A Reconfigurable and High Peformance PYNQ-based Accelerator for Small Convolutional Neural Networks

***Abstract*—Convolutional Neural Networks (CNN) is the most one important algorithms of machine Learning. In recent years, it has been widely applied in machine vision and Nature Language Processing. However, the percent software implementation solutions are based multi-core CPU, GPUs and custom ASICs in Datacenter environment, so it is a large power consumption and large size shortcomings. With the development of Hardware technology, there are many embedded hardware also can implement the CNN algorithm, such as IOT, Edge Device embedded chip. This work proposes to build a high performance and reconfigurable heterogeneous computing accelerator, using Xilinx PYNQ-Z2(ZYNQ 7020 CLG400-2) Platform. To implement CNN algorithms shortly and resource saving on the chip, so each layer work in pipeline structure and ping pong buffer structure to increase throughput. On the communication methodology which can find AXI DMA bus strategy to improve data transmission bandwidth FPGA and ARM CPU. And more, it uses ARM NEON FPUs to achieve other computation. In addition, on the resources optimized which use Verilog language to program and it was visualized in jupyter notebooks IDE. In this case study, we implement Lenet-5[1] on the platform, it can achieve a peek performance 138.9GOPs and power consumption with 1.863W under the 100Mhz clock, and Processing a picture requires only 135us.Finally it can support reconfigurable computation pattern to meet different CNN's Architecture.**

Keywords—CNN, configurable, FPGA, High Performance

I. INTRODUCTION

In recent years, Convolutional Neural Networks (CNN) has become one of most significate algorithms in Deep Learning field. It has been successfully applied in many computer vision, speech recognition, face recognition, inspired by the nervous system in human brain. These CNN algorithms are high accuracy and good performance in above task.

All know that running a CNN mode requires a lot of calculation power to simulate neuronal manipulation and data access in real world and also become more large and deep. So many ANN algorithms are implemented one Multi-Core CPUS, GPUS, and custom ASICs in workstation and severs. Also the CNN Network acceleration Scheme based on FPGA has become a hotspot for scientists and industries over the past few years, as evidenced by Microsoft's Catapult project [2], Intel's acquisition of Altera [3], and most recently Amazon's inclusion of FPGAs as part of their Amazon Web Services [4]. FPGA is especially suitable as the hardware accelerator owing to its flexibility and efficiency. However, there are still many challenges as below. First of all, the chip process technology is becoming more and more slow, we are live in the post-More law era, it limits the development of chip technology to get balance power and efficiency. Fortunately, some academics and companies have proposed heterogeneous computing and reconfigurable computation patterns to solve these problems such as Intel and IBM, the Heterogeneous computing Technology combined with CPU and FPGA or other computation Units integrated on a piece of silicon(SOC) , compared with traditional hardware it can reduce power consumption and increase bandwidth; for reconfigurable computing technology, the deep neural architecture(DNA) has presented by reference[5], with reconfigurable computation patterns for different models. The computation pattern comprises a data reuse pattern and a convolution mapping method. For massive and different layer sizes, DNA reconfigures its data paths to support a hybrid data reuse pattern, which reduces total energy consumption by 5.9~8.4 times over conventional methods. And another challenge is to be less friendly to hardware developers, which means they spend a lot of time working on hardware debugging and data visualization. To be friendly, some FPGA Manufacturers Proposed a new FPGA application development Method which is effective in Python and HLS for users, such as PYNQ[6].

The key contributions of this work are as below:

1. A reconfigurable and high performance, low power hardware Architecture is proposed for CNN accelerator with Heterogeneous style.
2. There are some Hardware optimization methods have presented. Each layer works in Pipeline to reduce latency and chip buffer resources like BRAM, increase data transmission bandwidth on the chip, also data synchronization between C2 and FC1 using ping-pong buffer technology on programming logic(PL) side. And more the AXI DMA bus events are applied to data transmission PL and Processor System(PL), increase communication bandwidth.
3. A state-of-the-art CNN, Lenet-5[1] is implemented on PYNQ-Z2 board which is high productivity tools for system on chip(SOC) heterogeneous. It's Benefit As above hardware optimization methods, it can achieve a peek performance with 138.9GOPS and 1.875W Energy consumption under the 100Mhz clock. and recognizes a 30*30 image which from MNIST dataset only 135us. It is 99.27 % for training recognition accuracy in this CNN model.

The rest of paper is organized as follows. Section II introduces some related works. In Section III as a background to describe Lenet-5 model for FPGA. Section IV is given a system and hardware architecture, It describes in detail the optimization methods and strategies for each layer of FPGA implements. And Section V describes our experimental results and detail, has mode comparation our implementation between and previous experiment in this Section. Section VI concluded this paper.

II. RELATED WORK

To design an artificial neural network accelerator. Many researchers mainly focus on three respects. 1) How to achieve a balance between power efficiency and performance.2) How to use the optimization strategy to increase the throughput rate of data and reduce the memory demand of a single network,

thus speeding up the inference calculation of ANN.3) How to use the architecture mode to adapt to the changing algorithm model, improve the reuse efficiency of hardware resources.

Artificial neural networks need a lot of computation to simulate the behavior of neurons in the process of data processing, and the von Neumann computer structure cannot meet this requirement because it cannot achieve energy efficiency balance. For this reason, many researchers have made a lot of efforts to solve this contradiction, much work has been done such as [7, 8, 9, 10, 11]. References [12] and [13] use parallelism in feature maps and convolution operations. References [11] and [13] have optimized both the bandwidth and the computation, which achieve a good balance. Interestingly, they use reconfigurable computing technology to improve resource utilization, adapt to different network models, and succeed in energy efficiency balancing, deployed on embedded computing devices and Edge computing devices in reference[5]. In this work, we design a high performance using reconfigurable pattern for CNN based on PYNQ to solve above conflict, and Improve development efficiency. The remainder of the article will discuss this architecture in detail, including refactoring mode, pipelining, system implementation, memory optimization, and communication bandwidth improvement methods on PYNQ(ARM+FPGA).

III. BACKGROUND

In this section, we will briefly introduce a typical CNN network structure which is LENET5, and then detail the optimization on the FPGA.

A. Primer on the Lenet5 Structure

All know all that Lenet5 is one typical mode of CNN structure in Fig 1 . It is composed of several layers and proposed by[1]. It is mainly composed of convolution layer, pool layer, full connected layer and classification layer. Each layer is transmitted by neighboring neurons, and when given input data, the layer will output the corresponding eigenvector. For FPGA, the common solution is to implement the inference calculation of CNN Network, it put the trained weights into memory, multiply and add the operation to simulate the neuron forward calculation.

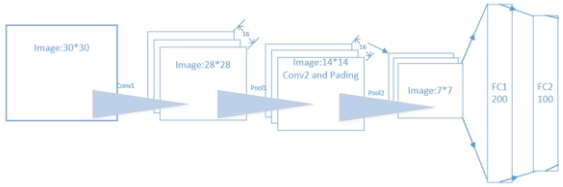

Fig.1. Lenet-5 model on FPGA implement

B. CONV layer

$$O_{a,b}^{mo} = f\left(\sum_{mi \in A_{mo}} \left(\beta^{mi,mo} + \sum_{i=0}^{K_x-1}\sum_{j=0}^{K_y-1} \omega_{i,j}^{mi,mo} \times I_{a\,S_x+i,b\,S_y+j}^{mi}\right)\right) \quad (1)$$

Mathematically, the convolution layer can be calculated by the Equation (1) to describe a two-dimensional convolution model, m and o represent the size of the image input, ω represents the weight of the convolution kernel, β threshold value. Given a frame of image, set the step size of the convolution, which will result in the eigenvector.

C. POOL layer

$$P_{a,b}^{mo} = \max_{0 \le i \le K_x, 0 \le j \le K_y} \left(I_{a+i,b+j}^{mi}\right) \quad (2)$$

Equation(2) describes the mathematical model of the pooled layer, which takes the maximum value of the pixel in a given region(m, o) as the output(P).

D. Full connected layer

$$a_j^{[l]} = \sigma\left(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}\right) \quad (3)$$

This is a Feedforward neural network base model, as shown in Equation(3). w represents the weight matrix, α represents the input data matrix, and b represents the threshold value.

E. Relu function

$$\mathrm{relu}(x) = \max(x, 0) \quad (4)$$

In this paper, we use relu[14] function set as a active function as Equation(4), because it is hardware-friendly, a comparison selector can be used to describe such formulas on an FPGA, and there is no mathematical reason for gradient descent to slow down.

F. Configuration for each layer in lenet-5

According to Fig.1, we set this CNN model parameters as in table I, Conventions are as follows. C(n) is convolution layer, P(n) is Pooling layer, FC(n) is full connected layer.

TABLE I. PARAMETERS CONFIGURATION OF LENET-5

Layer	*Core*[a]	*width*	*$size_{in}$*	*$Size_{out}$*	*$Size_{core}$*	*stride*	*Weights*
C1	16	8	28	28	3*3	1	144
P1	16	16	28	14	-	-	-
C2	16	16	14	14	3*3	1	144
P2	16	16	14	7		-	-
FC1	2	16	1*1	1*1	-	-	156800
FC2	2	32	1*1	1*1	-	-	2000

[a] Shows the number of kernels for C1 and C2, FC1 and FC2 are implemented in two parallel kernels.

IV. HARDWARE AND SYSTEM ARCHITECTURE

In this section. The system architecture is proposed for our accelerator firstly. Secondly there is a state-of-the-art hardware architecture to present. We introduce some hardware optimization strategies which include every basic computation module for each layer thirdly.

System Architecture

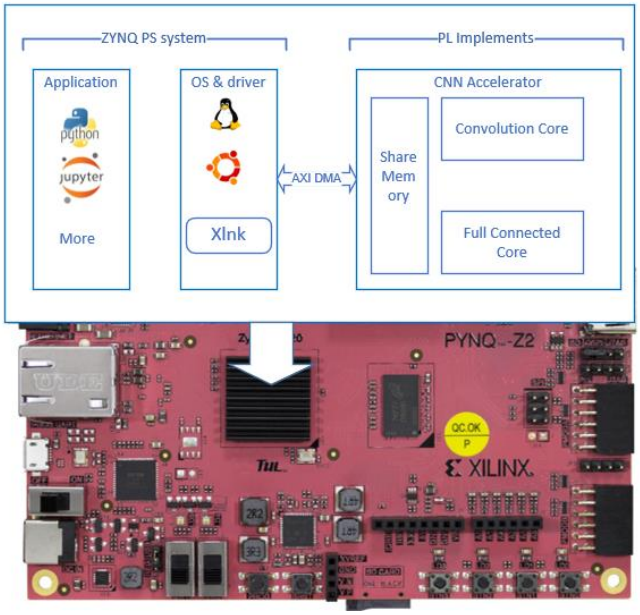

Fig.2. sytem architecture note:xlnk is driver of DMA of Xilinx ZYNQ for linux kernel.

As illustrated in Fig.5. the whole system run on PYNQ-Z2[18] board, The main processor is a SOC chip, which is composed of FPGA (PL) +ARM (PS) of a heterogeneous microcomputer system. The PL side will perform our design of the CNN Accelerator, while the PS side completes the control of the PL side and runs the traditional application program.the interface is AXI DMA bettwen PS and PL, Since then, algorithm researchers have been able to quickly deploy their algorithms in a specific hardware environment.

Hardware Architecture

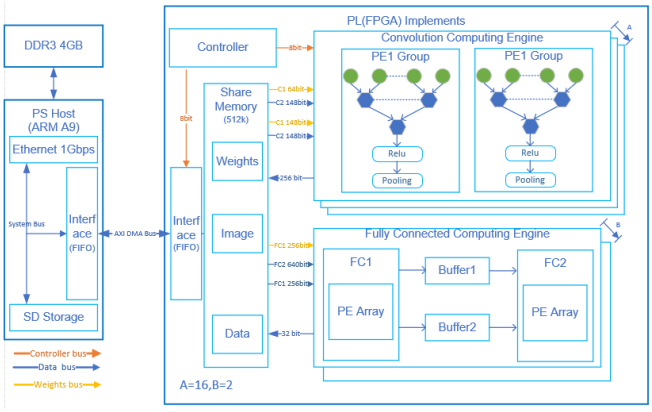

Fig.3. Hardware Architecture

To get a workload balance power efficiency and performance, we design a hardware architecture for our convolutional neural network accelerator as above picture. Inspired by reconfigurable computing technology [5], here take the lenet-5 model as an example.

Overall, this hardware architecture can be divided into PL and PS two parts, PS for the implementation of visual applications and providing user programming interface. The Lenet-5 model is implemented on PL side. C1 and C2, PL1 and PL2 run in convolution computing engine, this engine consists of 16 groups of parallel PE1 and PE2 clusters, totaling 256 PE parallel units. Each PE can support input 9 weights and image data, and same time, every PE unit contains an activation layer. Rest of FC1 and FC2 are implemented in Full connection computing engine, it has another 2*16 PE parallel core and 2*20 PE parallel units, It is obvious that the Ping-pong caching mechanism is used between C2 and FC1 and FC2 to synchronize data, which reduces memory consumption on the chip and increases performance consequently. Moreover, the image data, weights data, and output data are stored in share memory using Block RAM of FPGA, which are double port mode. For PS-PL communication, we use AXI Direct memory Access (DMA) bus to improve data transmission speed which is a feature of computer systems that allows certain hardware subsystems to access main system memory (random-access memory), independent of the central processing unit. In particular, it is important to note that all of these modules are controlled by their respective control units, and the control unit is implemented with a finite state machine (FSM), which is a common control strategy in the circuit.

Basic Components Optimization strategy

CONV and POOL layer

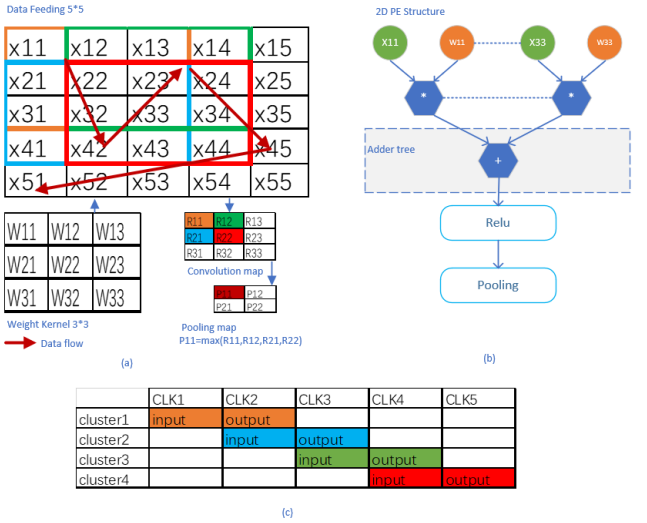

Fig.4. convolution and pool layer Optimization strategy. (a) data flow for C1 and C2,(b) 2-dimensional Processing Element(PE) structure to implement convolution operation with 3*3 kernel and data input 5*5 feeding,(c) Pipeline operation in C1,C2, P1,and P2

1) convolution layer data reuse: for the convolution calculation, it has the same characteristics of local input data, that is, the last convolution input data may be reused in the next convolution calculation, benefiting from the DNA reconfigurable computing [5], we propose different data stream input method with other papers as shown in Fig.4(a). Orange represents the first clock input data, blue represents the second clock input data, green represents the third clock input data, red represents the fourth clock input data, the data using the FIFO structure to cache, so that the calculation and storage can be executed asynchronously, And in the assembly line working mode 2) pool layer input data optimization: in the previous paper work, the convolution layer and the pooling layer between the connection between the need to

consume a large amount of memory resources to cache the convolutional layer output data, and we after the convolution layer input data sorted. The results of convolution calculation can be used as the input data of the pooling layer, and only a few register resources are needed to cache the data between the two layers.

Benefit from the above optimization strategy, so that each clock cycle can be the data convolution and pooling calculation, greatly improve the data throughput and reduce the device time delay.

c)Full connected layer

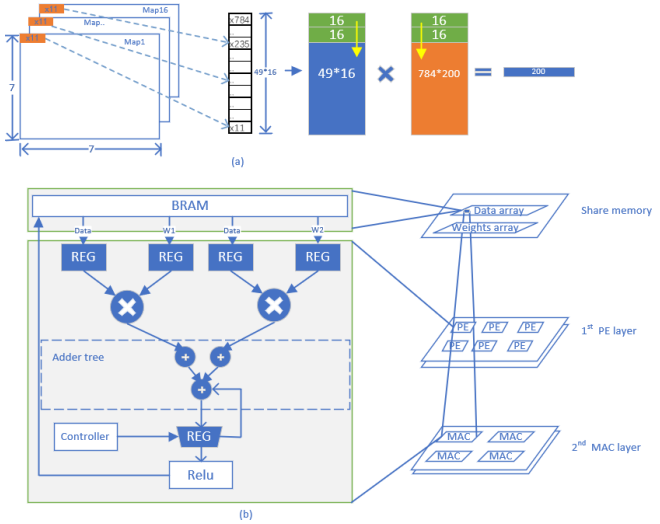

Fig.5.FC Layer Structure.a) Multidimensional to one-dimensional mapping

b) PE architecture for FC layer

As shown in Fig.5(a), shows the C2 and FC1 connection modes, usually we need to expand the output eigenvector of the C2 into a one-dimensional vector and then input to the FC1 layer for calculation. However, we are different from the previous optimization strategy [16], in the C2 output, but also can be FC1 layer calculation, that is, when the parallel convolution core to complete a data calculation, first cached in a one-dimensional vector, and input to the FC1 calculation unit to complete a MAC calculation, in our experiment, so iterate 49 times to simulate the amount of computation required for a FC1 neuron operation. Based on such an optimization strategy, the order of weights entered differs from the order on the host. As shown in Fig.5(b), the architecture of an optimized version of FC1 and FC2 based on a common DNA architecture is given, which differs from the convolution layer of the appeal by adding an accumulator inside the PE unit to accumulate the previously computed results, and when it iterates several times the input activation unit completes the calculation of the neuron.

Comprehensive appeals, which can increase the parallelism of the computations and thus reduce the delay time. In addition, FC2 also uses the same architecture to complete the calculation, the input weight matrix size is 200*10, the input data matrix size is 200, here is no longer described in detail.

V. EXPERIMENTAL RESULTS

We first use the Jupyter_notbook on PYNQ and Xilinx Vivado (2018.2) HLS Development Kit to develop according to the architecture shown in Fig.2 and Fig.3. Generate an FPGA-based CNN model bit stream file in vivado (2018.2) using the Verilog language and then import it as overlay into Jupyter_notbook, showing the results of the recognition in it, as shown in figure Fig.6.

Thanks to the simplicity of the Verilog language, the generation of bit-stream redundancy information, and combined with the results of the previous discussion, we can use the best optimization strategy on FPGA to simulate the calculation of CNN model, including data reuse, pipeline computation, asynchronous execution and high-bandwidth communication method , its resource optimization results as shown in TABLE II, in terms of communication, we have also made a lot of efforts to achieve the 50MB/s speed to realize the interaction of PS and PL with DMA, the identification of a single picture only takes 135us of time, including 120us of FPGA execution time and 15us of data communication time.

TABLE II. RESOURCE UTILIZATION

	FPGA on Xilinx ZYNQ 7020 CLG400-2				
	LUT	***FF***	***DSP48***	***BRAM***	***LUTBRAM***
Avialable	53200	106400	220	140	17400
Utilization	33200	33785	216	128.50	1587
Percent(%)	60.53	31.75	91.79	98.18	9.12

TABLE III. COMPARISON WITH OTHER IMPLEMENTATIONS BASED-ON SMALL CNN

project	***Platform***	***Processing Image/ms***	***POWER(W)***	***Parameter numbers***	***Clock***
Tesnor Flow on CPU	I7-8700k	1.2	95	159088	4.7Ghz
FPGA SqueezeNet[15]	Xilinx-VC709	3.6	27.7	N/A	110 Mhz
FPGA CNN[17]	VERTEX-5 XC5	0.272	1.584	184974	75Mhz
This work	ZYNQ 7020	0.135	1.865	159088	100 Mhz

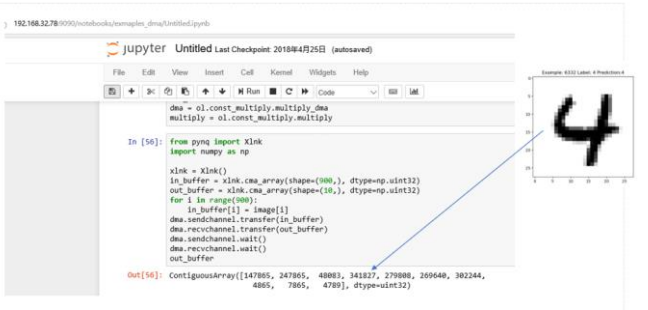

Fig.6 Implementation Platform and the result display.

The experiment results are illustrated in Fig.6. here we need to host screen to visit results via jupyter_notbook. It shows the computation result in hardware FPGA for character 4, max value is 341827 of its.

VI. CONCLUSION

In this paper, we implemented a simple CNN model with our reconfigurable architecture pattern to achieve a balance power and efficiency one the PYNQ platform, its training accuracy of this CNN model on the MNIST [19] dataset is 99.27%.

The optimization method proposed above effectively solves the above contradiction, can be 135us speed recognition input with 30*30 image data, power consumption is only 1.863W, so this model can be allowed to run in the embedded system, and this framework can increase the productivity of developers. In the future, we will continue to use the optimization of this architecture to enable a large-scale network to execute on the FPGA.

ACKNOWLEDGMENT

This work is supported by National Natural Science Sichuan University of China 610065. We also appreciate careful to our works.

REFERENCES

[1] Yann LeCun, L.Bottou,Yoshua Bengio, and Patrick Haffner, “Gradient-Based Learning Applied to Document Recognition,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 1-7, Nov 1998.

[2] A. Putnam et al., “A reconfigurable fabric for accelerating large-scale datacenter services,” in Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA). IEEE Press, June 2014.

[3] “Intelcompletesacquisitionofaltera,”https://newsroom.intel.com/newsreleases/intel-completes-acquisition-of-altera.

[4] “Amazon EC2 F1 instances,” https://aws.amazon.com/ec2/instancetypes.

[5] Fengbin Tu, Shouyi Yin, Peng Ouyang, Shibin Tang. “Deep Convolutional Neural Network Architecture With Reconfigurable Computation Patterns,” IEEE Press.Aug 2017

[6] Xilinx, “PYNQ: Python productivity for zynq,” 2018. [Online]. Available: http://www.pynq.io

[7] C. Zhang and P. Li, “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in ACM International Symposium on Field-Programmable Gate Arrays, 2015, pp. 161-170.

[8] S. Chakradhar and M. Sankaradas, “A dynamically configurable coprocessor for convolutional neural networks,” in ACM Sigarch Computer Architecture News, 2010, 38(3), pp. 247-257.

[9] N. Suda and V. Chandra, “Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks,” in ACM International Symposium on Field-Programmable Gate Arrays, 2016.

[10] M. Sankaradas and V. Jakkula, “A Massively Parallel Coprocessor for Convolutional Neural Networks,” in International Conference on Application-specific Systems, 2009, pp. 53-60.

[11] T. Chen and Z. Du, “DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning,” in ACM Sigplan Notices, 2014, 49 (4), pp. 269-284.

[12] Z. Du and R. Fasthuber, “ShiDianNao: Shifting vision processing closer to the sensor,” in ACM International Symposium Computer Architecture (ISCA), 2015.

[13] Y. Chen and Y. Luo “DaDianNao: A Machine-Learning Supercomputer,” in ACM International Symposium on Microarchitecture, 2014, pp. 609-622.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Neural Information Processing Systems, 2012, 25 (2), pp. 1097-1105.

[15] Chao Huang, Siyu Ni, Gengsheng Chen, “ A Layer--based Structured Design of CNN on FPGA”, Proceedings of 2017 IEEE 12th International Conference on ASIC.pp.1062

[16] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, Lingli Wang, “A High Performance FPGA-based Accelerator for Large-Scale Convolutional Neural Networks,” 2016 26th International Conference on Field Programmable Logic and Applications (FPL). Aug 2016.

[17] Zijian Yu, De M, “FPGA-based Accelerator for Convolutional Neural Network,”Comuter Engineering in China. pp1-7.Jan 2017.

[18] Andrew G. Schmidt, Gabriel Weisz, and Matthew French “Evaluating Rapid Application Development with Python for Heterogeneous Processor-based FPGAs,” 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM).April 2017.

[19] http://yann.lecun.com/exdb/mnist/