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

Ryu Takeda¹, Kazuhiro Nakadai², Kazunori Komatani¹

¹The Institute of Scientific and Industrial Research, Osaka University, Japan
²Honda 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 \( W_l \) 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
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 \( x_0 = [x_{1,1}, \ldots, x_{1,N_t}]^T \in \mathbb{R}^{N_t} \) is affine transformed, then activation functions \( b_l : \mathbb{R}^{M_l} \rightarrow \mathbb{R}^{N_{l+1}} \) are applied. Here, \( T \) denotes the transportation operator, \( N_t \) and \( M_l \) are dimensions of \( x_0 \) and temporal vector \( z_l = [z_{l,1}, \ldots, z_{l,M_l}]^T \) at the \( l \)-th layer, respectively. Therefore, the output of the \( L \)-th layer can be recursively described as \( l = 0, \ldots, L - 1 \) given the initial input vector \( x_0 \).

\[
z_l = W_l x_l + b_l, \quad x_{l+1} = h_l(z_l)
\]

where the matrix \( W_l \in [\mathbb{R}^{M_l \times N_t}] \) and vector \( b_l \in [\mathbb{R}^{M_l}] \) 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 \( h_l \). In ASR, the input vector \( x_0 \) corresponds to temporal speech features, and the final output vector \( x_L \) 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, \( q_{l,i} \), is defined using the following equation

\[
q(l, i | D) = N_0 \frac{N_0}{N_{l+1}} \log \frac{N_0}{N_{l+1}} - N_1 \frac{N_1}{N_{l+1}} \log \frac{N_1}{N_{l+1}}
\]

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 \( N_{l+1} = N_0 + N_1 \) 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 \( W_l \) in the \( l \)-th layer as \( w_{l,i} \). This \( w_{l,i} \) is a weight vector connected to the node \( x_{l,i} \). We defined the entropy of weights, \( q(l, i | w_{l,i}) \), of the node \( x_{l,i} \) as

\[
q(l, i | w_{l,i}) := - \sum_{w_l} p(w_l | i) \log(p(w_l | i))
\]

where \( \sum_{w_l} p(w_l | i) \log(p(w_l | i)) \) 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 \( N_{l+1} = N_0 + N_1 \) 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.

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 \( l = 1, \ldots, 6 \). 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 \( x_1 \) 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 \( e_w = q(l, i | w_{l,i}) \). The ideal score function is illustrated in Figure 3. The two-dimensional plane consists of the entropy of node
function based on node-activity and weight information as a agreement.

Although this score function is empirical, it easy because the edge of the weight-entropy axis matches the shape and 2) a comparison with node-entropy-based node pruning is complex to define its shape and to adjust parameters for slopes, and 3) a comparison with node-entropy-based node pruning is challenging to define its shape and to adjust parameters for slopes. The plot of the function is illustrated in Figure 4. The score monotonously decreases as the parameter values increase, with a value of 1 in the top-right region. The priority of the top-left region is higher for node pruning, and the score function takes a higher value according to the arrow. The pruning priority of the top-left region is higher because the risk to prune meaningful nodes and weights is higher.

We propose the following sigmoid-based score function of which the range is \([0, 1]\) for the final score of the node \(x_{1,i}\).

\[
q(l, i|w_{1,i}, D) = \left( \frac{1}{1 + \exp\left(-\frac{m_n - v_n}{v_w}\right)} - 1.0 \right) + 1.0
\]

where \(m_n\) and \(v_n\) are bias parameters, and \(v_w\) 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: \(m_n = \text{E}[e_n]\), \(v_n = \sqrt{\text{E}[(e_n - m_n)^2]}\), 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.

## 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

![Figure 2: Distribution of node entropy and weight entropy](image)

![Figure 3: Ideal cost](image)

![Figure 4: Proposed cost](image)

---

**Table 1: Configuration**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features for DNN</td>
<td>FBANK 825 dim. ([25+\Delta 25+\Delta\Delta 25] \times 11) frames</td>
</tr>
<tr>
<td># of DNN layer ((L))</td>
<td>7</td>
</tr>
<tr>
<td>DNN network size</td>
<td>(\text{input layer} (l = 0): 1024 \times 825)</td>
</tr>
<tr>
<td></td>
<td>(\text{middle layer} (l = 1, \ldots, 5): 1024 \times 1024)</td>
</tr>
<tr>
<td></td>
<td>(\text{output layer} (l = 6): 4000 \times 1024)</td>
</tr>
<tr>
<td>Training set</td>
<td>clean speech 223 hours</td>
</tr>
<tr>
<td></td>
<td>(799 males and 168 females)</td>
</tr>
<tr>
<td>Test set</td>
<td>clean speech 3.5 hours</td>
</tr>
<tr>
<td></td>
<td>(15 males and 5 females)</td>
</tr>
</tbody>
</table>

**Table 2: Computer specifications**

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Ubuntu 14.04.2 LTS</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel Core i5 3.50GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>32GB</td>
</tr>
<tr>
<td>Cache</td>
<td>6M</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights bits</td>
<td>(W_l(l = 0, \ldots, 5)) 2</td>
</tr>
<tr>
<td></td>
<td>(W_6) 8</td>
</tr>
</tbody>
</table>
The mini-batch size was set to 64 as [24] mentioned. We retrained the DNNs after applying node-pruning to improve the performance. The total number of weight parameters of our method was about 10–18% larger than those of the baseline method due to the large output nodes of DNNs. Thus, the proposed score function 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.

### 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 node-entropy 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 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 baseline 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


