INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Node pruning based on Entropy of Weights and Node Activity for Small-footprint Acoustic Model based on Deep Neural Networks

Ryu Takeda[1], *Kazuhiro Nakadai*[2], *Kazunori Komatani*[1]

[1]The Institute of Scientific and Industrial Research, Osaka University, Japan
[2]Honda Research Institute Japan, Co. Ltd., Japan

rtakeda@sanken.osaka-u.ac.jp, nakadai@jp.honda-ri.com, komatani@sanken.osaka-u.ac.jp

Abstract

This paper describes a node-pruning method for an acoustic model based on deep neural networks (DNNs). Node pruning is a promising method to reduce the memory usage and computational cost of DNNs. A score function is defined to measure the importance of each node, and less important nodes are pruned. The entropy of the activity of each node has been used as a score function to find nodes with outputs that do not change at all. We introduce entropy of weights of each node to consider the number of weights and their patterns of each node. Because the number of weights and the patterns differ at each layer, the importance of the node should also be measured using the related weights of the target node. We then propose a score function that integrates the entropy of weights and node activity, which will prune less important nodes more efficiently. Experimental results showed that the proposed pruning method successfully reduced the number of parameters by about 6% without any accuracy loss compared with a score function based only on the entropy of node activity.

Index Terms: speech recognition, deep neural networks, node pruning

1. Introduction

Deep neural networks (DNNs) are widely used as acoustic models in automatic speech recognition (ASR) instead of Gaussian mixture models (GMMs) because of their high word accuracy (WA) [1, 2, 3, 4]. The fully-connected networks are used as the classification layer in many neural network approaches in the ASR area. Because such fully-connected networks have many parameters, the computational cost and memory usage are high, restricting its applicable area, such as in small/micro computers or embedded systems with ordinary CPUs. Thus, small-footprint DNNs in terms of memory usage and computational cost are required.

Node pruning is a promising method to achieve small-footprint DNNs compared with other approaches in speech, text and image processing. Size-reduction methods are mainly based on 1) low-rank matrix factorization using singular value decomposition (SVD) [5] or decomposition methods [6], 2) node pruning [7, 8, 9], 3) a special network structure [10], 4) constraint in training [11, 12, 13, 14], and 5) combinations of node-pruning and quantization [15]. In particular, the last study [15] achieved a high compression rate of large convolutional NNs (CNNs) and DNNs for the MNIST and ImageNet data sets by combining the node-pruning and quantization methods. They showed an important fact that the SVD reduction method did not work as efficiently as quantization and node pruning. We also confirmed the efficiency of the combination method in ASR task [16]. Thus, node pruning is definitely one of the key techniques to achieve small-footprint DNNs.

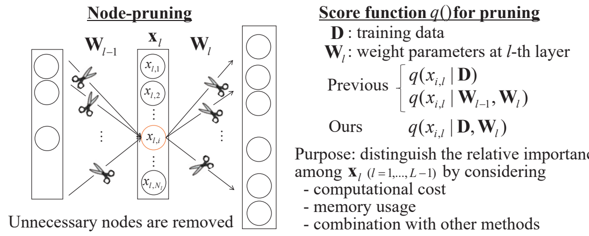

Figure 1: *Our problem and approach*

The key point of the node-pruning approach is the design of the score function of each node for pruning, as shown in Fig. 1. A score function q is usually defined to measure the importance of each node, and nodes with low importance are removed from networks. Here, D represents the data set for training, and $\mathbf{W}_l$ represents the weight parameters at the l-th layer. After sorting nodes by their scores, we remove nodes until the number of nodes reaches the pre-defined pruning ratio. Two approaches can be used to design the score function: 1) one is based on weight parameters [7], and 2) the other is based on node-activity [17, 18, 7]. The former uses weight-norm of each node to calculate the importance of the node. The latter uses the statistics of node activity during training to calculate the importance of the node, such as the mean, variance, and entropy.

The inconvenience of conventional node-pruning is the inefficient pruning of nodes caused by the score function based on either weight parameters or node activity. For example, the nodes in the last hidden layer should be removed with high priority because the layer has much more parameters than others and thus they can be removed in many cases. Therefore, the importance of nodes is affected by computational cost and memory usage at each layer. The score function based on weight parameters can consider such priority among nodes with the same node activity. *Note that the score based on node activity must be prior to that on weight parameters.* For example, we should prune a node of which activity level is low even though its weight score may indicate it is important. This is because there are many redundant weight parameters of DNNs and the score function using weight parameters does not necessarily represent the importance of nodes.

We propose using both the entropy of nodes and the entropy of its weights for the score function by considering their priority. Our contributions are 1) exploiting the weight-entropy score and 2) the experimental validation of our concept that uses both weight and node-activity information. First, the entropy of weights is defined by calculating the distribution of weights connected at each node. Because this entropy also depends on the number of weights, we can consider the difference in the number of weights among each layer. Next, we designed a score function that uses the entropy of weights and node. This score

http://dx.doi.org/10.21437/Interspeech.2017-779

function defines the priority for node pruning based on both of the weight entropy and the node entropy to reduce the risk of mis-removal of necessary nodes.

2. Node-pruning of Deep Neural Networks based on Node-entropy

2.1. Acoustic Model of Deep Neural Networks

The structure of continuous NNs is defined recursively on the layer index l. First, the input vector $\mathbf{x}_l = [x_{l,1}, ..., x_{l,N_l}]^T \in \mathbb{R}^{N_l}$ is affine transformed, then activation functions $\mathbf{h}_l : \mathbb{R}^{M_l} \to \mathbb{R}^{N_{l+1}}$ are applied. Here, $\cdot^T$ denotes the transportation operator, N_l and M_l are dimensions of $\mathbf{x}_l$ and temporal vector $\mathbf{z}_l = [z_{l,1}, ..., z_{l,M_l}]^T$ at the l-th layer, respectively. Therefore, the output of the L-th layer can be recursively described for $l = 0, ..., L-1$ given the initial input vector $\mathbf{x}_0$.

$$\mathbf{z}_l = \mathbf{W}_l \mathbf{x}_l + \mathbf{b}_l, \;\; \mathbf{x}_{l+1} = \mathbf{h}_l(\mathbf{z}_l) \tag{1}$$

where the matrix $\mathbf{W}_l = (w_{l,ij}) \in \mathbb{R}^{M_l \times N_l}$ and vector $\mathbf{b}_l = [b_{l,1}, ..., b_{l,M_l}]^T \in \mathbb{R}^{M_l}$ are the weight and bias parameters at the l-th layer, respectively. The sigmoid function and soft-max function are often used as an activation function $\mathbf{h}_l$. In ASR, the input vector $\mathbf{x}_0$ corresponds to temporal speech features, and the final output vector $\mathbf{x}_L$ is used for the acoustic likelihood of Hidden Markov Model (HMM) states [3].

2.2. Node-pruning based on Activity of Node

Node entropy is used as the criterion of node-pruning proposed by [7]. This criterion can find a node of which outputs are almost identical on all training data. Such a node can be removed or merged with the corresponding bias parameter [17]. Node-entropy q of the i-th node at the l-th layer, $x_{l,i}$, is defined using the following equation

$$q(l,i|D) = -\frac{N_0}{N_{0+1}} \log \frac{N_0}{N_{0+1}} - \frac{N_1}{N_{0+1}} \log \frac{N_1}{N_{0+1}} \tag{2}$$

where D is a data set, N_0 is the number of the sigmoid-output values ranging from 0 to 0.5, and N_1 is the number of values ranging from 0.5 to 1.0. The $N_{0,1} = N_0 + N_1$ equals the number of samples in the data set. All nodes are sorted using this entropy criterion and nodes with low entropy are removed. Note that the output-weight-norm-based method used in the study by [7] did not work well due to the different variance of weights in each layer.

2.3. Problem

The inconvenience of conventional node-pruning is the inefficient pruning of nodes caused by the score function based on either weight parameters or node activity. The score function based on weight parameters can consider the computational-cost priority among nodes with the same node-activity. However, the score based on node activity must be considered prior to that based on weight parameters. For example, we should prune a node of which activity level is low even though its weight score may indicate it is important. This is because there are many redundant weight parameters of DNNs and the score function using weight parameters does not necessarily represent the importance of nodes. Note that the importance of nodes will also be affected by the combination of other methods, such as weight parameter quantization [19, 15], because the computational cost and the memory usage of DNNs will also change.

3. Exploiting Weight Entropy for Node-pruning Score Function

First, we define the entropy of weights at each node to consider the number of weight parameters for node pruning. Next, we design the score function depending on the node entropy and weight entropy.

3.1. Entropy of Weight Parameters

We used the entropy function of weights because it can represent the influence of the number of weight parameters. We denote the i-th column weights of $\mathbf{W}_l$ in the l-th layer as $\mathbf{w}_{l,i}$. This $\mathbf{w}_{l,i}$ is a weight vector connected to the node $x_{l,i}$. We defined the entropy of weights, $q(l,i|\mathbf{w}_{l,i})$, of the node $x_{l,i}$ as

$$\begin{aligned} q(l,i|\mathbf{w}_{l,i}) \; &:= \; -\textstyle\sum_{\mathbf{w}_{l,i}} p(\mathbf{w}_{l,i}) \log(p(\mathbf{w}_{l,i})) &(3)\\ &= \; -N_{l+1} \textstyle\sum_{w} p(w|\pi_{l,i}) \log(p(w|\pi_{l,i})). &(4)\end{aligned}$$

Here, we assumed that 1) weights are discretized into n-bit to calculate the entropy for discrete variables, 2) each weight is independent and then the joint probability $p(\mathbf{w}_{l,i})$ becomes the product of marginalized probabilities and 3) the marginalized probabilities are I.I.D and each $p(w|\pi_{l,i})$ is a categorical distribution with the common parameters $\pi_{l,i} = [\pi_{l,i,1}, ..., \pi_{l,i,2^n}]^T$ ($\sum_j \pi_{l,i,j} = 1$). Note that the high entropy of weight means that the number of weights is large and that the weights take many patterns. The node with high entropy of weight should not be removed because of its complexity.

The parameters $\pi_{l,i}$ are calculated as the follows:

1. quantize weights with n-bit, and they are clustered into 2^n patterns.
2. calculate the number of weights $N_{l,i,j}$ assigned to each cluster j (histogram).
3. estimate $\pi_{l,i,j} = N_{l,i,j} / \sum_j N_{l,i,j}$ (normalization).

The reason we discretized weights is that we do not know the true probability density function of weight parameters.

We determine the actual distribution of entropies to understand the effect of weight entropy. Figure 2 shows the distribution of actual weights- and node-entropy of nodes at each layer ($l = 1, ..., 6$). The configuration of DNNs is shown in the experimental section of this paper. The horizontal axis denotes the node entropy, and the left side of the axis means low entropy (low importance). The vertical axis denotes the weight-entropy, and the bottom of the axis also means low entropy (low importance). Note that the scale of the vertical axis of the layer $l = 6$ is different from others due to the different number of weights.

We can see that weight entropy can distinguish the importance of the node even if the node entropy is the same. The tendency of the distribution differs in each layer, and the distribution of node entropy in the latter layer almost becomes bipolar. The distribution of the first layer $l = 1$ is ambiguous, indicating that the nodes of the first layer $\mathbf{x}_1$ should not be removed.

3.2. Design of Score Function

The purpose is to design the score function for node pruning that considers the impact of weight information and node activity. The score function used for node pruning is designed using node-entropy $e_n = q(l,i|D)$ and weight-entropy $e_w = q(l,i|\mathbf{w}_{l,i})$. The ideal score function is illustrated in Figure 3. The two-dimensional plane consists of the entropy of node

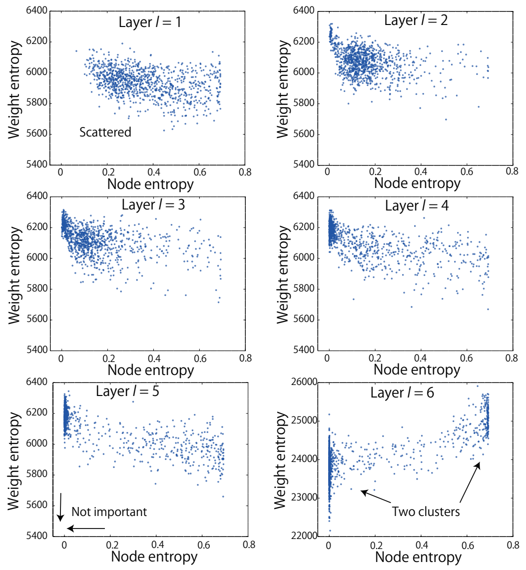

Figure 2: *Distribution of node entropy and weight entropy*

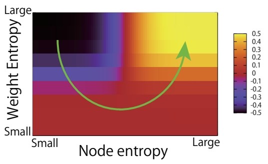

Figure 3: *Ideal cost*

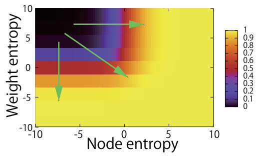

Figure 4: *Proposed cost*

e_n and weight e_w. The arrow denotes the ideal order of node-pruning, and the score function takes a higher value according to the arrow. The pruning priority of the top-left region is higher than that of others because the impact on memory usage and the number of parameters becomes stronger. The priority of the top-right region is lower than others because the risk to prune meaningful nodes and weights is higher.

We propose the following sigmoid-based score function of which the range is $[0, 1.0]$ for the final score of the node $x_{l,i}$.

$$q(l, i|\mathbf{w}_{l,i}, \mathbf{D}) = \left(\frac{1}{1+\exp(-\frac{e_n - m_n}{v_n})} - 1.0\right) \left(\frac{1}{1+\exp(-\frac{n_w - m_w}{v_w})}\right) + 1.0 \quad (5)$$

where m_n and n_w are bias parameters, and v_n and v_w are scaling parameters of node entropy and weight entropy. The plot of the function is illustrated in Figure 4. The score monotonously increases out from the top-left region. We used the mean and variance of node entropy and weight entropy for these parameters: $m_n = \mathrm{E}[e_n]$, $v_n = \sqrt{\mathrm{E}[(e_n - m_n)^2]}$, etc. The reasons we designed this cost are that 1) the ideal cost function is complex to define its shape and to adjust parameters for slopes, and 2) a comparison with node-entropy-based node pruning is easy because the edge of the weight-entropy axis matches the node-entropy cost. Although this score function is empirical, it is enough to show the concept and the usefulness of the score function based on node-activity and weight information as a first step. The investigation of the better score function is future work.

Table 1: *Configuration*

Item	Value
Features for DNN	FBANK 825 dim. [(25+ Δ 25 + $\Delta\Delta$ 25) $\times$ 11 frames]
# of DNN layer (L)	7
DNN network size	input layer ($l = 0$): 1024 $\times$ 825 middle layer ($l = 1, ..., 5$): 1024 $\times$ 1024 output layer ($l = 6$): 4000 $\times$ 1024
Training set	clean speech 223 hours (799 males and 168 females)
Test set	clean speech 3.5 hours (15 males and 5 females)

Table 2: *Computer specifications*

OS	Ubuntu 14.04.2 LTS
CPU	Intel Core i5 3.50GHz
Memory	32GB
Cache	6M

Table 3: *Quantization bits for RTF (quan.)*

Weights	bits
$\mathbf{W}_l (l = 0, ..., 5)$	2
$\mathbf{W}_6$	8

4. Experiments

4.1. Experimental Setup

We evaluated our node-pruning method by conducting large-vocabulary continuous-speech recognition experiments using the Corpus of Spontaneous Japanese (CSJ), which is a collection of Japanese lecture recordings [20]. The word accuracies (WAs), the real-time factor (RTF) of forward calculation of NNs and the number of weight parameters after using node-entropy-based pruning [7] and our proposed method were investigated. The RTF is defined as RTF $=$ (processing time)/(data duration). We tested two kinds of forward calculation of NNs; the naive computation without any special instruction sets (RTF), and the computation with weight parameter quantization and took-up-table technique for multiply-add operation used in [16] (RTF (quan.)). The bits for quantization at each layer are shown in Table 3. We showed the *RTF (quan.)* as *reference* to show the impact of node-pruning with a different computation scheme. Note that the actual WAs of RTF (quan.) will change because of the influence of parameter quantization.

The training data for the acoustic model of DNNs contained 223 hours of *Academic-Presentation-Speech* recordings The evaluation data for clean speech were test sets 1 and 2 of CSJ, i.e., 3.5 hours of lectures featuring 20 speakers (15 males and 5 females). The training data for the language model contained all transcriptions in the CSJ, except the evaluation data. We used a tri-gram language model with 65,000 words. Julius (ver. 4.3.1) was used for two-pass decoding [21], and the language model weight and insertion penalty were set to their default values, 8 and -2, respectively.

We first trained a GMM-HMM with a tri-phone, 4000 tied-states, and 32 Gaussian mixtures by using the Hidden Markov Model Toolkit (HTK) [1]. The 13 Mel-frequency cepstral coefficients (MFCCs) and delta and delta-delta coefficients with mean and variance normalization per utterance were used as speech features. The features were extracted at every 10-ms interval from the speech signal, the sampling rate of which was 16 kHz. The window size for short-time Fourier transformation (STFT) was 25 ms. The DNN-HMM used the same HMM with a GMM-based model and $L = 7$ layers with 1024 hidden nodes. The number of nodes at hidden layers is moderate [22]. The output dimensions were 4000 to classify the tied-states of the

[1] http://htk.eng.cam.ac.uk/

Table 4: *Performance of pruning based on node-entropy: WA, the number of parameters and nodes in each layer*

Pruned nodes (%)	Word Accuracy (%) w/o retrain	w/ retrain	# of params (mega)	# of nodes at the l-th layer 0	1	2	3	4	5	6	RTF	ref: RTF (quan.)
0	81.86	–	9.71	825	1024	1024	1024	1024	1024	1024	0.841	–
30	82.04	82.02	4.98	825	1024	976	912	594	410	385	0.428	0.250
40	77.84	81.72	4.00	825	1023	868	698	441	335	322	0.344	0.215
50	61.88	80.74	3.17	825	1017	606	456	377	313	303	0.273	0.195
60	20.33	79.34	2.48	825	945	335	287	308	298	285	0.213	0.173
70	0.01	76.75	1.90	825	731	165	171	230	281	265	0.164	0.146

Table 5: *Performance of proposed pruning: WA, the number of parameters and nodes in each layer*

Pruned nodes (%)	Word Accuracy (%) w/o retrain	w/ retrain	# of params (mega)	# of nodes at the l-th layer 0	1	2	3	4	5	6	RTF	ref: RTF (quan.)
30	81.72	81.89	4.53	825	1024	982	936	653	468	**238**	**0.390**	**0.204**
40	**78.57**	81.58	3.74	825	1024	899	730	454	345	**234**	**0.326**	**0.188**
50	**64.68**	80.68	2.94	825	1019	650	475	389	314	223	0.254	0.171
60	19.61	79.29	2.28	825	968	359	299	313	299	220	0.197	0.154
70	0.06	76.78	1.72	825	757	176	181	235	283	211	0.149	0.130

HMM. A total of 825 dimensions of features existed for DNN input, including 11 frames (the current, the previous 5 frames, the post 5 frames) of the basic features. The basic features consisted of 25 log filter bank coefficients and the delta and delta-delta coefficients. The mean and variance normalization were applied to the features. Some configurations are summarized in Table 1, and they were the same used in [16]. The computer specifications are also listed in Table 2.

Next, the DNNs were trained with Viterbi force alignment state-labels by using the GMM-HMM. We used a discriminative pre-training method [3] for the pre-training and used the AdaGrad method [23] for scheduling learning-rate parameters. The mini-batch size was 64 as [24] mentioned. We retrained the DNNs after applying node-pruning to improve the performance of both methods. The number of quantization bits for weight entropy calculation was 10.

4.2. Results

Tables 4 and 5 show the word accuracy, the total number of parameters, and the number of nodes at each layer l of the baseline method based on node entropy and our method under different pruning ratios of nodes. The WA includes the results with and without re-training of DNNs because the WA without re-training indicates the quality of node pruning itself.

Our node-pruning slightly outperformed the baseline method in terms of WA without re-training. The WA without re-training was improved by about 3 points with a 50 % pruning ratio. The difference between with and without re-training was almost zero after re-training.

The total number of weight parameters of our method was less than that of the baseline method by about 6–10% in the case of 30–50% ratio. This is because our method pruned nodes of the 6-th layer more than those of other layers even under the same pruning ratio. The number of weights at the 6-th layer, $\mathbf{W}_6$, was much larger than those of others due to the large output nodes of DNNs. Thus, the proposed score function could consider the number of weights of nodes in pruning. The 6–10% improvement seems to be a slight gain; however, the RTFs of our method were also less than those of baseline method especially in the case of combining node pruning and weight parameter quantization techniques. The relative improvement of RTFs (quan.) is about 10–18%.

4.3. Discussion

We showed that weight entropy is useful for node pruning in terms of reducing the number of parameters. Further improvements can be achieved by 1) improving the score function and 2) integrating node pruning with the other method.

The score function and its parameter optimization should be investigated. Because our settings in this study were not optimized, optimizing the parameters will further improve the performance. The difficulty is determining how to define the optimum parameters. The theoretical design of the score function, such as Bayesian modeling, is also required to achieve more flexible and efficient node pruning.

Integration with other methods, such as quantization, should be investigated to achieve smaller footprint DNNs. We previously showed that the combination of quantization and node pruning is also efficient for the ASR task. In such case, the number of bits for quantization also differs in each layer. Therefore, a pruning method that can consider the bits for quantization will further improve the performance. Defining the importance of nodes will become difficult.

We emphasize that our work will contribute to the future research that combines node pruning with quantization methods because the weight entropy could be used to evaluate each node at bit-level importance of weights. When we use quantization methods, we must consider the number of bits for quantization. Actually, the number of these bits for quantization also affects the total memory usage and computational cost [16]. Such a detailed evaluation of each node will be required to generate higher-level small-footprint DNNs.

5. Conclusions

Our goal was the developing small-footprint deep neural networks (NNs) with reduced memory and computational cost for acoustic models. The node-pruning approach is a promising method to achieve small-footprint DNNs. We improved the node-pruning method based on node entropy by considering the weight information. We proposed 1) exploiting weight entropy, 2) designing the score function using both weight and node entropy. Experiments showed that our node-pruning outperformed the node-entropy-based method by 6% in node reduction rate without any accuracy loss. Our future work is to design an efficient score function for pruning nodes.

6. 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 Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 82–97, 2012.

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

[3] F. Seide, G. Li, , and X. D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transaction," in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011, pp. 24–29.

[4] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using contex-dependent deep neural network," in *Proceedings of Interspeech*, 2011, pp. 437–440.

[5] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition." in *Proceedings of Interspeech*, 2013, pp. 2365–2369.

[6] 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 *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6655–6659.

[7] T. He, Y. Fan, Y. Qian, T. Tan, and K. Yu, "Reshaping deep neural network for fast decoding by node-pruning," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 245–249.

[8] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015.

[9] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," in *arXiv preprint arXiv:1606.09274*, 2016.

[10] V. Sindhwani, T. Sainath, and S. Kumar, "Structured transforms for small-footprint deep learning," in *Advances in Neural Information Processing Systems*, 2015.

[11] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[12] J. Li, R. Zhao, J.-T. Huang, and Y. Gong, "Learning small-size DNN with output-distribution-based criteria." in *Proceedings of Interspeech*, 2014, pp. 1910–1914.

[13] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *arXiv preprint arXiv:1606.07947*, 2016.

[14] T. Ochiai, S. Matsuda, H. Watanabe, and K. Shigeru, "Automatic node selection for deep neural networks using group lasso regularization," in *Procceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017, pp. 5485–5488.

[15] S. Han, H. Mao, and W.J Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[16] R. Takeda, K. Nakadai, and K. Komatani, "Acoustic model training based on node-wise weight boundary model for fast and small-footprint deep neural networks," *Computer Speech & Language (in press)*, 2017.

[17] J. Sietsma and R. JF Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991.

[18] S. Yamamoto, T. Oshino, T. Mori, A. Hashizume, and J. Motoike, "Gradual reduction of hidden units in the back propagation algorithm, and its application to blood cell classification," in *Proceedings of International Joint Conference on Neural Networks*, vol. 3. IEEE, 1993, pp. 2085–2088.

[19] V. Vanhouche, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proceedings of Deep Learning and Unsupervised Feature Learning NIPS Workshop*, vol. 1, 2011.

[20] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *Inproceedings of ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[21] A. Lee and T. Kawahara, "Recent development of open-source speech recognition engine Julius," in *Proceedings of Asia-Pacific Signal and Information Processing Association, Annual Summit and Conference*, 2009, pp. 131–137.

[22] T. Moriya, T. Tanaka, T. Shinozaki, S. Watanabe, and K. Duh, "Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy," in *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2015, pp. 610–616.

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

[24] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.