An Energy-efficient Speech Classification Convolution Neural Network Accelerator Based on FPGA and Quantization

Dong Wen[1], Jingfei Jiang[1,*], Yong Dou[1], Jinwei Xu[1], Tao Xiao[1]

Abstract

Deep convolution neural networks (CNN) have been shown to own unique advantages in acoustic tasks over recurrent neural networks (RNN). However, activation data in convolution neural networks is often indicated in floating format, which is both time-consuming and power-consuming when be computed. Quantization method can turn activation into fix-point, replacing floating computing into faster and more energy-saving fix-point computing. Based on this method, this article provides a design space searching method to quantize a binary weight neural network with least accuracy loss. We also design a specific accelerator on FPGA platform, which is high-throughput and energy-efficient compared with CPU or RNN-based accelerators.

Keywords: energy-efficient; reconfigurable computing; FPGA; quantization; sound classification

1. Introduction

Sound classification is a typical information analyzing task which is widely used in military and speech controlling. Since Deng, Yu et al introduced RNN and LSTM (Long-Short Stage Memory) [1] acoustic model into speech recognition and sound classification, LSTM has reached series of excellent performance in this area (Geoffrey H, et al 2012; Wan H, et al 2019). However, deep neural networks based on RNN are hard to be trained and parallelized due to complex structure and recurrent computation. When applied in actual missions, RNN models usually demand high performance GPU and CPU to satisfy computing power need. Such hardware platforms have up to hundreds of watt power consumption, which cannot meet requirements for energy-sensitive circumstance. On the contrast, CNN has been found to get excellent performance on acoustic model (Tom S, et al 2016; Muckenhirn H, et al, 2018; Palaz D, et al, 2015). Audio files can be transferred into feature maps or feature matrixes by wave-filtering algorithms (such as Mel Frequency Cepstral Coefficients algorithm) (Pakyurek M,, et al, 2020), acoustic CNN models then run on these maps just like input images for computer vision models. With tiny 3x3 or 5x5 convolution kernels, CNNs can be trained and forward faster than RNN for convoluting operation is easier to be parallelized and accelerated than recurrent computation. This advantage makes it possible to accelerate an acoustic CNN model on specific power-efficiency hardware platforms.

Gradient descent algorithm (Jyrki Kivinen M K W, et al, 1997), which is sensitive to numerical fluctuation (Perkins S, et al, 2003), is widely applied to train deep neural networks (DNN). To pursue best DNN performance, it is necessary to store data in full-precision format during training process. However, floating data format needs longer word-length to store and more circuit parts to compute, leading to more energy consumption and bigger circuit designing area (Liu S, et al, 2011). Also, the computing complexity of floating-point data makes it

[1] National University of Defense Technology, Changsha, China.
* Corresponding Author.
Corresponding Address: National University of Defense Technology, Changsha, China.
jingfeijiang@nudt.edu.cn

difficult to reduce computing cycle counts. Floating computation, although can keep precision well, has become the bottleneck of power-efficiency high performance computing.

Fortunately, some works (Han S, et al, 2016; Dundar G, et al, 1995) have proven that floating-point data is unnecessary to CNNs' forwarding tasks, low precision computing can achieve similar performance as well. These works provide quantization method, turning weight and activation data into fix-point data, integer data or even binary data with little accuracy loss. Based on kinds of quantized CNN models, there comes BNN (Binary Neural Network) accelerators (Guo P, et al, 2019; Liang S, et al, 2018; Conti F, et al, 2018) and GPUs supporting 8 bits integer data (Michael D, Ashish K, David R. Nvidia's Xavier et al, 2018; Nvidia. 2018) etc. These designs reduce power consumption greatly and have up to hundreds of speedup ratios compared with CPU platforms. It turns out that hardware with corresponding quantized CNN models can achieve excellent computing performance as well as high energy efficiency.

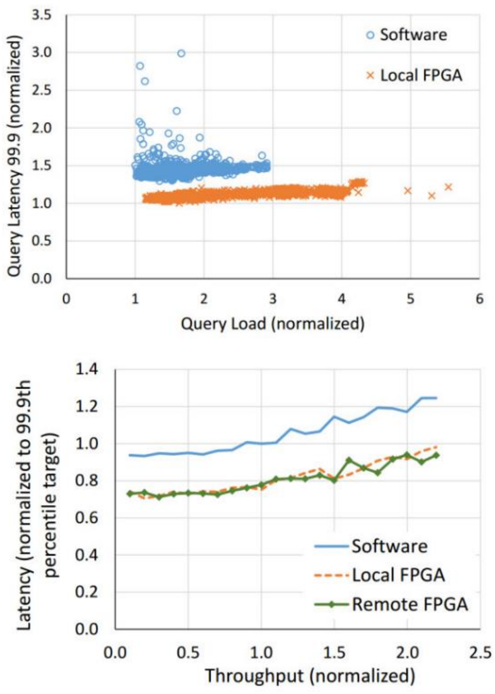


Fig.1 FPGA Accelerating Microsoft Bing Searching Task. (Hernandez, P eta al, 2018)

Compared with CPU and GPU, ASIC and FPGA are more suitable to accelerate a specific task for their reconfigurable feature. These reconfigurable platforms can be customized by setting pipeline and expanding parallelism degree, lowering power consumption and raising computing performance. Although ASIC owns huge advantages over FPGA on power and speed, expensive designing and manufacturing cost limits it's general application. On the contrast, FPGA keeps a good balance between performance, power, flexibility and expense due to programmable feature and mature industry design. Now, FPGA has been widely used in cloud computing and intelligence computing by Microsoft, Amazon and Alibaba (Gwennap L, et al, 2017; Turan F, et al, 2020), becoming an important part of high- performance computing.

The sound classification model which focuses on specific speech instructions or acoustics signal, is a basic component of intelligent scenario analysis in both cloud and edge end. Such applying circumstance needs a low-power but high-performance computing platform especially. Typical deep convolution neural networks can do coarse sound classification work well. However, there still exists some space to accelerate CNN model and reduce computing platform's energy consumption by quantization and customized hardware design. To implement this power-efficient sound classification computation platform, we choose a typical CNN-based speech classification model where weight value is +1 or -1 and activation data is in full precision floating-data format (Bo L, et al, 2018). We design an accelerator based on Xilinx XCKU-115 FPGA platform and run this BWN (Binary Weight Network) model. Compared with state-of-art CPU platform, our accelerator achieves 18-300x throughput speed up ratio and high energy efficiency. The main contributions of this work are as follows:

1. We turn float-type feature data into fix-point data each level by design space exploration method. Model's loss on accuracy will be covered by the performance of fix-point computing on FPGA platforms.
2. We design a multi-PE BWN accelerator on FPGA, which has shared weight storage, balanced pipeline structure and low-delay pipeline between CNN's levels. Also, the performance, power consumption and energy efficiency of this accelerator are discussed.
3. The target speech classification model is tested under single thread, multi-thread and multi-node environments to get sufficient performance baseline. Compared with these test results, our design has absolute advantage on performance per watt and throughput.

2. Neural Network Forwarding Quantization

When training a deep neural network, researchers usually choose full-precision data format to ensure best model accuracy. However, in inference task, these parameters will not be changed and therefore we can prune them in an offline method. (Bo L, et al, 2018) raises an algorithm to compress floating-point DNN parameters into binary data, which is consisted of +1 and -1. Compared with common DNN with floating-point weight and activation, this compression method not only sharply reduces parameter storage, but also replaces multiplication and division with add and minus. Less storage space and multiplication mean less memory-consuming energy and less computing cycles, leading to faster lower power consumption and faster working speed.

(Matthieu C, et al, 2016)brings out a method to turn activation into binary format. Unlike parameter in neural networks, activation data fluctuates numerically with different input (such like input image or input audio feature map). Although (Matthieu C, et al, 2016) still keeps a good model accuracy on very deep CNNs like VGGNet, great numerical precision loss would be brought out binary activation data, which may cause vital influence on some small-size CNNs (Jacob B, et al, 2017; Xu Y, et al, 2018). In this situation, turning floating data into fix-point format can keep a good balance between computing performance and model accuracy: fix-point data computation needs less computing cycles compared with floating data, and, fix-point can adapt to data's numerical distribution by flexible allocation of integer bitwise and decimal bitwise.

In Fig 2, when integer part is allocated with more bitwise, it can indicate bigger data; and when we give decimal part longer word-length, it can improve numerical precision correspondingly. However, increasing the length of fix-point data format will add cycle counts to computation or excess hardware's limitation, so taking speed and accuracy into account, it is important to find the relatively best data format.

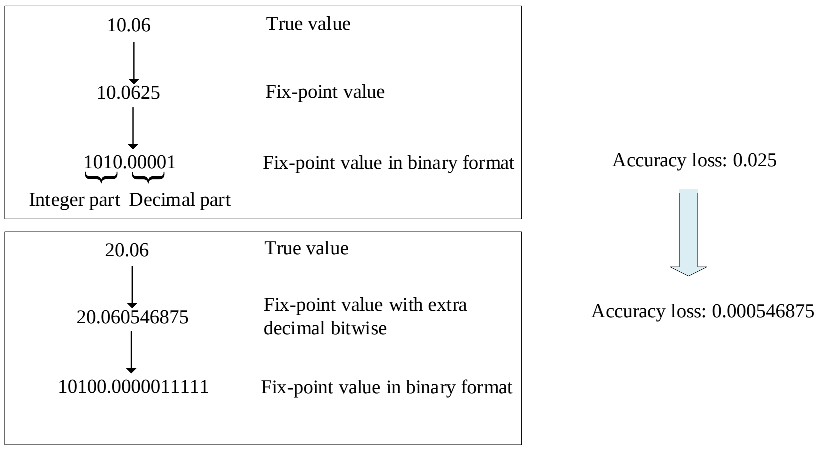


Fig.2 How Bitwise Influences Numerical Precision

3. Speech Classification Model

3.1 Model Architecture and Weight Binarization

This CNN-based speech recognition model is trained on Tensorflow speech command dataset. It can recognize the six sort of short speech segment “up”, “down”, “yes”, “right”, “left” and “unknown words”. This model first uses MFCC algorithm turning an audio file into a float-type tensor, whose dimension is 20x49x1. Then this tensor will be sent into a convolution neural network, which is consisted of two convolution layers, three full-connected layers and binary weight parameters. The detail information of model architecture is shown in Fig 3. All convolution kernel size is 3 and convolution stride is 1. There is no padding and expansion operation in this network, which is convenient for us to accelerate. To be noticed that activation is still in float format at this stage. Via softmax function, this model outputs the possibility of six type of labels.

After model parameters being fixed in the training process, we can transfer float weight value into (-1, 1). Fig 4 shows how we processing weight data. We assume the distribution of primitive parameter is normal distribution, the numerical distributing range is then modified by tanh function and a series of scale methods. Finally, all parameters are discretized to -1 or +1. This stage’s BWN model (activation data is still in floating format) can provide the accuracy no less than 85%.

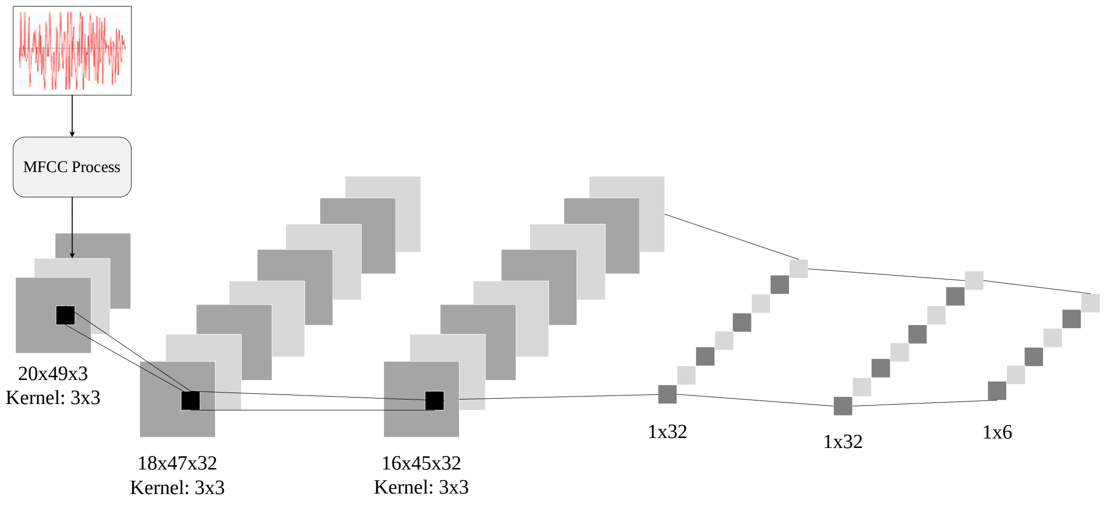


Fig. 3 Convolution Neural Network Architecture

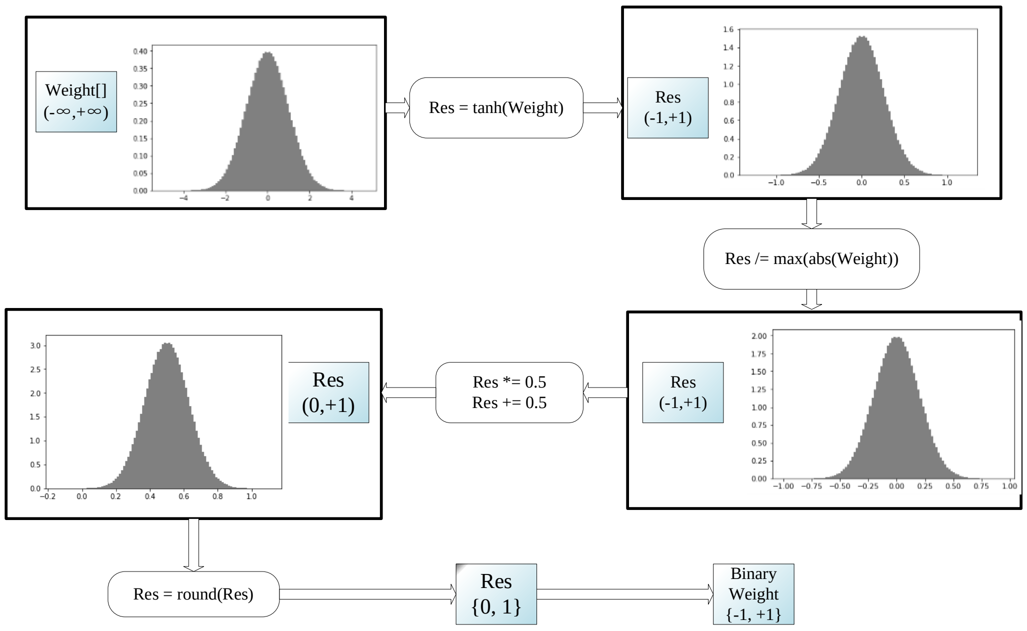


Fig. 4 Weight Binarization Process

3.2 Feature Data Quantization

Once we change floating input feature into fix-point format, middle result, activation and other hyper-parameter (such as bias and batch-normalization parameter) will be in fix-point format naturally. The data computed in each level can be divided to two parts: middle results after MAC (Multiply-Accumulate) operations and the batch-normalized output data (this result will be transferred to next level). In some cases, absolute value and variance value of middle results can be huge, both of them will be narrowed down in batch-normalization process. This batch-normalization step is vital to model's final accuracy (Santurkar S, et al, 2018; Liu M, et al, 2017). However, the multiplication and division in this process not only depend on prior mean and variance, but also are sensitive to numerical precision (Giri E P, et al, 2016). To ensure both hardware implement and model's performance, we need to apply special bitwise allocation method to two kinds of data.

In order to turn middle result and batch-normalized data into fix-point format with least precision loss, the approximate data range of these two kinds of data needs to be determined first for numerical distribution range deciding data's integer part bitwise. After allocating bitwise for integer, the decimal bitwise of prior parameters in batch-normalization should be set properly for extra precision demands from this process. Based on these ideas, we raise a design space search method to find best bitwise combination for two types of data and finally get a fix-point-activation BWN sound classification model. In the Experiment section, we will discuss the detail quantization experiment results.

4. Accelerator Architecture

The target model is small, shallow and weight-binarized, so the main methods we focus are setting shared parameters storage on chip, accelerating neural network level by level and designing pipeline between levels.

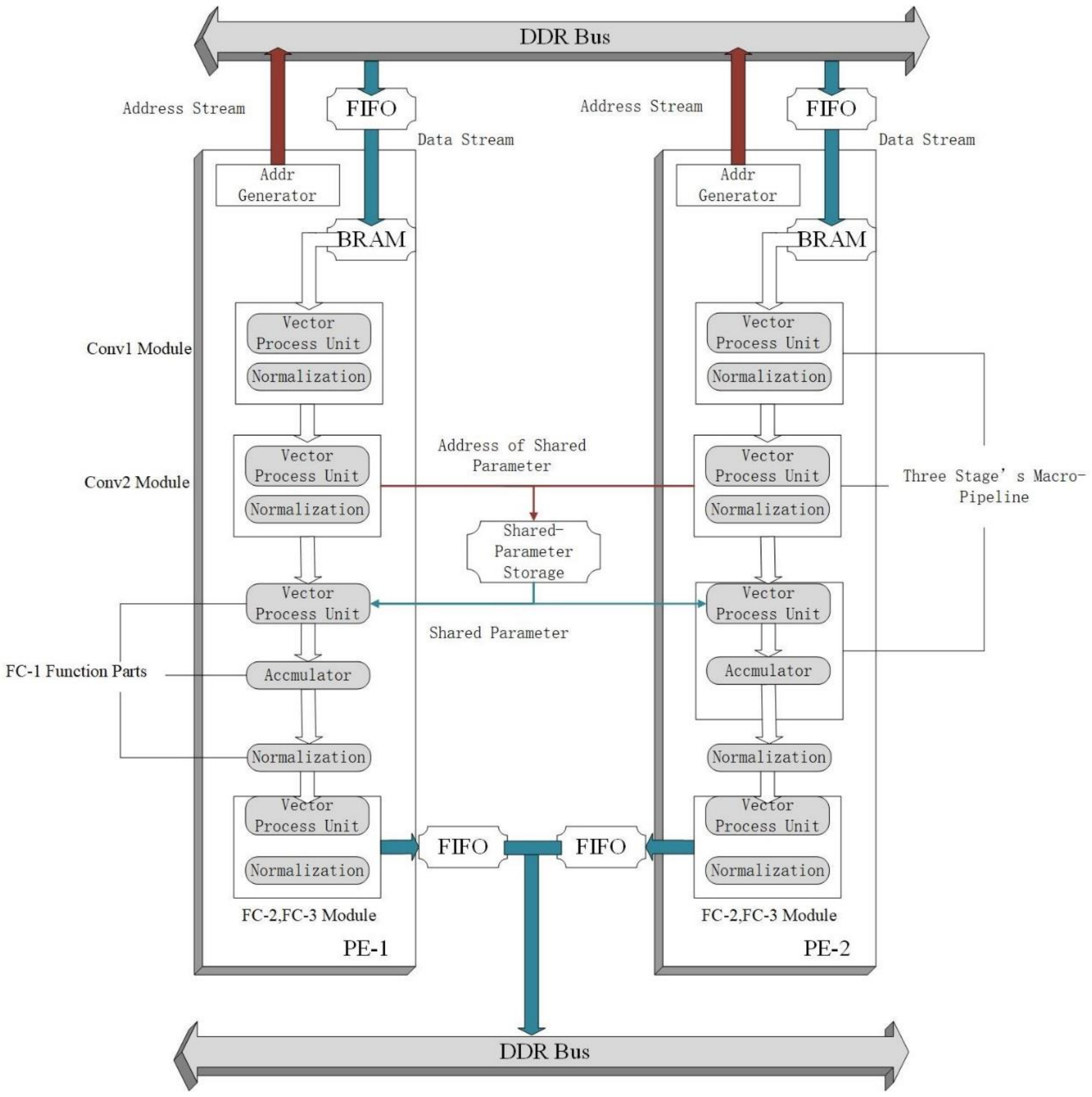


Fig.5 Hardware Architecture

4.1 Parameter Storage

Unlike some neural network accelerators having to use DRAM instead of on-chip memory to store parameters (Cheung K, et al, 2012; Alessandro A, et al, 2018; Chen T, et al, 2015), our accelerator can store all parameters on chip due to small-sized network architecture. With whole parameters stored on chip, we can save time cost caused by communication with DRAM. The detail information of parameters is showed in Table 1.

FC-1 layer occupies most of the parameter size, while other layers' data is rather tiny and can be directly stored on chip. Considering the scale of FC-1 parameter, it is natural to share them between several PEs. In order to simplify the design, we set all PEs working synchronously and fetching exactly the same pretrained data at FC-1 computing step. Shared memory structure is consisted of 32 block memory generators, each for one kernel in FC-1.

Level	Filter	Kernel	Parameter	Parameter Size
conv1	32	3*3*1	288	36B
conv2	32	3*3*32	9216	1152B
fc1	32	16*45*32	737280	92160B
fc2	—	—	1024	128B

fc3	—	—	192	24B

Table.1 The Detail Information of Parameter

PEs send target parameters' loaf addresses to shared storage structure. These addresses are line addresses to BRAM blocks so BRAM blocks can access data with no delay. Parameters are stored along the third dimension in BRAM, which is consistent to the input data's organization method of FC-1 layer. Shared-parameter data is sent to all PEs by broadcasting.

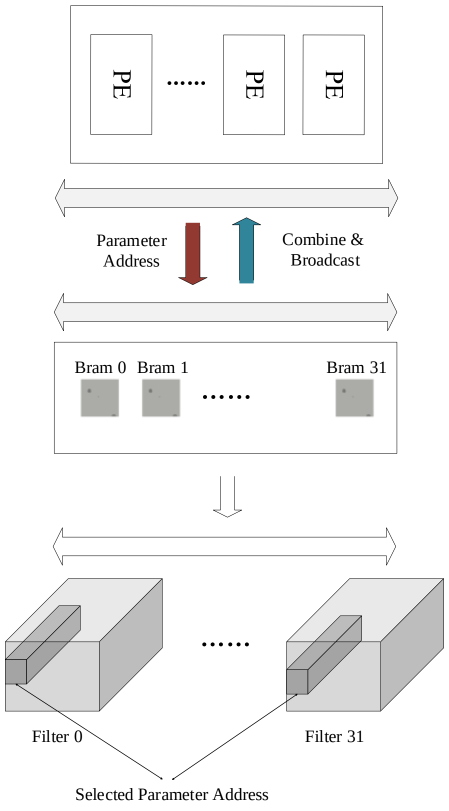


Fig.6 FC-1's Parameter Sharing Structure

As to other binary weight, we can store them directly on the chip. Similar to the shared parameter structure, each BRAM block is responsible for one filter. When the vector processing unit in Fig 7 starts computing, BRAM blocks directly send related parameters to the unit and these binary data then compute with activation. The vector unit will either keep original activation value or reverse it due to the input binary weight (indicating +1 or -1), each channel will have 32 temporary results. All temporary data will then be put through parallel adder tree to compute for result, each channel will get one valid data every computation and 32 valid data for 32 channels.

Prior batch-normalization parameters also need to be storage on chip. To pursue quantization accuracy, these data bitwise varies and thus it is better to storage batch-normalization parameter in flexible register.

4.2 Bitwise Expansion

Inside one convolution or full-connected layer, the input fix-point feature data is usually under normal distribution and varies in relatively small range, however, after MAC, the data's variance can be huge and irregular (Wei Z, et al, 2017). Batch-normalization relies on these middle results' mean and variance to improve data distribution, which is vital to final accuracy. If we simply apply the same data format as the input feature to data in batch-normalization step, it will lead unnecessary loss to final result.

To handle this situation, we introduce bitwise expansion method which gives extra decimal bitwise to both feature data and normalization parameter when computing. After DSP outputting multiply results, those extra decimal bitwise will be cut and the data will return to original input format.

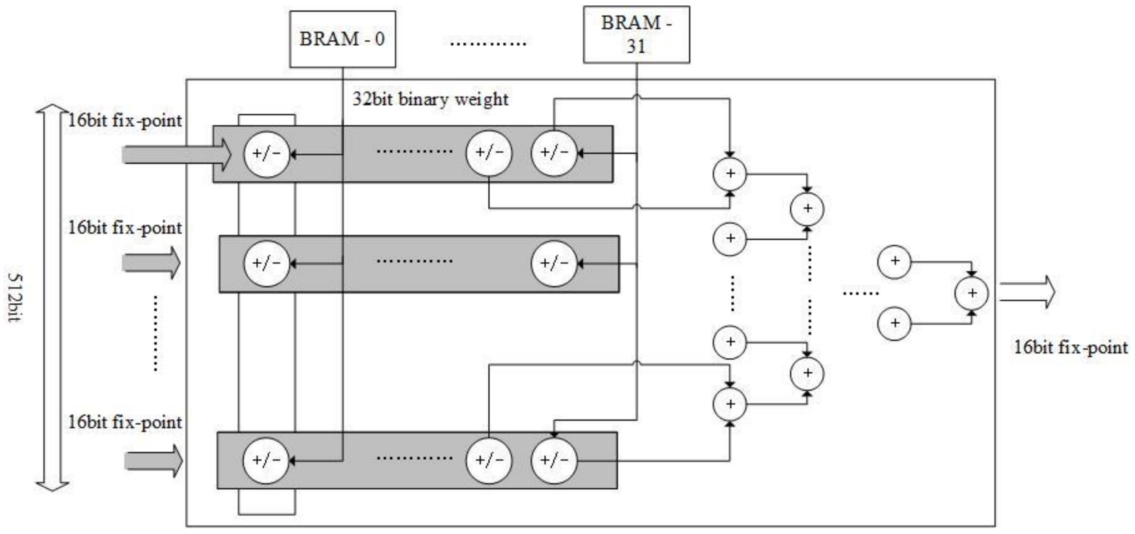


Fig.7 Vector Processing Unit

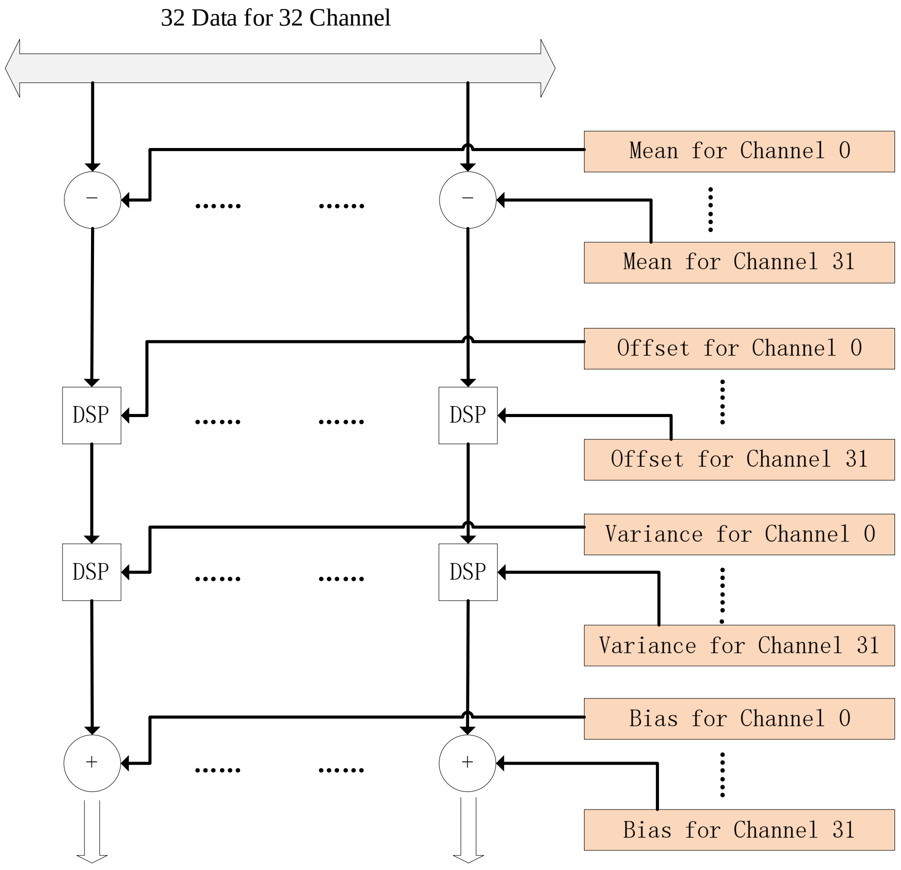


Fig.8 Batch-Normalization Unit

Similarly, middle results of MAC process have same needs for bitwise expansion. The different point is MAC results usually need more bitwise in integer part instead of decimal number. We apply expansion to both normalization step and accumulator in adder tree of MAC, it turns out that this method ensures computing accuracy on hardware.

4.3 Level-by-level Pipeline

The key point of level-by-level pipeline's implementation is to balance running periods between levels. In deep convolution neural networks such as VGG-16 and AlexNet (Simonyan K, et al, 2014; Krizhevsky A. et al, 2012), it is difficult to keep this balance due that as networks going deeper, deep levels will demand levels ahead to generate feature result at a faster speed, which is beyond current hardware's computing power limit.

The target neural network is shallow and tiny, so it is relatively easy to keep balance between two convolution layers. The data-fetching address generating strategy of Conv-1 layer is adjusted to cooperate with Conv-2 layer's computing mode. Also, we expand the scale of parallelism of Conv-1 layer's function module in order to generate one loaf of Conv-2 layer's input data in one level pipeline's beat.

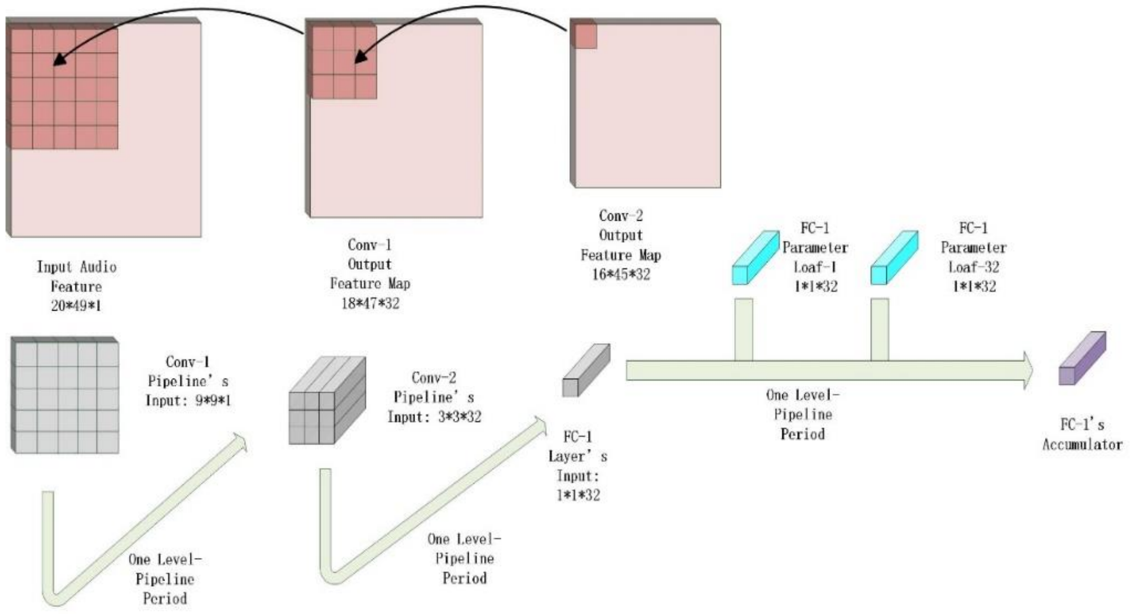
To produce one group of result in Conv-2 layers, this neural network has to compute one 3x3 slide window on Conv-1's output map and compute nine 3x3 slide windows from a 5x5 area on input audio feature. We expand the scale of Conv-1's computing array to make it generate nine 32x1 vectors in one macro pipeline period, and in next period these vectors will be sent to Conv-2 and generate one 32x1 vector for FC-1 layer. Conv-2's function is running in one macro period as well. FC-1 layer is the bottleneck of this accelerator, it is unbearable to keep balance this layer with two convolution layers for its huge computing resource utilization, but we can ensure FC-1 layer keep getting one vector in each macro period for accumulation.

By adapting pipeline on levels, target neural networks can be accelerated without putting between-levels results into DRAM and thus reducing memory accessing cost. In another word, we keep data stream in level pipeline from input audio feature to final predict results without stop. The accelerator only communicates with DRAM in fetching and final writing back. Inside each level's pipeline, we also divide all computing into some function parts like vector computing unit, normalization unit in pipeline method, which help to rise hardware running frequency.

5. Experiment

5.1 Quantized Model's Performance

We use Matlab-2018a's quantizer function to turn feature and batch-normalization parameters into fix-point format with saturate mode. In order to find the best bitwise allocation scheme, we run a couple of quantization experiments and compare their accuracy performance.



Input Audio Feature 20*49*1
Conv-1 Output Feature Map 18*47*32
Conv-2 Output Feature Map 16*45*32
FC-1 Parameter Loaf-1 1*1*32
FC-1 Parameter Loaf-32 1*1*32
Conv-1 Pipeline' s Input: 9*9*1
Conv-2 Pipeline' s Input: 3*3*32
FC-1 Layer' s Input: 1*1*32
One Level-Pipeline Period
One Level-Pipeline Period
One Level-Pipeline Period
FC-1' s Accumulator

Fig. 9 Balancing Conv-1 and Conv-2

To run neural networks on our specific hardware platform, the data format must fit the hardware design code. We set the bitwise of all kinds of data must range in 16 bits to 32 bits, which is the boundary condition for design space searching. The actual experiments show that the integer bitwise of middle results usually needs 8~12 bits while normalization parameter needs 20~21 bits for integer part. This result can help us to determine the upper bound of decimal bitwise. In deep neural networks, activation and middle results are relatively unsensitive to numerical precision, so we do not have to devote too much work on these data's decimal bitwise. On the other hand, normalization parameter needs more data format accuracy than middle results, so in principle, we give it decimal bitwise no less than middle result.

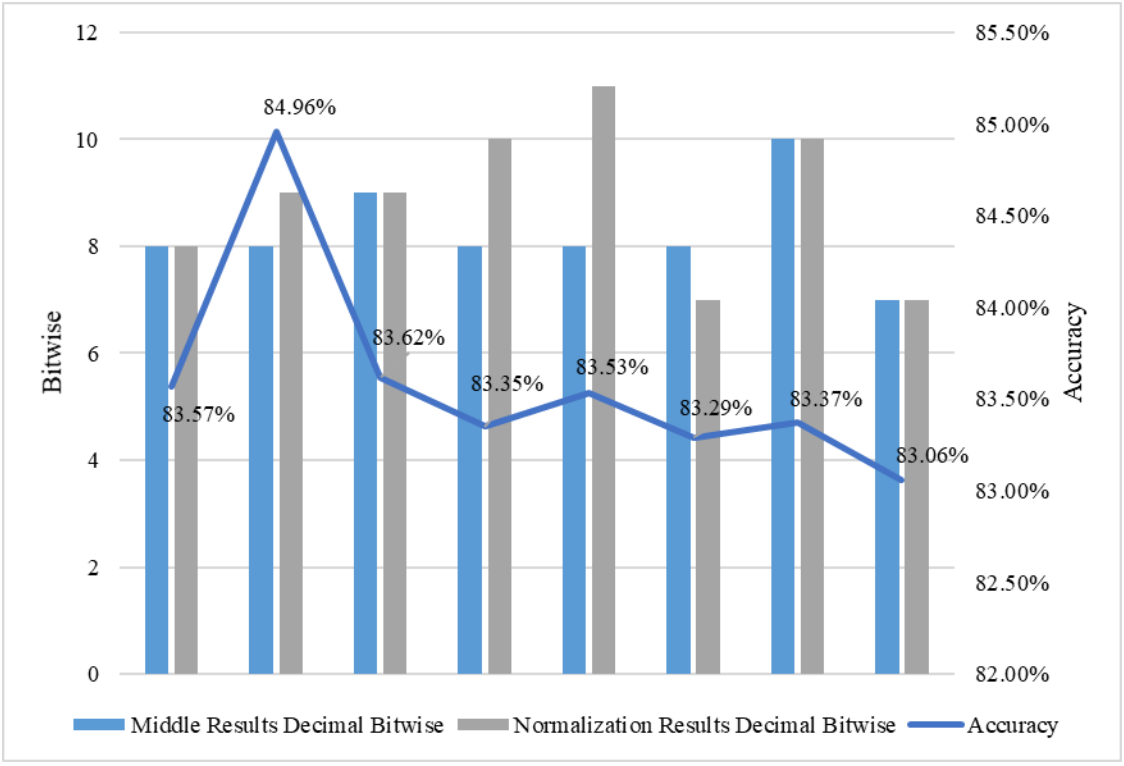


Fig.10 Accuracy Experiment Result

The result in Fig 10 shows that when middle data apply 8-bit decimal bitwise and normalization parameter apply 9-bit decimal bitwise the model will achieve best accuracy. We also implement the experiment that middle result owns more accuracy space than normalization data's which violates our searching principle, the final result supports our idea effectively.

There is one thing need to be noticed, hardware cannot handle division operation as easy as Matlab code, so we turn variance's division in batch-normalization into reciprocal multiplication and expand decimal bit-width to ensure accuracy.

We run our non-quantization version of Matlab code on Intel core i7-8700K with and without multi-thread accelerating library. We divide whole program into several function segments and test their running time. Table 2 shows that when ignore the MFCC segment, the second convolution layer is performance bottleneck on CPU platform and has vital effect to the model, which is corresponded to the large computing scale of Conv-2 layer.

8700K without Parallelism

Function Segment	Time (Seconds)	Ration in Total Time
Data Loading & Pre-process	9.413601	7.40%
Conv-1	3.084229	2.43%
Conv-2	114.582	90.14%
FC-1	0.023377	<1%
FC-2	0.001659	<1%
FC-3	0.001589	<1%
Total	127.1163	100%
	8700K with Parallelism	
Function Segment	Time (Seconds)	Ration in Total Time
Data Loading & Pre-process	9.804942	19.74%
Conv-1	1.803836	3.63%
Conv-2	38.02353	76.56%
FC-1	0.025257	<1%
FC-2	0.001316	<1%
FC-3	0.000981	<1%
Total	49.66721	100%

Table.2 Running Time on CPU Platform

5.2 Accelerator's Performance

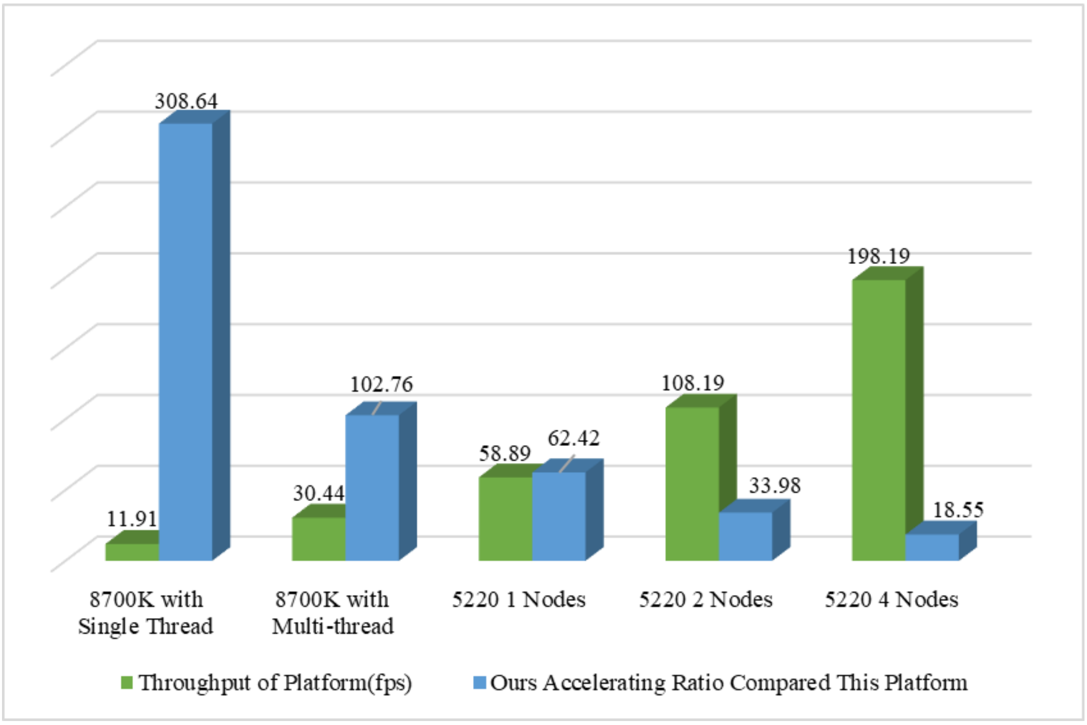


Fig. 11 Performance Compared with Multiple CPU Platforms

We run this sound classification model on intel Core i7-8700K (3.7GHz, 6 cores, 95W) with single thread, intel Core i7-8700K with multi-thread and multi-node intel Xeon 5220 (2.2GHz, 18 cores, 125W) with Matlab distributed parallel library, the whole dataset contains 1512 audio files. We only test the running time of neural network's forwarding part for we do not implement the MFCC and data pre-process on FPGA. Fig 11 shows that compared to state-of-art CPU platform, our single PE version accelerator achieves 18~300x throughput speed up ratio. Table 2 shows that Conv-2 layer is the most time-costing function, however, by using balanced level-by-level pipeline, our accelerator can eliminate bottlenecks in the original neural networks and achieve excellent accelerating performance. Also, the data stream inside the pipeline reduce the DDR bus communication tremendously, eliminate the DRAM limit on CPU platform.

The multi-PE version of our accelerator is implemented on Xilinx Vivado 2018.3 and KU-115 FPGA platforms. We conduct the test of computing accuracy, speed-up ratio and power performance validating hardware design. To ensure reliability, we conduct extensive pressure test for up to two hours. The implement results are shown in Table 4.

We compute the energy efficiency of state-of-art CPU platforms and ours. The results in Table 5 shows that by customized circuit design and replacing floating data with fix-point data, our accelerator has great energy efficiency improvement on this sound classification task.

PE	LUT Utilization	FF Utilization	BRAM Utilization	DSP Utilization	Frequency
1	25%	14%	9%	12%	150MHz
2	47%	26%	10%	23%	150MHz

Table.4 Accelerator’s Implementation Results

	8700K	5220-1Node	5220-2 Node	5220-4 Node	1PE BWN	2PE BWN
Perf. Per Watt (fps/W)	0.23	0.47	0.43	0.4	471.63	753.41
Power (W)	95W	125W	250W	500W	7.794W	9.758W

Table.5 Energy Efficiency on Different Platforms

In Table 6, we choose two typical FPGA accelerators based on RNN models and compare their performance with our BWN accelerator. It turns out that when working frequency is similar, our work has great advantages on power, peak performance, throughput and energy efficiency over RNN accelerators. We take advantage of Deep convolution neural network’s being easily parallelized and obtain excellent performance. When processing sound classification task, quantized acoustic deep convolution neural network and customized accelerator can meet the need of high performance and energy-efficient at the same time.

Accelerator	Sicheng L, et al	Andre C, et al	1 PE BWN	2 PE BWN
Platform	FPGA	FPGA	FPGA	FPGA
Frequency	150MHz	142MHz	150MHz	150MHz
Power	25W	1.942W	7.794W	9.758W
Peak Perf.	9.6GOPS	0.266GOPS	23.85GOPS	47.7GOPS
Throughput	65.85fps	1073fps	3675.9fps	7351.8fps
GOPS/W	0.38	0.1549	2.37	4.89
FPS/W	2.63	558.85	471.63	753.41

Table.6 Comparison with Typical FPGA-Based RNN Accelerator

6. Conclusion

To pursue power-efficiency, high-performance computing and model accuracy, we first optimize a sound classification algorithm based on deep convolution neural network. By quantization method, the activation size is reduced sharply and time-consuming floating computation is replaced by faster fix-point computation. Our accelerator design then focuses on parameter-shared storage structure, bitwise expansion and balanced level-pipeline. With the combination of deep convolution neural network quantization and customized circuit design, we bring out a high-throughput, energy-efficient and high-performance sound classification computing platform based on FPGA. Compared with current state-of-art CPU platform and other RNN-based acoustic task accelerator, our hardware design has great advantages on both computing performance and power efficiency. We implement our design on Xilinx FPGA, it turns out this accelerator is a reliable and high-performance intelligent computing platform.

Acknowledgment

This work is supported by National Science and Technology Major Project 2018ZX01028101.

Compliance with ethical standards

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Reference

Geoffrey H, Li D, Dong Y, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition [J]. IEEE Signal Processing Magazine, 2012, 12:82-98.

Wan H , Guo S , Yin K , et al. CTS-LSTM: LSTM-based neural networks for correlated time series prediction[J]. Knowledge Based Systems, 2019, 191.

Tom S, Vaibhava G. Advances in Very Deep Convolutional Neural Networks for LVCSR[C/OL]. arXiv:1604.01792v2[cs.CL].

Muckenhirn H , Magimai.-Doss M , Marcell S . [IEEE ICASSP 2018 - 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) - Calgary, AB (2018.4.15-2018.4.20)] 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) - Towards Directly Modeling Raw Speech Signal for Speaker Verification Using CNNS[C]// 2018:4884-4888.

Palaz D , Magimai.-Doss M , Collobert R . Convolutional neural networks-based continuous speech recognition using raw speech signal[C]// 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015.

Pakyurek M, Atmis M, Kulac S, et al. Extraction of Novel Features Based on Histograms of MFCCs Used in Emotion Classification from Generated Original Speech Dataset[J]. Electronics & Electrical Engineering, 2020, 26:46-51.

Jyrki Kivinen M K W . Exponentiated Gradient versus Gradient Descent for Linear Predictors[J]. Information and Computation, 1997, 132(1):1-63.

Perkins S , Lacker K , Theiler J . Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space[J]. Journal of Machine Learning Research, 2003, 3(3):1333-1356.

Liu S , Pattabiraman K , Moscibroda T , et al. Flikker: Saving DRAM Refresh-power through Critical Data Partitioning[J]. Computer architecture news, 2011, 39(1):p.213-224.

Han S , Mao H , Dally W J . Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[C]// ICLR. 2016.

Dundar G , Rose K . The effects of quantization on multilayer neural networks[J]. IEEE Transactions on Neural Networks, 1995, 6(6):1446-1451.

Guo P , Ma H , Chen R , et al. A High-Efficiency FPGA-Based Accelerator for Binarized Neural Networks[J]. Journal of Circuits System & Computers, 2019.

Liang S , Yin S , Liu L , et al. FP-BNN: Binarized neural network on FPGA[J]. Neurocomputing, 2018, 275(JAN.31):1072-1086.

Conti F , Schiavone P D , Benini L . XNOR Neural Engine: A Hardware Accelerator IP for 21.6-fJ/op Binary Neural Network Inference[J]. IEEE Transactions on Computer Aided Design of Integrated Circuits & Systems, 2018, 37(11):2940-2951.

Michael D, Ashish K, David R. Nvidia's Xavier Soc[C]. IEEE Hot Chips 2018.

Nvidia. Deep Learning Accelerator[C]. IEEE Hot Chips 2018.

Hernandez P. Microsoft Uses Intel FPGAs for Smarter Bing Searches[J]. Technology, Knowledge and Learning, 2018.

Gwennap L . Microsoft Brainwave Uses FPGAs[J]. Microprocessor Report, 2017, 31(11):25-27.

Turan F , Roy S S , Verbauwhede I . HEAWS: An Accelerator for Homomorphic Encryption on the Amazon AWS FPGA[J]. IEEE Transactions on Computers, 2020, PP(99):1-1.

Bo L, Hai Q, Yu G, et al. EERA-ASR: An Energy-Efficient Reconfigurable Architecture for Automatic Speech Recognition with Hybrid DNN and Approximate Computing[J]. IEEE ACCESS, 2018, 6:52227-52237.

Matthieu C, Itay H, Daniel S, et al. Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1[C/OL]. arXiv:1602.02830v3[cs.LG].

Jacob B , Kligys S , Chen B , et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference[J]. 2017.

Xu Y , Wang Y , Zhou A , et al. Deep Neural Network Compression with Single and Multiple Level Quantization[J]. 2018.

Santurkar S , Tsipras D , Ilyas A , et al. How Does Batch Normalization Help Optimization [J]. 2018.

Liu M , Wu W , Gu Z , et al. Deep Learning Based on Batch Normalization for P300 Signal Detection[J]. Neurocomputing, 2017:S0925231217314601.

Giri E P , Fanany M I , Arymurthy A M , et al. Ischemic Stroke Identification Based on EEG and EOG using 1D Convolutional Neural Network and Batch Normalization[C]// ICACSIS. IEEE, 2016.

Cheung K , Schultz S R , Luk W . A Large-Scale Spiking Neural Network Accelerator for FPGA Systems[C]// Proceedings of the 22nd international conference on Artificial Neural Networks and Machine Learning - Volume Part I. Springer, Berlin, Heidelberg, 2012.

Alessandro A , Hesham M , Enrico C , et al. NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps[J]. IEEE Transactions on Neural Networks & Learning Systems, 2018:1-13.

Chen T , Du Z , Sun N , et al. A High-Throughput Neural Network Accelerator[J]. IEEE Micro, 2015, 35(3):24-32.

Wei Z , Jingyi Q , Renbiao W . Straight Convolutional Neural Networks Algorithm Based on Batch Normalization for Image Classification[J]. Journal of Computer-Aided Design & Computer Graphics, 2017.

Simonyan K , Zisserman A . Very Deep Convolutional Networks for

Large-Scale Image Recognition[J]. Computer Science, 2014.

Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[C]// NIPS. Curran Associates Inc. 2012.

Sicheng L, Chunpeng W, Hai L et al. FPGA Acceleration of Recurrent Neural Network Based Language Model[C]. 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Mahcines, pp. 111-118.

Andre C, Berin M, Eugenio C. Recurrent Neural Networks Hardware Implementation on FPGA[C/OL]. arXiv: 1511.05552v4[cs.NE]