[DL] A Survey of FPGA Based Neural Network Accelerator

KAIYUAN GUO, SHULIN ZENG, JINCHENG YU, YU WANG AND HUAZHONG YANG, Tsinghua University, China

Recent researches on neural network have shown great improvement in computer vision over traditional algorithms based on handcrafted features and models. Neural network is now greatly adopted in regions like image, speech and video recognition. But the great computation and storage complexity of neural network based algorithms poses great difficulty on its application. CPU platforms are hard to offer enough computation capacity. While GPU platforms are highly parallelized, the energy efficieny is low. The high energy cost of GPU causes problems for a wide application of neural network.

To address the above problems, various FPGA based hardware accelerators for neural networks have been proposed. Specialized hardware are designed to achieve high speed and low power neural network process. In this paper, we give an overview of previous work on neural network accelerators based on FPGA and summarize the main techniques used. Investigation from software to hardware, from circuit level to system level is carried out to complete analysis of FPGA based neural network accelerator design and serves as a guide to future work.

Additional Key Words and Phrases: FPGA, Neural Network

ACM Reference Format:
Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang AND Huazhong Yang. 2017. [DL] A Survey of FPGA Based Neural Network Accelerator. *ACM Trans. Reconfig. Technol. Syst.* 9, 4, Article 11 (December 2017), 4 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Recent research on Neural Network (NN) is showing great improvement over traditional algorithms in computer vision. Various network models, like convolutional neural network (CNN), recurrent neural network (RNN), have been proposed for image, video, and speech process. CNN [7] improves the top-5 image classification accuracy on ImageNet [8] dataset from 73.8% to 84.7% and further helps improve object detection [2] with its outstanding ability in feature extraction. RNN [3] achieves state-of-the-art word error rate on speech recognition. In general, NN features a high fitting ability to a wide range of pattern recognition problems. This makes NN a promising candidate to many artificial intelligence applications.

But the computation and storage complexity of NN models are high. The research on NN is also increasing the size of NN models. The largest neural network model for an 224×224 image classification requires upto 39 billion floating point operations (FLOP) and more than 500MB model parameters [9]. As the computation complexity is propotional to the input image size, processing images with higher resolutions may need more than 100 billion operations.

Traditional hardware platforms are not suitable for neural network process. A common CPU can perform 10-100G FLOP per second, and the power efficieny is usually below 1GOPs/W. So CPUs neither meet the high performance requirements in cloud applications nor the low power

Author's address: Kaiyuan Guo, Shulin Zeng, Jincheng Yu, Yu Wang AND Huazhong Yang, Tsinghua University, Tsinghua University, Beijing, Beijing, 100084, China, gky15@mails.tsinghua.edu.cn,yu-wang@mail.tsinghua.edu.cn.

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor, or affiliate of the United States government. As such, the United States government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for government purposes only.
© 2017 Association for Computing Machinery.
1936-7406/2017/12-ART11 $15.00
https://doi.org/0000001.0000001

requiremetns in mobile applications. In contrast, GPUs offer upto 10TOP/s peak performance and is a good choice for high performance neural network applications. Development frameworks like Caffe [6] and Tensorflow [1] also offers easy-to-use interfaces which makes GPU the first choice of neural network acceleration. But GPUs are power consuming and thus not suitable for mobile applications.

On the other hand, FPGA is becoming a candidate to implement energy efficient neural network accelerator. With a specific hardware design, FPGAs are able to implement high parallelism and make use of the properties of neural network computation to remove unecessary logic. Therefore FPGAs are possible to achieve higher energy efficieny compared with CPU and GPU.

But FPGA based accelerator designs are still faced with two problems:

- Current FPGAs usually support working frequency at 100-300MHz, which is much less than CPU and GPU. The FPGA's logic overhead for reconfigurability also reduces the overall system performance. Straight forward design on FPGA is hard to achieve high performance and high energy efficiency.
- Implementation of neural networks on FPGAs is much harder than that on CPUs or GPUs. Development framework like Caffe and Tensorflow for CPU and GPU is needed for FPGA.

Many researches on the above two problems have been carried out for energy efficient and flexible FPGA based neural network accelerator. In this paper, we summarize the techniques proposed in these work. Specifically, we will introduce the techiques from the following aspects:

- We investigate current techniques for high performance and energy efficient neural network accelerator designs. Techniques in both software level and hardware level are evaluated.
- We investigate state-of-the-art automatic design methods of FPGA based neural network accelerators.

The rest part of this paper is organized as follows:

2 PRELIMINARY ON NEURAL NETWORK

In this section, we introduce the basic operations included in neural network algorithms. Neural network is a bio-inspired model, which usually includes several layers. Each layer receives input from a set of neurons and output a set of neurons. The synapses connecting input and output neurons are modeled as parameters, which is referred to as weights in this paper. In the rest part of this section, we introduce different types of layers in neural network models.

2.1 Fully Connected Layer

Fully conencted (FC) layer implements a connection between every input neuron and output neuron with a weight. This type of layer is adopted in both CNN and RNN. The input and output neurons of an FC layer are two vectors $\mathbf{x}$ and $\mathbf{y}$. The weights of this layer can be modeled as a matrix W. A bias vector b is added to each of the output neuron. The computation of this layer is described as equation 1.

$$\mathbf{x} = W\mathbf{y} + \mathbf{b} \tag{1}$$

2.2 Convolution Layer

Convolution (CONV) layer is used for 2-d neuron process. This is commonly adopted in CNN for image process. The input and output neurons of this layer can be described as sets of 2-d feature maps, F_{in} and F_{out}. Each feature map is denoted as a channel. A CONV layer implements a 2-d convolution kernel K_{ij} for each input and output channel pair and a bias scalar b_i for each output channel. The computation of a CONV layer with M input channels and N output channels can be

described as equation 2.

$$F_{out}(j) = \sum_{i=0}^{M-1} \text{conv2d}(F_{in}(i), K_{ij}) + b_j,\ j = 0, 1, ..., N-1 \quad (2)$$

There are varieties of 2-d convolutions in CONV layer. Usually standard convolution with padding is used when the kernel size is 3×3. For larger kernels like 5×5 and 7×7, a stride larger than 1 is usually used to reduce the number of operation. Recent work is also using 1×1 convolution kernels [4, 5].

2.3 Non-linear Layer

Non-linear layer applies a non-linear function on each of the input neurons. Sigmoid function and tanh function are commonly adopted in early models are are still used in RNN for acoustic or speech recognition. Rectified linear unit (ReLU) [7] is the adopted in many state-of-the-art models. This function maintains the positive neurons and filters negative neurons as zero. Varieties of ReLU are also used, such as PReLU and Leaky ReLU [10].

2.4 Pooling Layer

Pooling layer is also used for 2-d neuron process like CONV layer. A pooling layer downsamples each of the input channel respectively, which helps reduce feature dimension. There are two kinds of down sampling method: average pooling and max pooling. Average pooling splits a feature map into small windows, i.e. 2×2 windows, and finds the average value of each window. Max pooling method finds the maximum value in each window. Common window size includes 2×2, stride=2 and 3×3, stride=2.

2.5 Element-wise Layer

Element-wize layer is usually used in RNN and is introduced in ResNet [4]. This layer receives two neuron vectors of the same size and applies element-wise operations on corresponding neurons of the two vectors. In ResNet, this layer is element-wise addition. For RNN, this layer can be element-wise subtraction or multiplication.

REFERENCES

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

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[3] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[5] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360* (2016).

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093* (2014).

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[9] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[10] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853* (2015).