EasyChair Preprint

№ 1021

Application of fault tolerant calculation in small-footprint keyword spotting neural network

Yicheng Lu, Weiwei Shan and Jiaming Xu

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

May 26, 2019

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 0?/?? pp???-???
Volume 18, Number 3, June 2013

Application of fault tolerant calculation in small-footprint keyword spotting neural network

Yicheng Lu, Weiwei Shan[*], Jiaming Xu

Abstract: In recent years, many applications of voice wake-up technology have entered peoples field of vision. The key technology is Keyword Spotting. The system needs to detect the ambient voice waiting for a wake-up at any time, so it requires a low hardware energy and high recognition accuracy. This paper aims at real-time speech keyword detection applications. Based on Googles open source speech commands dataset and Librispeech dataset, combined with various fault-tolerant calculations, a deep neural network that suitable for low-power integrated circuits are constructed and trained. The main structure of the network is the Depthwise Convolution Network (DSC). The energy consumption and resource overhead of the model in hardware implementation is reduced by combining various fault-tolerant calculation methods such as approximation addition, quantification, and binarization. The fault tolerance of the model is improved through retraining method. We proved that the fault-tolerant calculation method of quantization with approximation addition has great potential in small-footprint keyword spotting neural network.

Key words: Keyword spotting; Approximate addition; Quantification; Depthwise convolution

1 Introduction

The goal of keyword spotting is to detect a relatively small set of voice commands. It is commonly used in mobile phones and smart hardware. Keyword detection on such devices typically has two main uses. One is to identify common command word recognitions such as ”on” and ”off” and other common words such as ”yes” and ”no”. Second, keyword detection can be used to identify wake words, such as ”Hi Siri.” This gives the device a clear indication of the opening.

The online speech recognition systems need to be called from the cloud to call the model on the device, which brings some user privacy leakage during the networking process. In addition, the application devices are mostly power-consuming, performance-sensitive devices, and the keyword wake-up system usually needs to maintain a normally open state, thus not allowing the system to have large power consumption and memory usage. So keyword spotting task requires a low power offline model.

In recent years, neural networks have be proven to provide an effective solution for low-power keyword wake-up systems.The biggest difference between the KWS keyword wake-up system based on deep neural network and the traditional hidden Markov model (HMM)[12] is that the deep keyword wake-up system does not need to separately distinguish the phonemes of the speech, but an end-to-end model.

In this work, we focus on convolutional neural networks (CNNs), which have also be proven to be suitable for keyword wake-up systems in recent years[13]. We use the neural network smiliar to MobileNet[11], combined with various fault-tolerant calculation methods such as approximation addition,

• Yicheng Lu is with School of Electronic Science and Engineering, Southeast University, NanJing and 210096, China. E-mail:220181346@seu.edu.cn
∗ Weiwei Shan is with School of Electronic Science and Engineering, Southeast University, NanJing and 210096, China. E-mail:wwshan@seu.edu.cn
• Jiaming Xu is with School of Electronic Science and Engineering, Southeast University, NanJing and 210096, China. E-mail:seu_xjm@163.com

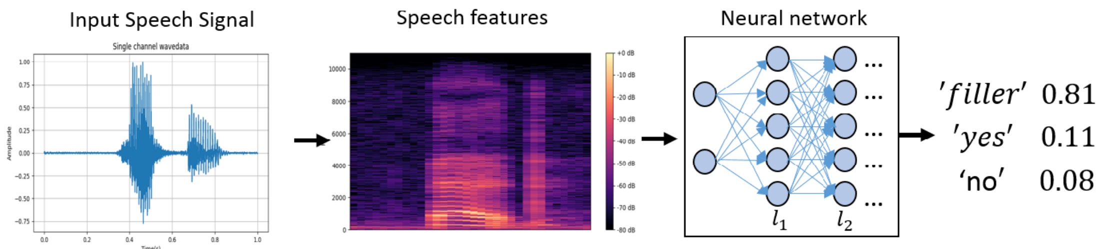


Fig. 1 Deep keyword spotting system

quantization and binarization, etc. to reduce the amount of parameters and calculation of the model.

We proved that the method of quantization with approximation addition can reduce storage and computational complexity at the same time with an acceptable decline of performance, which is better than the method only use quantization.

We describe our model and fault-tolerant calculation methods used in section2. Experimental environment and training methods are elaborated in section3. Results and some discussion follow in Section4. Section5 makes a summary.

2 System design

The general process of end-to-end keyword wake-up system can be divided into several parts is shown in Figure 1. First, feature extraction of the original audio data is performed to obtain a highly representative feature vector. The feature vector is then sent to the deep neural network, which outputs the probability that the audio belongs to a certain type of word. The most important part of KWS system this paper talk about is feature extraction and neural network model.

2.1 Feature Extraction

The feature extraction method is the Mel Frequency Cepstrum Coefficient(MFCC), which is common to related work[9][19]. The frame window length is 36ms and step length is 16ms. The first 10 Mel frequency filters are used for each frame. The word duration of Speech commands dataset is 1000ms but we use 512ms to reduce power consumption. The output of MFCC module is a 300-D feature vector, corresponds to 30 frames of 512ms audio clip(10 filters/frames). Actually, the MFCC features mainly distributed between -64 and 63, as shown in Figure 2. So we quantify the input value to 8bit.

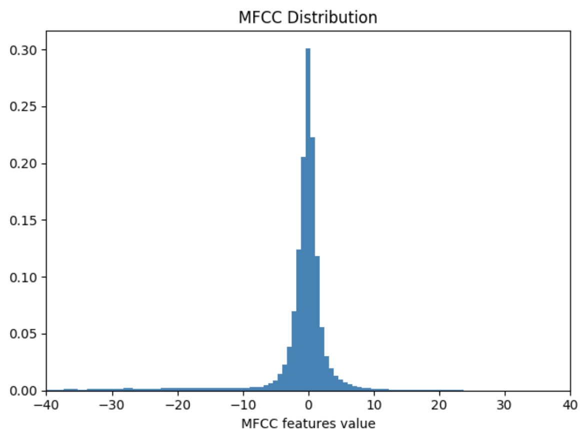
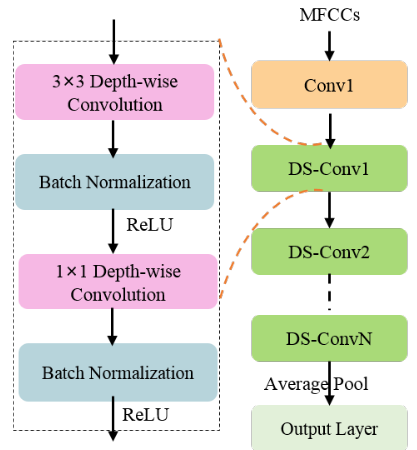
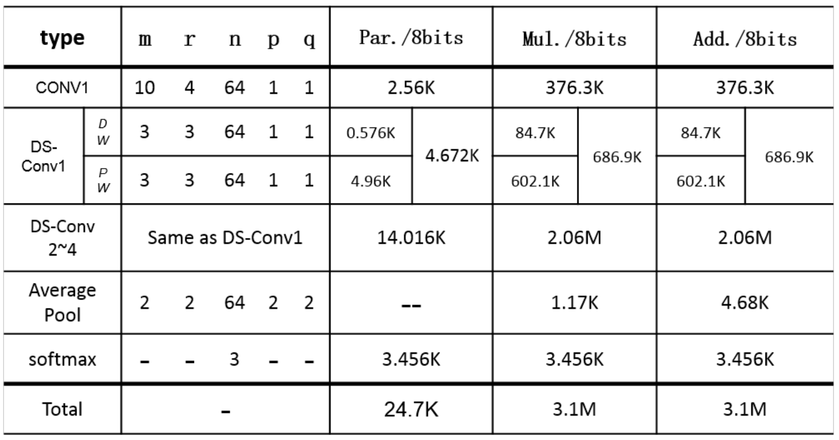


Fig. 2 Distribution of MFCC output

2.2 Neural network model structure

Recently, depthwise convolution(DSC) has been proposed to replace the standard 3-D convolution operation[10] and has been used in the field of computer vision to implement compact network architectures[11][15]. The DSC first convolves each channel in the input feature map with a separate 2-D convolution kernel and then combines the outputs using point-by-point convolution (ie, 1 1 convolution). By decomposing standard 3D volume into 2D convolution, depthwise convolution is more efficient in terms of the number of parameters and operations, which allows more complex networks to migrate to hardware-sensitive applications such as mobile. The baseline network structure we use is similar to the structure in[4]. As shown in the Figure 3, the input MFCC feature values are first processed by a traditional 2-D convolution layer. Then the output is connected with several DS-Conv convolution modules and finally flattened and connected to the fully connected output layer. Each DS-Conv module consists of a 3x3 Depth-wise Conv, a 1x1 point-by-point convolution, and a number of Batch Normalizations[16]. The details of model parameter and operator number are listed in





type		m	r	n	p	q	Par./8bits		Mul./8bits		Add./8bits	
CONV1		10	4	64	1	1	2.56K		376.3K		376.3K	
DS-Conv1	DW	3	3	64	1	1	0.576K	4.672K	84.7K	686.9K	84.7K	686.9K
	PW	3	3	64	1	1	4.96K		602.1K		602.1K	
DS-Conv 2~4		Same as DS-Conv1					14.016K		2.06M		2.06M	
Average Pool		2	2	64	2	2	--		1.17K		4.68K	
softmax		-	-	3	-	-	3.456K		3.456K		3.456K	
Total		-					24.7K		3.1M		3.1M	

Fig. 3 Architecture of baseline model. m and n represent the time and frequency span of the convolution kernel. The kernel can stride by a non-zero amount p in time and s in frequency. n represent the output channel number of convolution kernel. The last three columns count the amount of model parameters and the number of multiplications and additions. The initial parameters are all 8 bit numbers.

Figure 3.

2.3 Fault-tolerant calculation method

Since the neural network itself has certain fault tolerance, especially through the retraining method, the negative impact of the erroneous operation on the network recognition rate can be weakened or even offset. This article focuses on several methods of approximate calculation: (i)approximate addition, (ii)quantification, and (iii)binarization.

2.3.1 Approximate addition

Approximate addition has been studied in related papers[20][21] this year. The approximate adder used in this paper is shown in the fig 4 and is divided into two parts. One part is the precision adder and the other part is the approximate adder. Assuming that two n bits data are added, setting the exact number of bits to m, the exact adder calculates the sum of the first m bits of the n bits data, and the remaining $n - m$ bits are replaced with logical 'OR' operations. In fact, the use of 'OR' operations eliminates the carry compared to the exact addition, so the amount of computation can be reduced to some extent.

2.3.2 Quantification

Quantification, also known as fixed point. Refers to quantizing the weights and outputs in the network to a fixed number of bits. In the training of neural networks, data of full precision(32 bits) is generally used. In fact, too high data precision is redundant. Therefore, the main idea of the quantification method is to lower

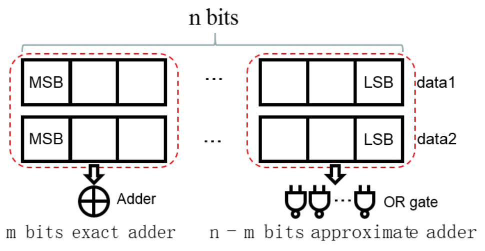


Fig. 4 Approximate adder

the weight and output value of the original high bit width by a certain bit width. Some related work[3][4] change the network data from 32bit to 8bit, which usually has little impact on the accuracy of the network. The quantification method used in this paper is shown in Algorithm1. Suppose the input is a 32-bit floating point number. First round it to an integer, then limit its maximum and minimum values to $(-2^{n-1}, 2^{n-1} - 1)$, where n represents the quantization data width.

Algorithm 1 : Quantize input float data x to integer y, given a target bit width m.

```
$n \leftarrow bit\_width(x)$
$b \leftarrow 2^{n-1}$
$x^{'} \leftarrow round(x)$
$y \leftarrow clip(x^{'}, -b, b-1)$
```

2.3.3 Binarization

Binarization method uses the two values for the weight or activation value. The method of binarization was first proposed by Bengio et al. in 2016 [5][17]. They use the values +1 and -1 to binarize both the weights and activations. Many forward multiply operations can be converted to 'XOR' operations during forward propagation[18], greatly simplifying the design of hardware circuit. The binarization method used in this paper is based on the results of Bengio et al., using the symbolic function $sign()$ for weights or activation values.

3 Experimental

3.1 Dataset

Our training set consists of two parts of the data. The first set is Google's Speech Commands dateset[1], which contains 30 kinds of keywords and several background noise. The second set is Librispeech[2], which is one of the world's largest ASR dataset. The wake-up keywords this paper use are Happy and Dog. Speech Commands dataset contains around 1700 clips of each word. To extend the data set, we cut out the keywords audio clips from Librispeech dataset with Deepspeech model[6], which can transform speech to text(STT). Finally, dataset contains about 4000 clips of each keyword and the remaining 60,000 audios are set as filler(unknown word) words. The dataset is split into training, validation, and test with a ratio of 8:1:1.

3.2 Train

All models are trained in the same framework(Google Tensorflow framework[7]), using the cross-entropy loss and Adam optimizer[8]. Most fault-tolerant calculation functions have no continuous derivatives. So the model with fault-tolerant methods are trained by straight-through estimator(STE)[14], which simply replaces the derivative of each fault-tolerant functions with with the identity function. Models are trained with a mini-batch of 100 and iterate 15k times. The initial learning rate is $5 * 10^{-3}$, which is reduced to $1 * 10^{-4}$ after 12k iterations. During the training process, background noise from Speech commands dataset is randomly mixed into the audio.

4 Results and Discussion

4.1 Training results

The baseline model is the 8bits version of DSC model mentioned in Section 2. We first compare the baseline model with traditional DNN[9] and CNN[4] model in terms of parameter number, calculation amount and test set accuracy. The result can be seen in Table 1. Obviously, the baseline model trades a larger amount of calculation for a relatively small amount of parameters and a high test set accuracy.

Model	Par. /8bits	Ops (Mul. and Add.)	Test accuracy
DNN1[9]	81.2k	195k	98.921%
CNN1[4]	45.1k	5.1M	99.442%
DSCNN	24.7k	6.2M	99.586%

Table 1 Comparison of baseline model with traditional DNN[9] and CNN[4] model

4.1.1 Quantification results

To reduce the amount of calculation without changing the model structure, the calculation process needs to be streamlined. There are roughly two ways to reduce the amount of calculations. The most straightforward method is to quantify network parameters and activations. The other way is to simplify the multiply and add calculation. Table 2 shows the result of quantify parameters of baseline model with less bits. In order to compare the quantized network calculations, we introduce the bit operations(bOps):

$$bOps = Bit\_width * Operations$$

For example, the baseline model's bOps is $8 * 6.2M = 49.6MbOps$. The most extreme quantification is to quantify the parameters to 1bit, which equivalent to binarized neural network(BNN) as shown in the last row of the table 2. Figure 5 is the modified receiver operating characteristic (ROC) curves of quantized models, where we use false reject rate on Y-axis instead of true positive rate. Assume the maximum probability of model output is P_{max} and the threhold of output is T. If $P_{max} > T$, the prediction label of the model is the one corresponding to P_{max}. Otherwise, the prediction label will be considered as 'filler'. When sweeping the threhold T, model's prediction will change, so that different false reject rate and false alarm rate can be caculated. Each group of false alarm rate (X axis) and false reject rate (Y axis) corresponds to one point on ROC. The closer the ROC is to the coordinate axis, the

Model	bOps	Test accuracy	AUC (10^{-3})
Baseline(8bits)	49.6M	99.586%	0.16
quantized-7bits	43.4M	99.628%	0.25
quantized-6bits	37.2M	99.598%	0.26
quantized-5bits	31.0M	99.572%	0.19
quantized-4bits	24.8M	99.172%	0.76
quantized-1bits(BNN)	6.2M	98.187%	4.50

Table 2 Quantized network comparison

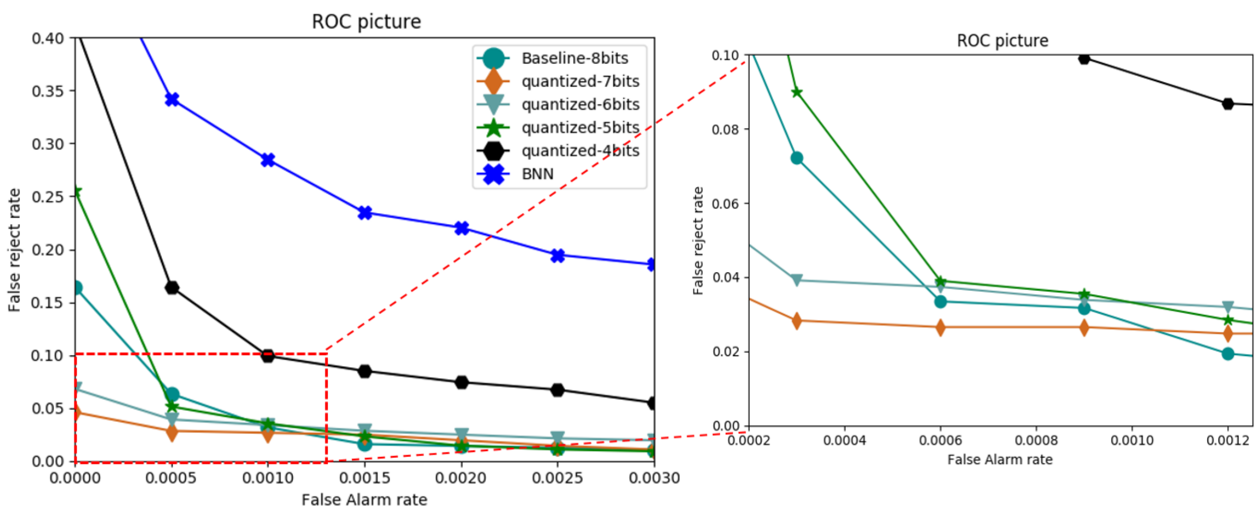


Fig. 5 ROC of quantized models

better the model is. The last column of table 2 is AUC, which is the under area of ROC curve. Smaller AUCs is better. It can be seen from the table 2 and Figure 5 that when the quantization bit width is gradually reduced from 8 to 5, the difference between the test set accuracy rate and the AUC value of the model is small, and when it is reduced to 4 bits, the accuracy of the model and the AUC value have a large decline. The model that was quantized to 1 bit(BNN) performed the worst, which is also in line with expectations.

4.1.2 Approximate addition result

Quantization only changes the calculated bit width of the model, which reduces the amount of calculation to some extent. The approximate addition greatly reduces the computational complexity from the addition operation. Based on the quantized modelwe replace the addition operation in the framwork with approximate adder mentioned in Section 2. On a model quantized to 7-bit, 6-bit, and 5-bit, about half of the number of bits is approximatly added. The result is shown in the Table 3 and Figure 6. Qx_Ay means the parameters of the model are quantized to x bits, where y bits are approximated. From the perspective of model accuracy

Model	Test accuracy	AUC (10^{-3})
Q7_A4	98.900%	1.16
Q7_A3	99.171%	0.49
Q6_A3	99.286%	0.65
Q5_A3	99.305%	0.66
Q5_A2	99.357%	0.45

Table 3 Quantized network with approximate addition

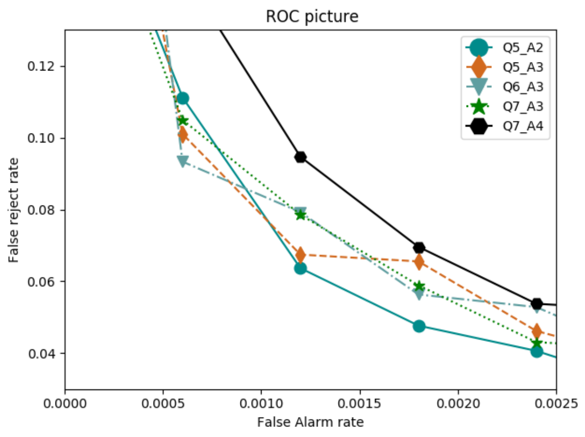


Fig. 6 ROC of Approximate addition model.

and AUC values, the model that quantized to 5-bit with 2-bit approximate adder achieves better results. By quantifying to 5 bits and accumulating approximately 3 bits, the computational complexity of the model can be minimized under the premise of obtaining higher model accuracy and AUC values.

5 Conclusions

We tested a variety of fault-tolerant calculation methods on our DSC baseline model. For the quantization model, we find that the model still maintains high accuracy and AUC value when quantization bit is reduced from 8 bits to 5 bits. The model accuracy begins to drop sharply while the number of quantization bits continues to decrease from 4 bits to 1 bit(BNN). Compared to the baseline 8-bit version, the 5-bit quantized model can achieve a $\times 37.5\%$ reduction in bOps. In addition, we found that the 5-bit quantization with 3 bits approximate adder model can further reduces the computational complexity of the model and sacrifices only 0.26% accuracy rate and 0.47×10^{-3} AUC relative to the 5-bit quantization model. It can be seen that the fault-tolerant calculation method of quantization with approximation addition can reduce storage and computational complexity at the same time with an acceptable decline of performance, which has great potential in small-footprint keyword spotting neural network.

References

[1] Warden P . Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition[J]. 2018.

[2] Panayotov V , Chen G , Povey D , et al. Librispeech: An ASR corpus based on public domain audio books[C]// ICASSP 2015 - 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015.

[3] Shah M , Wang J , Blaauw D , et al. A fixed-point neural network for keyword detection on resource constrained hardware[C]// Signal Processing Systems. IEEE, 2015.

[4] Zhang Y , Suda N , Lai L , et al. Hello Edge: Keyword Spotting on Microcontrollers[J]. 2017.

[5] Courbariaux M , Bengio Y . BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1[J]. 2016.

[6] Amodei D, Ananthanarayanan S, Anubhai R, et al. Deep speech 2: End-to-end speech recognition in english and mandarin[C]//International conference on machine learning. 2016: 173-182.

[7] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467,2016

[8] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[9] Chen G, Parada C, Heigold G. Small-footprint keyword spotting using deep neural networks[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014: 4087-4091.

[10] Chollet F. Xception: Deep learning with depthwise separable convolutions[J]. 2016. 18001807.

[11] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. 2017.

[12] Rose R C. A hidden markov model based keyword recognition system[J]. Proc of Icassp Albuquerque Nm Usa, 1990, 1:129132 vol.1.

[13] Arik S O, Kliegl M, Child R, et al. Convolutional recurrent neural networks for small-footprint keyword spotting[J]. 2017.

[14] Tjandra A, Sakti S, Nakamura S. End-to-end feedback loss in speech chain framework via straight-through estimator[C]//ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019: 6281-6285.

[15] Chollet F. Xception: Deep learning with depthwise separable convolutions[J]. 2016. 18001807.

[16] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. 2015. 448456.

[17] Courbariaux M, Bengio Y, David J P. Binaryconnect: training deep neural networks with binary weights during propagations. International Conference on Neural Information Processing Systems, 2015. 31233131.

[18] Rastegari M , Ordonez V , Redmon J , et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks[J]. 2016.

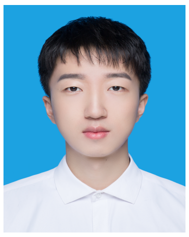
[19] Wang J C , Wang J F , Weng Y S . Chip design of MFCC extraction for speech recognition[J]. Integration, The VLSI Journal, 2002, 32(1-2):111-131.

[20] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan and K. Roy, IMPACT: IMPrecise adders for low-power approximate computing, IEEE/ACM International Symposium on Low Power Electronics and Design, Fukuoka, 2011, pp. 409-414.

[21] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie and C. Lucas, Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications, in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850-862, April 2010.

Yicheng Lu received the B.S. degree in electronic engineering from Southeast University, Nanjing, China, in 2018, where he is currently pursuing the M.S. degree in electronic engineering. His research mainly focuses on Keyword Spotting circuit design and On Chip training.

Weiwei Shan received the B.S. degree in microelectronics from Tianjin University, Tianjin, China, in 2003, and the Ph.D. degree in Microelectronics from Tsinghua University, Beijing, China, in Jan. 2009. She is an associate professor in National ASIC center at Southeast University, Nanjing, China. She was a visiting professor at Columbia University, New York, USA from 2018-2019. Her research mainly focuses on variation resilient adaptive VLSI circuits, ultra-low power SoC design and countermeasure techniques of security circuits. She has published over 40 technical papers in conferences and journals, and authorized over 15 invention patents.



Jiaming Xu received the B.S. degree in electronic engineering from Southeast University, Nanjing, China, in 2017, where he is currently pursuing the M.S. degree in electronic engineering. His research mainly focuses on low power integrated circuit design and Keyword Spotting circuit design.