aligned}\Omega_2(\mathbf{w}) &= \frac{1}{2}\int \|\nabla \log p(\mathbf{o}_t|\mathbf{w})\|^2 p(\mathbf{o}_t) d\mathbf{o}_t \\ &= \frac{1}{2}\sum_{i=1}^{D}\left[\frac{\partial \log p(\mathbf{o}_t|\mathbf{w})}{\partial o_{ti}}\right]^2 \approx \frac{1}{2}\sum_{i=1}^{D}\sum_{k=1}^{K}(w_{ik}h_k)^2 .\end{aligned} \tag{4}$$

Here, $\mathbf{w} = \{w_{ik}\}$ denotes the weight parameters between a visible layer with D inputs $\mathbf{o}_t = \{o_{ti}\}$ and a hidden layer with M neurons $\mathbf{h} = \{h_k\}$. In Eq. (4), the likelihood function of visible data is defined under a Gaussian-Bernoulli RBM as

$$p(\mathbf{o}_t|\mathbf{w}) = \frac{\sum_{\mathbf{h}} e^{-\mathcal{E}(\mathbf{o}_t,\mathbf{h}|\mathbf{w})}}{Z(\mathbf{w})} \tag{5}$$

with a normalization term $Z(\mathbf{w})$ and an energy function $\mathcal{E}(\mathbf{o}_t, \mathbf{h}|\mathbf{w})$ given by

$$\sum_{i=1}^{D}\frac{(o_{ti} - w_{i0})^2}{2} - \sum_{k=1}^{K} w_{0k}h_k - \sum_{i=1}^{D}\sum_{k=1}^{K} o_{ti}w_{ik}h_k \tag{6}$$

where the Gaussian with unit variance is assumed, $\{w_{i0}, w_{0k}\}$ denotes the bias parameters, and $h_k = \sigma(w_{0k} + \sum_{i=1}^{D} o_{ti}w_{ik})$. After training the first stack of Gaussian-Bernoulli RBM, the outputs of hidden neurons are treated as the visible data for training the next stack of Bernoulli-Bernoulli RBM. DNN structure is stacked through a sequence of RBMs. The hybrid regularization is performed to assure the robustness of DNN model with respect to the over-trained parameters and the shift variations of visible data. The cross-entropy training is then applied to estimate the model parameters from the senone labels of training samples and calculate the senone posterior probabilities from DNN outputs.

3. DEEP RECURRENT NEURAL NETWORK

After building the regularization neural network, we further capture the temporal correlation in DNN training through the recurrent neural network (RNN). When the hidden layer l is modified as the recurrent layer m, where $1 \leq m \neq l < L$, the outputs of neurons in this layer are calculated by

$$z_{tk}^{(m)} = f((\mathbf{w}_k^{(m)})^T \mathbf{z}_t^{(m-1)} + (\mathbf{w}_k^{(mm)})^T \mathbf{z}_{t-1}^{(m)}) \tag{7}$$

where $\mathbf{w}_k^{(mm)}$ denotes the recurrent weights at layer m with the connection to neuron k.

3.1. Local gradients for error backpropagation

According to the criterion in Eq. (2), we calculate the local gradients of the neurons in output layer L and in hidden layer l

$$\delta_{tk}^{(L)} = \frac{\partial E_t(\mathbf{w})}{\partial a_{tk}^{(L)}} = \frac{\partial E_t(\mathbf{w})}{\partial y_{tk}}\frac{\partial y_{tk}}{\partial a_{tk}^{(L)}} = y_{tk} - r_{tk} \tag{8}$$

$$\boldsymbol{\delta}_t^{(l)} = \left((W^{(l+1)})^T \boldsymbol{\delta}_t^{(l+1)}\right) * \mathbf{z}_t^{(l)} * \left(\mathbf{1} - \mathbf{z}_t^{(l)}\right) \tag{9}$$

where $*$ denotes the element-wise multiplication, $\boldsymbol{\delta}_t^{(l)} = \{\delta_{tk}^{(l)}\}$, and $W^{(l)} = [\mathbf{w}_1^{(l)} \ldots \mathbf{w}_K^{(l)}]$. The local gradients from batch data $\{\mathbf{o}_1, \ldots, \mathbf{o}_N\}$ are then calculated to form the matrix $\boldsymbol{\delta}_{1:N}^{(l)} = [\boldsymbol{\delta}_1^{(l)} \ldots \boldsymbol{\delta}_N^{(l)}]$ which is applied to find the gradient over the whole training data with respect to the weight matrix $W^{(l)}$ as

$$\frac{\partial E_{1:N}}{\partial W^{(l)}} = \mathbf{z}_{1:N}^{(l-1)} (\boldsymbol{\delta}_{1:N}^{(l)})^T \tag{10}$$

where $\mathbf{z}_{1:N}^{(l-1)} = [\mathbf{z}_1^{(l-1)} \dots \mathbf{z}_N^{(l-1)}]$. Such a batch training is different from SGD training where the parameter updating is performed at each frame t by using $\mathbf{z}_t^{(l-1)}(\boldsymbol{\delta}_t^{(l)})^T$. A meaningful tradeoff between batch training and SGD training is to conduct the minibatch SGD training where the weight parameters are updated right after observing a set of M training samples $\{\mathbf{o}_1, \dots, \mathbf{o}_M\}$ where $M < N$.

Considering the updating of weight matrix in recurrent layer m in a minibatch SGD training, we calculate the local gradients for two conditions. First, the local gradient for the last frame $t = M$ with the minibatch size M has the same form as Eq. (9) since no recurrent operation is done in the end of minibatch. Second, for all the other frames within the minbatch $t < M$, their local gradients are calculated by

$$\begin{aligned}\boldsymbol{\delta}_t^{(m)} = &\left((W^{(m+1)})^T \boldsymbol{\delta}_t^{(m+1)} + (W^{(mm)})^T \boldsymbol{\delta}_{t+1}^{(m)}\right) \\ &* \mathbf{z}_t^{(m)} * \left(\mathbf{1} - \mathbf{z}_t^{(m)}\right).\end{aligned} \tag{11}$$

Figure 1 shows the forward and backward passes in DRRNN and illustrates how the local gradient $\boldsymbol{\delta}_t^{(m)}$ is obtained.

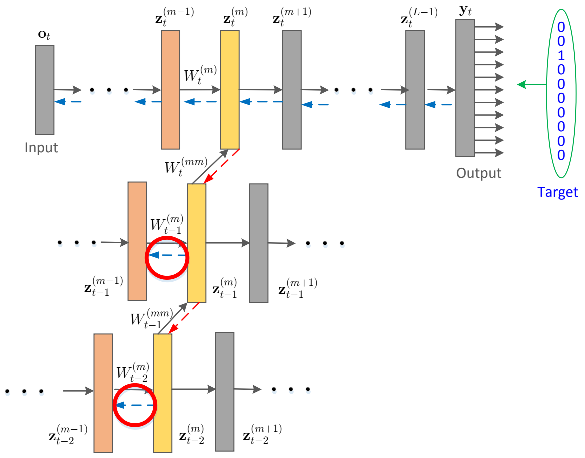

Fig. 1. Backpropagation through time and layer in deep recurrent neural network with two steps back in time $T = 2$.

3.2. Backpropagation through time and layer

Backpropagation through time (BPTT) is applied for deep RNN training over a minibatch $\{\mathbf{o}_1, \dots, \mathbf{o}_M\}$. A truncated BPTT was modified to deal with the non-uniform problem in calculation of minibatch gradient vector [10]. The recurrent weights $W^{(mm)}$ are updated by minimizing $E_{1:M}$ using

$$\begin{aligned}\sum_{t=1}^{M} \frac{\partial E_t}{\partial W^{(mm)}} &\approx \sum_{\tau=1}^{T} \sum_{t=1}^{M} \mathbf{z}_{t-\tau}^{(m)} (\boldsymbol{\delta}_{t-\tau+1}^{(m)})^T \\ &= \sum_{\tau=1}^{T} \mathbf{z}_{1-\tau:M-\tau}^{(m)} (\boldsymbol{\delta}_{1-\tau+1:M-\tau+1}^{(m)})^T\end{aligned} \tag{12}$$

where the truncated BPTT with T steps back in time is implemented. Therefore, the local gradient for recurrent weights $W^{(mm)}$ at time step τ [14] is yielded by

$$\boldsymbol{\delta}_{t-\tau}^{(m)} = \left((W^{(mm)})^T \boldsymbol{\delta}_{t-\tau+1}^{(m)}\right) * \mathbf{z}_{t-\tau}^{(m)} * \left(\mathbf{1} - \mathbf{z}_{t-\tau}^{(m)}\right). \tag{13}$$

As shown in the red dashed arrows and red circles in Figure 1, the truncated minibatch BPTT does not only consider the error backpropagation for the recurrent weights $W^{(mm)}$ but also the feedforward weights $W^{(m)}$ between recurrent layer and its previous layer. For this consideration, the updating of feedforward weights $W^{(m)}$ is performed by

$$\begin{aligned}\sum_{t=1}^{M} \frac{\partial E_t}{\partial W^{(m)}} &\approx \sum_{\tau=1}^{T} \sum_{t=1}^{M} \mathbf{z}_{t-\tau+1}^{(m-1)} (\boldsymbol{\delta}_{t-\tau+1}^{(m)})^T \\ &= \sum_{\tau=1}^{T} \mathbf{z}_{1-\tau+1:M-\tau+1}^{(m-1)} (\boldsymbol{\delta}_{1-\tau+1:M-\tau+1}^{(m)})^T.\end{aligned} \tag{14}$$

Without loss of generality, we call this algorithm as the backpropagation through time and layer (BPTTL) where the BPTT is not only applied for recurrent layer but also the corresponding feedforward layer. The local gradient for feedfoward weights $W^{(m)}$ at time step τ is calculated by

$$\boldsymbol{\delta}_{t-\tau}^{(m)} = \left((W^{(m+1)})^T \boldsymbol{\delta}_{t-\tau}^{(m)}\right) * \mathbf{z}_{t-\tau}^{(m)} * \left(\mathbf{1} - \mathbf{z}_{t-\tau}^{(m)}\right). \tag{15}$$

4. EXPERIMENTS

4.1. Experimental setup

We evaluated the proposed method for speech recognition by using the corpora of Resource Management (RM) and Aurora4. The training set in RM contained 3990 utterances from 109 speakers with 3.8 hours of speech. The test set had 1460 utterances from 59 speakers. RM had a vocabulary size of 997 words. Aurora4 was extended from the Wall Street Journal (WSJ0) corpus with noise contaminations under different noise types and signal-to-noise ratios. The training set contained 7137 utterances from 83 speakers with 14 hours. The evaluation set had 4620 utterances (330 utterances × 14 test sets) from 8 speakers, with 40.19 minutes of speech data, sampled from the 5K-word closed vocabulary based on the WSJ0 NOV-92 corpus. The 14 test sets were grouped into four conditions: A (clean data), B (noisy data), C (clean data with channel distortion), and D (noisy data with channel distortion). An additional development set in RM and Aurora4 was used to tune $\{\lambda_1, \lambda_2\}$ and learning rate for evaluation of stopping criterion. We followed the shell scripts of RM and Aurora4 from the Kaldi toolkit [15]. The baseline GMM-HMM triphone system was trained by using the feature vector from a context window of seven frames with 13-dimensional MFCC at each frame. LDA was applied to project the concatenated features into 40 dimensions which were further adapted by MLLT. GMM-HMM system was used to find the senone labels of speech frames from the forced-alignment during the initial training of an DNN-HMM system. Decoding was performed by using bigrams for RM and trigrams for Aurora4.

In DNN training, we adopted the feature vector from a context window of 11 frames (± 5 frames). Input layer consisted of 440 units (40 × 11). For RM task, the DNN topology was shaped by 6 hidden layers with 1024 neurons in each hidden layer. The softmax output layer with 1483 units was constructed for the corresponding senones. For Aurora4 task, we used the 40-dimensional FBNK features to train DNN system which automatically learned the dependencies in FBNK features. The DNN topology was formed by 7 hidden layers while each layer had 2048 neurons. The softmax output layer with 2035 units was used. In pre-training phase, we referred the practical guide for RBM training [16] and followed the initial learning rate and momentum setup [17]. DNN system was initialized by stacking the layer-wise RBM based on the CD algorithm with one step of Markov-chain Monte Carlo sampling. After the pre-training, we estimated the initial weights and concatenated the output layers to build the model structure. The weights between last hidden layer and output layer were randomly drawn from $\mathcal{N}(0, 0.01)$. The RNN with five steps back in time ($T = 5$) was adopted.

Methods	WER
GMM	1.93
DNN	1.88
+ Tikhonov	1.78
+ ℓ_2	1.54
+ Tikhonov + ℓ_2	**1.46**
DRRNN-BPTT	1.41
DRRNN-BPTTL	**1.39**

Table 1. Comparison of WERs (%) in RM task.

4.2. Evaluation on model regularization

We evaluate the performance of individual and hybrid regularization in a pre-training procedure of a DNN-HMM system in terms of word error rate (WER) (%) as reported in Table 1 for RM task and in Table 2 for Aurora4 task. The results of GMM-HMM are included. The individual and hybrid regularization parameters $\lambda_1 = 0.0001$ and $\lambda_2 = 0.0002$ were selected for RM task while $\lambda_1 = 0.00005$ and $\lambda_2 = 0.0002$ were chosen for Aurora4 task. We can see that the hybrid Tikhonov and ℓ_2 regularization consistently reduces the WER over the individual regularization in different tasks. This shows the benefit of dealing with model regularization in DNN via joint compensation of the perturbations of training speech and the over-trained parameters due to the ill-posed condition.

Methods	Conditions				Avg.
	A	B	C	D	
GMM	7.29	12.97	12.61	27.66	18.83
DNN	4.33	9.13	11.83	24.69	15.65
+ Tikhonov	3.85	7.94	10.69	22.54	14.10
+ ℓ_2	3.36	7.83	10.54	21.93	13.75
+ Tikhonov + ℓ_2	3.08	7.58	10.14	21.65	**13.47**
DRRNN-BPTT	3.44	7.23	8.80	21.05	12.99
DRRNN-BPTTL	3.29	7.09	8.71	20.83	**12.82**

Table 2. Comparison of WERs (%) in Aurora4 task.

4.3. Evaluation on recurrent neural network

The DNN-HMM with hybrid Tikhonov and ℓ_2 regularization is trained by several learning epochs where the DNN parameters are updated in each epoch by using new alignment based on the decoding procedure using the new posterior probabilities from DNN outputs. Learning epoch is continued until the performance gain is saturated. Given the alignment from the last learning epoch of the regularized DNN-HMM, we further train the RNN according to the BPTT and the proposed BPTTL algorithms. The resulting methods are denoted as the DRRNN with BPTT (DRRNN-BPTT) and the DRRNN-BPTTL. In the implementation using RM and Aurora4, the recurrent weights $W^{(m)}$ were initialized randomly. The recurrent layer was specified as the third hidden layer in RM and the fourth hidden layer in Aurora4. The minibatch size was set to be 256 frames for SGD learning. The initial learning rate was 0.008 and 0.004 for RM and Aurora4, respectively. The sentence-level shuffling was performed in RNN. The WERs are shown in Tables 1 and 2. We find that the system performance is improved by using DRRNN where the regularization issue is tackled and the temporal correlation is captured. The improvement of BPTTL over BPTT is obtained in using RM as well as Aurora4.

5. CONCLUSIONS

This paper investigated the importance of model regularization and temporal correlation in DNN acoustic modeling. We presented the hybrid Tikhonov and weight-decay regularization to achieve the invariance property in speech samples and model parameters during a pre-training procedure of DNN-HMM system. The rate of convergence was increased and the ML learning was attained. The recurrent neural network was implemented to characterize the temporal dependency between speech samples in deep learning. The backpropagation through time and layer was proposed to conduct the weight updating for recurrent weights as well as the corresponding feedforward weights in a minibatch training procedure. Experimental results on RM and Aurora4 showed the consistent improvement of WER by using the proposed DRRNN over the conventional DNN.

6. REFERENCES

[1] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, V. Vanhoucke, P. Nguyen, T. 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, no. 6, pp. 82–97, 2012.

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

[3] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network language model," in *Proceeding of IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 206–211.

[4] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *The Computing Research Repository (CoRR)*, vol. abs/1207.0580, 2012.

[5] L. Wan, M. D. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceding of International Conference on Machine Learning (ICML)*, 2013, vol. 28, pp. 1058–1066.

[6] T. Robinson, M. Hochberg, and S. Renals, "IPA: improved phone modelling with recurrent neural networks," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1994, vol. 1, pp. 37–40.

[7] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2010, pp. 1045–1048.

[8] O. Vinyals, S. Ravuri, and D. Povey, "Revisiting recurrent neural networks for robust ASR," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012, pp. 4085 – 4088.

[9] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, pp. 6645 – 6649.

[10] C. Weng, D. Yu, S. Watanabe, and B.-H. Juang, "Recurrent deep neural networks for robust speech recognition," in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 5569–5573.

[11] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed problems*, W.H. Winston, 1977.

[12] K. Cho, A. Ilin, and T. Raiko, "Tikhonov-type regularization for restricted Boltzmann machines," in *Proceeding of International Conference on Artificial Neural Networks*, 2012, pp. 81–88.

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

[14] T. Mikolov, *Statistical Language Models Based on Neural Networks*, Ph.D. thesis, 2012.

[15] 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 *Proceeding of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[16] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Neural Networks: Tricks of the Trade (2nd ed.)*, pp. 599–619. 2012.

[17] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proceedings of Annual Conference of International Speech Communication Association (INTERSPEECH)*, 2013, pp. 2345–2349.