Boundary Contraction Training for Acoustic Models based on Discrete Deep Neural Networks

Ryu Takeda, Naoyuki Kanda, and Nobuo Nukaga

Central Research Laboratory, Hitachi Ltd., 1-280, Kokubunji-shi, Tokyo, Japan
{ryu.takeda.qh, naoyuki.kanda.kn, nobuo.nukaga.br}@hitachi.com

Abstract

The boundary contraction training for acoustic models based on deep neural networks in a discrete system (discrete DNNs) is presented in this paper. Representing the parameters of DNNs with small bits (such as 4 bits) can reduce not only memory usage but also computational complexity by utilizing a CPU cache and look-up tables efficiently. However, simply quantizing parameters of the normal continuous DNNs degrades the recognition accuracy seriously. We tackle this problem by developing a specialized training algorithm for discrete DNNs. In our algorithm, continuous DNNs with boundary constraint are first trained, and the trained parameters are then quantized to meet the representation of discrete DNNs. When training continuous DNNs, we introduce the boundary contraction mapping to shrink the distribution of parameters for reducing the quantization error. In our experiments with 4-bit discrete DNNs, while simply quantizing normally trained DNNs degrades the word accuracy by more than 50 points, our method can maintain the high word accuracy of DNNs with only two points degradation.

Index Terms: speech recognition, deep neural network, boundary contraction, quantization

1. Introduction

Automatic speech recognition (ASR) is widely used in many systems, such as in dictation, translation, car navigation, and smartphones. The speech interface in these systems is convenient for us because voice reflects natural actions and thus users are free from having to perform annoying operations, such as pushing some buttons or writing characters. On the other hands, the accuracy, the speed of response and the portability are required for practical ASR systems because we usually expect the speech interface to work at any systems and to return the correctly recognized text as soon as possible.

The accuracy of ASR is dramatically improved by using an ASR system that is based on deep neural networks (DNNs) compared to that based on Gaussian Mixture Model (GMM) [1, 2, 3, 4, 5, 6]. The DNNs are one of the neural networks containing many layers and nodes that improve its classification performance beyond that of the GMM. On the other hand, the response time of DNNs-based ASR without any additional techniques tends to be long because of the high computational cost of DNNs. Implementations using a graphics processing unit (GPU) [7] or distributed computing [8] are effective for speed-up of DNNs. However, such an expensive approach cannot be applied to restricted resource systems, such as embedded systems with a CPU.

The interesting approaches to run DNNs-based ASR on a standard computer architecture are mainly based on the sparseness of parameters [9], a special instruction set of a CPU [10], or a multi-frame prediction [11]. The first approach skips the multiply-and-add calculations related to the “zero” parameters of DNNs by increasing the sparseness of parameters during training. The second one utilizes a special CPU instruction set, e.g., Supplemental Streaming SIMD Extensions 3 (SSSE3) that performs parallel multiply-and-add operations by quantizing parameters to 8 bits. The last one adopts the network model for predicting not only the passing time but also future time. Although these researches contribute to a real-time recognition on a CPU, they do not utilize the computer architecture enough to realize a more flexible DNNs-based ASR.

We focus on the DNNs in a discrete system, i.e. “discrete DNNs”, that is inspired by quantized DNNs to achieve an efficient implementation on resource restricted computers (Fig. 1). Parameters represented by small bits will reduce the amount of memory and computational complexity by utilizing a CPU cache and a look-up table. For example, the number of inner product patterns of 3D vectors with a 4-bit integer can be expressed by using the look-up table of 24-bit patterns. However, simply quantizing parameters of the normal continuous DNNs degrades the recognition accuracy seriously.

We solve this problem by developing a specialized training algorithm for discrete DNNs. In our algorithm, continuous DNNs with boundary constraint are first trained, and the trained parameters are then quantized to meet the representation of discrete DNNs. This enables us to avoid directly solving the combinational optimization problem of discrete DNNs parameters. When training continuous DNNs, we introduce the boundary contraction mapping to shrink the distribution of parameters for reducing the quantization error. In Section 2, we give an overview of DNNs in ASR, and discuss the boundary contraction in Section 3. Our experiments and their results are presented in Sections 4 and 5, and then we conclude the paper.

2. Deep Neural Networks in ASR

2.1. Acoustic Model in Speech Recognition

Speech recognition is the estimation of a sentence \( \hat{W} \) given the speech feature vectors sequence \( \mathbf{O} = [o_1, \ldots, o_T] \) with a length of \( T \). The following maximization problem is formulated by using a Bayesian framework, Hidden Markov Model (HMM) and Viterbi approximation [12]:

\[
\hat{W} = \arg \max_W p(W) \max_s \prod_t p(o_t | s_t) p(s_t | s_{t-1}) \quad (1)
\]
where \( s_t \) is the state of the HMM at index \( t \), \( s = [s_1, ..., s_T] \) is the latent states sequence of the HMM, and \( p(o_t | s_t) \) represents the acoustic model. \( p(s_t | s_{t-1}) \) is the state transition probability of the HMM and \( p(W) \) represents the language model.

The acoustic model \( p(o_t | s_t) \) can be rewritten as follows by using its posterior probability:

\[
p(o_t | s_t) \propto \frac{p(s_t | o_t)}{p(s_t)}. \tag{2}
\]

Here, the prior, \( p(s_t) \), of the states is estimated by using training data. The DNNs model this posterior probability \( p(s_t | o_t) \) of a state given an observation vector \( o_t \) [3].

### 2.2. Model and Parameter Training of DNNs

The structure of the neural networks is defined recursively on the layer index \( l \). At first, the input vector \( x_l \in \mathbb{R}^{N_l} \) is affine transformed, and then arbitrary functions \( h_l : \mathbb{R}^{M_l} \rightarrow \mathbb{R}^{N_{l+1}} \) are applied. Therefore, the output of the \( L \)-th layer can be recursively described for \( l = 0, ..., L - 1 \) given the initial input vector \( x_0 = [x_{0,1}, ..., x_{0,0}]^T \).

\[
z_l = W_l x_l + b_l, \tag{3}
\]

\[
x_{l+1} = h_l(z_l), \tag{4}
\]

where the matrix \( W_l \in \mathbb{R}^{M_l \times N_l} \) and the vector \( b_l = [b_{l,1}, ..., b_{l,M_l}]^T \in \mathbb{R}^{M_l} \) are the weight and the bias parameters at \( l \)-th layer, respectively. The dimension of the vector \( x_l \) represents the number of nodes at \( l \)-th layer in this paper. The sigmoid function and the soft-max function are often used as an activation function \( h_l \) of the middle and the last layer. Therefore, the outputs can be considered the posterior probability, \( p(s_t | o_t) \), in Eq. (2).

The parameters are usually estimated by using a supervised training that is based on back propagation [13]. Given the cost function \( E \) and supervisory signal vector \( r = [r_1, ..., r_N] \in \mathbb{R}^N \), the update rules can be derived by using the recursive structure and chain-rules as

\[
\text{Initial errors} \quad \epsilon_L = \left( \frac{\partial E}{\partial x_L} (r, x_L) \right)^T, \tag{5}
\]

for \( l = L - 1, ..., 0 \)

\[
\delta_l = \left( \frac{\partial h_l}{\partial z_l} (z_l) \right)^T \epsilon_{l+1}, \quad \epsilon_l = W_l^T \delta_l, \tag{6}
\]

\[
W_l \leftarrow W_l - \eta \delta_l x_l^T, \tag{7}
\]

\[
b_l \leftarrow b_l - \eta \epsilon_l, \tag{9}
\]

where \( \eta \) is the learning-rate parameter. The cross entropy function is used as the cost function \( E \) [3].

### 2.3. Quantized DNNs and Efficient Implementation

The DNNs on CPUs are built by quantizing the weights \( W_l \) into several integer values [10]. For example, the inner product of the matrix in the \( l \)-th layers can be described as

\[
z_{l,t} = \alpha_l \beta_l \sum_k w_{l,ik} x_{l,k} + b_{l,t} \approx \gamma_l \sum_k \tilde{w}_{l,ik} \tilde{x}_{l,k} + b_{l,t}, \tag{10}
\]

where \( \alpha_l, \beta_l \) and \( \gamma_l \) are the scaling factors of the weights, input, and unification of them, respectively. The variable \( \tilde{w}_{l,ik} \) is the \( i \)-th row and the \( k \)-th column element of \( W_l \) in the \( l \)-th layer. The scaled values \( \frac{w_{l,ik}}{\alpha_l} \) and \( \frac{x_{l,k}}{\beta_l} \) are approximated by integers, \( \tilde{w}_{l,ik} \) and \( \tilde{x}_{l,k} \).

The quantized DNNs enable us to achieve efficient implementations of DNNs on CPU machines. For example, the SSSE3 instruction set of an Intel CPU performs the parallel multiply-and-add operations of eight variables related to \( \tilde{w}_{l,ik} \) and \( \tilde{x}_{l,k} \) [10]. Few-bit representation of parameters also enables us to utilize a look-up table of multiply-and-add operations. Moreover, the compression of parameters also reduces the amount of memory and utilizes a CPU cache for accelerating processing speed.

### 2.4. Problem and Approach

The accuracy of quantized DNNs is obviously affected by the quantized parameters \( \tilde{w}_{l,ik} \) and \( \tilde{x}_{l,k} \). Although 8-bit quantization does not degrade the word accuracy [10], smaller quantization that contributes to efficient implementations of DNNs will degrade the accuracy seriously due to its quantization error. Since solving the problem without any clues is difficult, we review and understand the quantized DNNs more, and define a new DNNs problem in a different system to find some hits for modeling.

### 3. Boundary Contraction Training

#### 3.1. Bounded Discrete System and Our Strategy

We define a discrete DNNs as the DNNs whose parameters excepting scaling parameters are a discrete set (Fig. 2). Here, we assume that the range of \( x_l \) is bounded in \([0, 1]\) and \( \beta_l = 1 \) by using sigmoid activation function at middle layers. Note that the input layer is represented by continuous value like in [10]. The quantized DNNs meet a representation of discrete DNNs.

Our strategy for obtaining parameters of the discrete DNNs is to quantize the parameters of continuous DNNs that are trained under the boundary constraint of weights (boundary model). This strategy enables us to avoid a complex combinational optimization problem of discrete DNNs. Moreover, we also refine the training method that shrinks the boundary of parameters to reduce quantization error (contraction mapping).

#### 3.2. Boundary Model and Parameter Adaptation

An effective way to reduce the quantization error of parameters is to contract their range. Therefore, we adopt the bounded model for weights to explicitly express their range and to control their boundaries.

The weights \( W_l \) are controlled by the latent variables \( V_l \in \mathbb{R}^{M_l \times N_l} \), and their ranges are constrained by scaling parameter \( \alpha_l \). Given the independent mapping among \( V_l \) and \( W_l \), the bounded weights are represented by

\[
W_l = \alpha_l g_l(V_l) \quad (w_{l,ij} = \alpha_l g_l(v_{l,ij})), \tag{11}
\]

where \( g_l \) is a bounded function. The updating rules can be deduced by using back-propagation and the chain rules.

\[
\alpha_l \leftarrow \alpha_l - \eta \delta^T g_l(V_l) x_l \tag{12}
\]

\[
V_l \leftarrow V_l - \eta \alpha_l (\delta^T x_l \circ \frac{\partial g_l}{\partial V_l}(V_l)) \tag{13}
\]
where \( \circ \) is an operator of the element-wise multiple of matrices. \( V_l \) is updated instead of \( W_l \) in the training.

The update of scale parameters \( \alpha_l \) is weighted by the boundary function concerning the correlation of input and propagated error. Therefore, the contraction by weights is expected. The updates of \( V_l \) are also controlled by the gradient of \( V_l \). The design of the bounded function and the training rule is important to achieving the weight contraction.

We adopt the well-known bounded function \( \tanh(x) \) whose range is \([-1, 1]\) (Fig. 3). In this paper, we try following two functions with different derivatives.

\[
g_{1,l}(x) = g_1(x) = \tanh(x), \quad (14)
g_{2,l}(x) = g_2(x) = \tanh(x^2). \quad (15)
\]

Function \( g_2 \) is expected to force the parameters and boundary to shrink compared to \( g_1 \), because of the steep gradient \( g_2 \). Figure 4 shows the shape of each derivative. In Eq. (13), \( g_1 \) weights the interval, where the value is small when updating \( V_l \), and \( g_2 \) does the interval around 1.0.

### 3.3. Contraction Mapping

The training method described above can enforce the parameters to be inside the defined boundary. However, parameters around boundary could not be updated due to the small gradient of these functions. The contraction mapping is used to enforce the boundary shrinkage to compensate for the small gradient problem of \( g_1 \) and \( g_2 \). We adopt the following empirically-defined contraction mappings in this paper (Fig. 5).

\[
\alpha_l \leftarrow \max_{i,j} \left| w_{l,ij} \right| \quad (16)
\]

\[
v_{l,ij} \leftarrow \text{sgn}(v_{l,ij}) \frac{|w_{l,ij}|}{\alpha_l} \quad \text{for } g_1 \quad (17)
\]

\[
v_{l,ij} \leftarrow \text{sgn}(v_{l,ij}) \frac{|w_{l,ij}|^{1/3}}{\alpha_l} \quad \text{for } g_2 \quad (18)
\]

This operation moves the values from the large-value region to the small-value one although it does not move the originally small values around 0. The timing and the frequency of applying this mapping during training is empirical and described in Section 4, and thus, we evaluated the effect by experiments. The mathematical correctness or analysis, such as the stability of convergence, has not been tackled in this paper.

### 4. Evaluation

#### 4.1. Experimental setup

We evaluated our training method by conducting large vocabulary continuous speech recognition experiments using the Corpus of Spontaneous Japanese (CSJ), which is a collection of Japanese lecture recordings [14]. While the original sampling rate of CSJ is 16 kHz, our evaluation was conducted under the sampling rate of 8 kHz. At first, the performances of our DNNs without quantization were compared to the DNNs with normal training parameters and a GMM trained by using the minimum phone error criterion [15]. Here, the performance was measured by the word and phone accuracy with/without a language model (LM). Note that we evaluated the accuracy without LM, which means the language model weight was set to 0, to see the pure performance of each acoustic model. Next, we revealed the relationship between the number of bits for quantization of \( W_l \) and the ASR performances. Finally, the statistics of weights were analyzed.

The training data for the acoustic model of the DNNs and GMM contained 270 hours of lecture recordings. The evaluation data were test sets 1 and 2 of the CSJ, i.e., 4.5 hours of lectures featuring 20 speakers (15 male and 5 female). The training data for the language model contained the transcriptions of 2,671 lectures. We used a tri-gram model with 65,000 vocabulary words. A one-pass decoder based on weighted finite state transducer was used for the decoding [16].

The GMM-HMM contained 2,734 states based on a HMM with a tri-phone tied-state [17] and 32 Gaussian mixtures at each state. 13 mel-frequency cepstral coefficients (MFCCs), delta coefficients, and delta-delta coefficients with mean and variance normalization per utterance [18] were used as speech features.

The DNNs-HMM used the same HMM with a GMM-based model and \( L = 8 \) layers with 1024 hidden nodes. There were a total of 825 dimensions of features, including 11 frames (previous 5 frames and following 5 frames) of basic features. The basic features consist of 25 log Mel-filter bank coefficients (including one log energy feature), and the delta and delta-delta coefficients. The mean and variance normalization was applied to it. We used a discriminative pre-training method [2] for the pre-training and also used the AdaGrad method [19] for scheduling the update parameter \( \eta \). After fine tuning of the parameters, the bounded training was conducted with \( \eta = 0.005 \). The contraction mapping was applied at each epoch of the training and stopped at the appropriate epoch.

### 4.2. Results

#### 4.2.1. Accuracy and Quantization Error

Table 1 classifies the performance of DNNs and GMM without quantization. Here, the baseline represents the normally trained DNNs, the conditions without + represents discrete DNNs with only boundary model and + represents discrete DNNs with a combination of boundary model and contraction mapping. All the DNNs outperformed the GMMs by eight points in word accuracy and 16 points in phone accuracy w/o LM. This proves the high classification performance of DNNs. The differences between the baseline and \( g_1, g_2 \) with/without contraction mapping are small. We confirmed that our method does not degrade all the accuracies of the ASR.

Tables from 2 to 4 specify the ASR accuracies for differ-
ent numbers of bits for quantization. In the case of seven bits, the accuracies only slightly differ between the baseline and the others. However, our method, especially with the contraction mapping, outperformed the baseline for the quantization with smaller bits. By exploiting the language model, the degradation of $g_1 +$ with a 4-bit quantization was only two points in terms of word accuracy. Even the 4-bit quantized DNNs outperformed the GMM-based ASR.

The averaged quantization errors at each bit are listed in Table 5. The error for contraction mapping decreases to half the error of the baseline. The effectiveness of our method in error reduction was also confirmed. Moreover, these results also indicate that the bias parameters, $b_l$, play an important role in the classification especially in the deeper layers, because the scale of terms, $W_l\alpha_k$, in Eq. (3) is small due to the boundary contraction, and the influence of the terms on the outputs is small.

4.2.2. Statistical Analysis

Table 6 classifies the percentage of zero elements of weight for each layer in the 4-bit quantization. The ratio of zeros tended to decrease toward the last layer. The ratio for the contraction mapping was smaller than that of the others, which means the parameters were trained to be dense under restricted information bits. The sparseness of parameter and quantization training may be incompatible in principle.

The normalized kurtosis of weights for each layer is listed in Table 7. Kurtosis is a criterion to measure the sharpness of the probability density function (PDF), and those of Gaussian and uniform are 0 and −1.2. Since the kurtosis of the baseline became zero, the PDF of weights was just like that of Gaussian. On the other hand, the kurtosis of contraction mapping went below 0. This means the distribution of weights was close to that of uniform. Therefore, the contraction mapping has the function to make the distribution of weights relatively uniform.

5. Discussions

The bounded model and contraction mapping reduces the quantization error and maintains a high level of word accuracy. However, several issues remain in the training and implementation of DNNs in a bounded discrete system.

For example, the training scheme of contraction mapping is empirical, and a mathematical analysis of the convergence is required. Adding a constraint of uniform PDF or bit-level sparseness to the weights could be effective to reduce the quantization error. The usage of a different scale parameter $\alpha_l$ according to each output node is also promising. Finally, the parameter training as a combinational problem could be pursued.

The effective implementation of DNNs-based ASR in a discrete system is also important for creating applications. The size reduction of the look-up table is the main issue, and some efficient encoding techniques using the properties of the weights, such as the sparseness, are required by taking into account the actual computer architecture.

6. Conclusions

Representing the parameter of DNNs by small bits can reduce not only memory usage but also computational complexity. The quantization error degrades the word accuracy of DNNs-based ASR. The discrete DNNs are defined to formulate the quantized DNNs, and we focused on its boundary property. Then we introduced a bounded weight constraint and contraction mapping in training continuous DNNs parameters. In our experiments with 4-bit discrete DNNs, while simply quantizing normally trained DNNs degrades the word accuracy by more than 50 points, our method can maintain the high word accuracy of DNNs-based ASR.

We focused on its boundary property. Then we introduced a bounded weight constraint and contraction mapping in training continuous DNNs parameters. In our experiments with 4-bit discrete DNNs, while simply quantizing normally trained DNNs degrades the word accuracy by more than 50 points, our method can maintain the high word accuracy of DNNs-based ASR. The bounded model and contraction mapping reduces the quantization error and maintains a high level of word accuracy. However, several issues remain in the training and implementation of DNNs in a bounded discrete system.

For example, the training scheme of contraction mapping is empirical, and a mathematical analysis of the convergence is required. Adding a constraint of uniform PDF or bit-level sparseness to the weights could be effective to reduce the quantization error. The usage of a different scale parameter $\alpha_l$ according to each output node is also promising. Finally, the parameter training as a combinational problem could be pursued.

The effective implementation of DNNs-based ASR in a discrete system is also important for creating applications. The size reduction of the look-up table is the main issue, and some efficient encoding techniques using the properties of the weights, such as the sparseness, are required by taking into account the actual computer architecture.
7. References


