NEURAL NETWORK QUANTIZATION METHODS FOR VOICE WAKE UP NETWORK

Shurui Zhang1*, Xiangyu Li2, Chen Zhang3

1 Department of Microelectronics and Nanoelectronics, Tsinghua University, Beijing, 100089, China
1*Department of Microelectronics and Nanoelectronics, Tsinghua University, Beijing, 100089, China
2 Department of Microelectronics and Nanoelectronics, Tsinghua University, Beijing, 100089, China
3 Department of Microelectronics and Nanoelectronics, Tsinghua University, Beijing, 100089, China

Corresponding author’s e-mail: zsr18@mails.tsinghua.edu.cn

Abstract. Quantifying the pre-trained neural network model can reduce its storage size and speed up the forward-inference process. Previous researches for the quantization of voice wake up networks couldn't reach our anticipation for applying it to applications. The voice wake up network is mainly composed of recurrent neural network and classification function. Recurrent neural network can take preorder words into consideration and it widely used in voice network. In this paper, we propose method to quantize the structure of recurrent neural network including RNN, GRU and LSTM by optimizing the method of quantifying the activation function. At the same time, using the translation invariance of the classification function to improve classification accuracy on Speech commands dataset version 2. Using the method proposed in this paper, we can make the quantized model achieve a better voice wake up rate.

1. Introduction

Deep Neural Networks become important tools for artificial intelligence in applications like computer vision[1], speech recognition[2], natural language processing[3], and computer games.

However, inference and training of a Deep Neural Networks always need billions of calculations and millions inputs likes images[4]. A pre-trained Deep Neural Networks also have large number of storage size. The huge cost of computing and storage greatly limits the application of Deep Neural Networks in embedded devices. To solve the problem, many methods had been proposed from both hardware and software perspective. One method to solve the problems is to build efficient models from the ground up[5][6][7]. However training such models is usually time-consuming and difficult. Another method is applying quantization to compress the Deep Neural Networks[8]. Currently, quantization is the most widely used method in industry.

One method is the post training quantification method proposed by Google[9]. Another method is the quantization method based on K_L divergence proposed by NVIDIA[10]. This quantization method can directly quantify the pre-trained neural network models, and at the same time can achieve good quantization results. Other quantitative research on RNNs networks has the problem that the...
quantization accuracy cannot meet the needs of actual scenarios, can not be directly applied to embedded devices\cite{11}[12].

This paper makes the following three contributions:
1. Through the saturation quantization interception method of the activation functions tanh and sigmoid in the RNNs network, optimizing the existing quantization method for the RNNs network achieved better accuracy.
2. Using the translation invariance principle of softmax, the common classifier softmax in deep learning network models is quantified and optimized, and a better quantization effect is obtained.
3. The quantization and optimization of the neural network algorithm for the entire process of awakening the voice model is realized, and a good quantization effect is achieved. The quantization accuracy can meet the accuracy requirements of the actual scene.

2. Related Work

2.1 Preprocessing of weight and data

Weighting is to change the weight parameters in neural network to 8bit. Assume that the size of the weight is C*I*H*W, C is the output channel size, I is the input channel size.

Algorithm 1: Quantization of Weight

1. Calculate the maximum value: \(th^c\): Maximum absolute value of C output channels
2. Calculation weight quantization factor: \(S^c=\text{th}^c/127.0\)
3. Quantization of weight: \(W_{c\text{int}}=\text{RoundClip}(W^c/S^c)\)

The activation quantization is mainly to quantize the input and output of the convolution to 8bit. The activation quantification here uses Nvidia’s K-L divergence quantification method\cite{10}, by fitting the data distribution for quantification.

Algorithm 2: Quantization of Activation

1. Input: FP32 histogram H with 2048 bins: bin[0], ..., bin[2047]
2. For i in range(128, 2048)
   - reference_P = [bin[0], ..., bin[i-1]]
   - outliers_count = sum(bin[i], bin[i+1], ..., bin[2047])
   - reference_distribution_P[i-1] += outliers_count
   - P/=sum(P)
   - candidate_Q=quantize(bin[0], ..., bin[i-1]) into 128 levels
   - expand candidate_distribution_Q to ‘i’ bins
   - Q/=sum(Q); divergence[i] = KL_divergence(reference_P, candidate_Q)
3. Output: Find index ‘m’ for which divergence[m] is minimal
   - Threshold = (m + 0.5)*(width of a bin)

2.2 Quantitative method of saturation interception

Tanh and Sigmoid functions are the most commonly used activation functions in RNNs networks, playing important role in the voice wake up networks.

The calculation formulas of the tanh function and sigmoid function are as follows:

\[
\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (1)
\]

\[
\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2)
\]

The image is shown in figure1 and figure2:
Observing the images of tanh and sigmoid, we can find that tanh and sigmoid functions have the characteristics that the function is not sensitive to the input beyond a certain range.

Among them, we focus on using the feature of the sigmoid and tanh functions. We discover that the tanh function enters the saturated state after $x>3$, and the sigmoid function enters the saturated state after $x>5$. Therefore, we use the saturation interception method for quantization.

Algorithm 3 : Quantization of Tanh

1. Input: $maxc=3$
2. $Sc=\frac{thc}{127.0}$
3. Input($c$)int8=RoundClip(InputC/$Sc$)

Algorithm 4 : Quantization of Sigmoid

1. Input: $maxc=5$
2. $Sc=\frac{thc}{127.0}$
3. Input($c$)int8=RoundClip(InputC/$Sc$)

2.3 Quantization of Softmax

Softmax is a commonly used output function in machine learning. The calculation formulas of the tanh function and sigmoid function are as follows:

$$\text{softmax}(a) = \frac{\exp(a_i)}{\sum_j \exp(a_j)}$$

An important property of Softmax is translation invariance. $\text{softmax}(a + b) = \text{softmax}(a)$

Due to the existence of translation invariance, the model only needs to learn the relative size of the elements in $ai$, not the absolute size.

The image of $y = e^x$ is shown in figure 3. From the image, we can see that the exponential function is not sensitive to the numerical change when the value of $x$ is small, and the change of the exponential function can be clearly observed when the value of $x$ is large. Therefore, when quantizing the softmax function, we can use the translation invariance of softmax to increase the calculation accuracy of the exponential function, and then proceeding coefficient point, achieving a better quantization effect.
First, we quantized the input data of Softmax to get the input data of integer type, the quantization method is the same as the excitation quantization algorithm in algorithm 2. It is worth noting that when preprocessing the input data, we quantify the input data of an entire channel, and the data of a channel is composed of multiple matrices, and each matrix is composed of m*n rows and columns. The mapping range of 8bit quantization is [-128,127], so we choose the maximum value of each axis quantization result in the matrix. We use 127 minus the maximum in each matrix axis as the bias. The existence of the bias causes the input data to be translated. According to the translation invariance principle of the Softmax function, we know that the translation does not change the calculation result of the Softmax. But after the shift, the value of the exponential function is more discriminative, improving the quantization accuracy.

The algorithm for quantifying the Softmax function is shown below.

Algorithm 5 : Quantization of Softmax
1. Input: I_int8[M,N]
2. Shift: Max_axis = ReduceMax(I_int8,axis); Bias = 127 - Max_axis ; I_int8 = I_int8 + Bias
3. LUT: E_int8 = LUT(I_int8)

3. Quantizing the Recurrent neural network

3.1 Quantization of RNN
We first investigate quantization of RNN as it is structurally simpler. The basic structure of RNN cell can be described as follows:

\[ h_t = U x_t + W S_{t-1} \]
\[ S_t = f(h_t) \]
\[ O_t = g(V * S_t) \]

From the structure, we can see that there is a fully connected calculation and a tanh layer in the RNN network. In the calculation process of the RNN, it is necessary to separate H and X for fully connected operations, so there are two fully connected operations in the entire process, and each fully connected operation will have a weight. Therefore, a total of 9-layer quantization coefficients need to be calculated when quantizing the RNN network.

Algorithm 6 : Quantization of RNN
1. Weight Quantization: Using algorithm I to quantify the weight of the full-connected layer
2. Activation Quantization: Using algorithm 2 to quantify the input and output of each layer.

3. Tanh Quantization: Using algorithm 3 to quantify the function of tanh.

### 3.2 Quantization of GRU

The structure of GRU can be described as follows:

\[
\begin{align*}
Z_t &= \sigma(W_z[h_{t-1}, x_t]) \\
r_t &= \sigma(W_r[h_{t-1}, x_t]) \\
\tilde{h}_t &= \tanh(W*[r_t * h_{t-1}, x_t]) \\
h_t &= (1 - Z_t)* h_{t-1} + Z_t * \tilde{h}_t
\end{align*}
\]

Where \( \sigma \) stands for the sigmoid function.

There are three fully connected calculations in the GRU network, which are reset gate, update gate, and tanh layer. In the calculation process of the RNN layer, it is necessary to separate \( H \) and \( X \) for fully connected operations, so there are six fully connected operations in the entire process, and each fully connected operation will have a weight. Therefore, a total of 9-layer quantization coefficients need to be calculated when quantizing the GRU network.

Algorithm 1: Quantization of GRU

1. Weight Quantization: Using algorithm 1 to quantify the weight of the full-connected layer.
2. Activation Quantization: Using algorithm 2 to quantify the input and output of each layer.
3. Tanh Quantization: Using algorithm 3 to quantify the function of tanh.
4. Sigmoid Quantization: Using algorithm 4 to quantify the function of sigmoid.

### 3.3 Quantization of LSTM

The structure of LSTM can be described as follows:

\[
\begin{align*}
\tilde{f}_t &= \sigma(W_f[h_{t-1}, x_t] + b) \\
\tilde{i}_t &= \sigma(W_i[h_{t-1}, x_t] + b) \\
\tilde{C}_t &= \tanh(W_c[h_{t-1}, x_t] + b) \\
C_t &= \tilde{f}_t * C_{t-1} + \tilde{i}_t * \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b) \\
h_t &= o_t * \tanh(C_t)
\end{align*}
\]

There are four fully connected calculations in the LSTM network, which are forget gate, reset gate, update gate, and tanh layer. In the calculation process of the LSTM layer, it is necessary to separate \( H \) and \( X \) for fully connected operations, so there are eight fully connected operations in the entire process, and each fully connected operation will have a weight. Therefore, when quantizing the GRU network, a total of 12-layer quantization coefficients need to be calculated.

The method of quantifying the LSTM network is the same as that of the GRU network.

### 4. Experiment

A voice wake up network is always made up of the RNNs network and the Softmax classification function. So in order to check the effectiveness of the quantitative method mentioned above, we built a common voice neural network wake up model. On Speech commands dataset version 2\cite{13}.

#### 4.1 Network structure

In order to test the wake up rate of the quantified result for the voice network, we constructed a voice wake up network following network structure, the network model is shown in figure 4. By verifying the wake up rate of the network, we can get more practical experimental conclusions.
4.2 Results
We conducted the four set of comparative experiments.

Table 1. The accuracy of quantify RNNs

| Model | Float | K_L-8bit | 4bit | 8bit | 16bit |
|-------|-------|----------|------|------|-------|
| RNN   | 0.8614| 0.8274   | 0.8276| 0.8301| 0.8332|
| GRU   | 0.8648| 0.8464   | 0.8443| 0.8478| 0.8516|
| LSTM  | 0.8676| 0.8489   | 0.8427| 0.8502| 0.8547|

From Table 1, we can see that when only the activation function in the RNNs network is quantized by the method in this paper, it can achieve a precision similar to the K_L-8bit quantization method when performing 4bit quantization.

Table 2. The accuracy of quantify Softmax

| Model | Float | K_L-8bit | 4bit | 8bit | 16bit |
|-------|-------|----------|------|------|-------|
| RNN   | 0.8614| 0.8274   | 0.8282| 0.8312| 0.8357|
| GRU   | 0.8648| 0.8464   | 0.8463| 0.8492| 0.8537|
| LSTM  | 0.8676| 0.8489   | 0.8479| 0.8523| 0.8566|

From Table 2, we can see that when only the Softmax classifier is quantized, the quantization accuracy has improved a lot.

Table 3. The accuracy of quantify RNNs and Softmax

| Model | Float | K_L-8bit | 4bit | 8bit | 16bit |
|-------|-------|----------|------|------|-------|
| RNN   | 0.8614| 0.8274   | 0.8344| 0.8391| 0.8395|
| GRU   | 0.8648| 0.8464   | 0.8573| 0.8539| 0.8664|
| LSTM  | 0.8676| 0.8489   | 0.8522| 0.8574| 0.8648|

In the experiment in Table 3, we used the quantization method in this article for the entire wake-up network, and achieved the best quantization effect.

Table 4. The accuracy compared to Google-Retrain 8bit

| Model | Float | Google-Retrain 8bit | 8-bit |
|-------|-------|---------------------|-------|
| RNN   | 0.8614| 0.8402              | 0.8391|
| GRU   | 0.8648| 0.8546              | 0.8539|
| LSTM  | 0.8676| 0.8603              | 0.8574|

From Table 4, we can see that using the quantization method in this article, we can achieve similar accuracy to the quantization method proposed by Google that requires retraining, can save the retraining process and directly quantify the neural network.

4.3 Analysis of results
Through analyzing the quantitative results, we can see that compared with the current mainstream direct quantification methods, the quantification method in this paper can achieve better quantification results.

For activation functions such as tanh and sigmoid that have saturation values, the quantization method using saturation interception can improve the quantization accuracy to a certain extent. For Softmax function, the translation invariance of the Softmax function is used to translate the original data to the quantized sensitive area, obtaining better classification accuracy.

Compared to the quantization method proposed by Google, our accuracy have slight loss. But the quantization method proposed by Google need 32-bit floating-point operation during backward. So we give priority to the direct quantization method.

5. Conclusion and Future Work
We propose an effective neural network quantification method for voice wake up networks. To quantizing the structure of RNN, GRU and LSTM through optimizing the method of quantifying the activation function, together with the translation invariance of the Softmax classification function to
improve the effects. Our methods have achieved the state of the art in accuracy under the Speech commands dataset version 2.

Acknowledgments
I would like to express my gratitude to Mr. Li for his help in the process of paper.
This work is supported by the Research and Development Plan in Key Areas of Guangdong Province, China (No.2019B010117002).

Reference
[1] Alex Krizhevsky and Ilya Sutskever. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pp. 1097–1105, 2012.
[2] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning. Coursera, video lectures, 264, 2012b.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.
[4] Lanctot, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.
[5] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. “Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation,” 2018.
[6] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, and W. Wang. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017.
[7] F. N. Iandola, M. W. Moskewicz, K. Ashraf, and S. Han. “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size,” CoRR, vol. abs/1602.07360, 2016.
[8] V. Sze, Y. Chen, T. Yang, and J. S. Emer. “Efficient processing of deep neural networks: A tutorial and survey,” CoRR, vol. abs/1703.09039, 2017.
[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, and M. Tang, “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,” Dec. 2017.
[10] Nvidia, “8 bit inference with TensorRT.”http://on-demand.gputechconf.com/2017/s7310-8-bit-inferencewith-tensorrt.pdf.
[11] Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent neural networks with limited numerical precision. arXiv preprint arXiv:1608.06902, 2016.
[12] Itay Hubara, Matthieu Courbariaux, and Daniel Soudry. Quantized neural networks: Training neural networks with low precision weights and activations. arXiv:1609.07061, 2016a.
[13] http://download.tensorflow.org/data/speech_commands_v0.02.tar.gz