A Survey on FPGA-based Deep Neural Network Accelerators

Mingyuan Li[1], Hengyi Li[2], and Lin Meng[2]

[1] School of computer science and information engineering, Hefei University of Technology, Xuancheng, Anhui China.
limingyuan0827@163.com
[2] Dept.of Electronic and Computer Engineering, Ritsumeikan University. Kusatsu, Shiga, Japan.
{gr0468kx@ed,menglin@fc} .ritsumei.ac.jp

Abstract

Currently, deep learning technologies have achieved great success in applying deep neural networks(DNNs) to multiple domains. However, their high functional intensity, whether computational or memory, has become a heavy burden in the utilization of deep learning, especially in constrained resource platforms. A potential solution is FPGA, which provides effective means for optimizing and accelerating DNNs. Therefore, an important of field of research has been the development of DNN applications with FPGA accelerators. In this paper, existing optimization techniques are evaluated to provide a comprehensive overview of FPGA-based DNN accelerators. The review herein addresses software- and hardware-level acceleration techniques (including, but not limited to, model compression, parameter quantization, and energy-efficiency in structural design).

1 Introduction

In recent years, deep neural networks(DNNs) have made substantial progress across a broad array of applications with excellent performance in all instances. Such applications include computer vision tasks, natural language processing, protection of cultural heritage [1, 2], and many others besides. As such, the flexibility of DNNs brings great convenience to modern standards of living. However, the demand for computation and memory in both quantity and complexity makes their deployment a heavy burden on resource constrained hardware platforms, such as robotics, mobile devices, etc. Nevertheless, training processes can be implemented on powerful devices such as GPUs, for which the inference process always works on these limited resource platforms. At the same time, research has revealed that there is massive redundancy in a given DNN's operations [3]. Therefore, research on the optimization and acceleration of DNNs has grown increasingly prominent.

One avenue of study is the field programmable gate array(FPGA) which provides an effective solution for DNN acceleration. FPGA has superior energy efficiency when compared to GPUs and CPUs, and although deeper networks have higher accuracy, they also greatly increase the number of parameters and the model size. The deeper network also brings an increased requirement for computing, bandwidth, and storage, meaning DNNs exert a heavy strain on resource constrained devices. Thanks to the rapid evolution of DNNs, possessing reprogrammable and reconfigurable hardware makes FPGA-based devices well suited to supporting them. FPGA's also feature high throughput, low power consumption, and high parallel workflow, which make the device's performance excellent for DNNs. In particular, the latest release of Intel FPGA-Agilex possesses improved chip layout, optimizing the architecture's design and algorithm which results in considerable enhanced flexibility and stability. Based on

this, FPGA can accelerate DNNs with a high level of efficiency on edge devices, and research has proposed multiple techniques for further acceleration in performance.

In this paper, we conduct a survey on the FPGA-based optimization for DNNs. For their implementation, optimization techniques as detailed within can be divided into two categories: software level, on algorithms; and hardware level, based on the FPGA itself. Subsequent to this review will be the introduction of optimizing methods on the software level, the expansion of acceleration techniques based on FPGA architecture, and a summary conclusion.

2 DNNs optimization on software level

Multiple techniques on software level to improve DNNs with high efficiency have been proposed. This section gives an overview of software level optimizations on DNNs.

2.1 Pruning and quantifying

Pruning and quantification are effective ways to compress neural n etworks. Network pruning is to remove redundant connections and ensure the effectiveness of neural network connections to improve efficiency. Data quantization is to quantify the DNN model parameters by replacing the float-point r epresentation w ith t he fi xed-point re presentation or re ducing th e nu mber of bits used for the representation. And with experimental verification, t he q uantified da ta has little impact on the accuracy of the models. At the same time, for the reason that FPGA is not suitable for floating-point o peration a nd t o o ptimize t he p arameters o f D NNs, data quantification is also essential for D NN m odels t o b e d eployed o n FPGA.

The Binarization network is a very effective solution. Many scientists have engaged in research. Rastegari M et al. [4] proposed two kinds of binarization networks to reduce the model storage space through the binarization operation of weight. The former is called Binary-Weight-Networks, which is to approximate its property weight with binary values, and the convolution operation is estimated only by addition and subtraction. The latter is called XNOR-Networks, the weights and the inputs of the convolutional layers as well as fully connected layers are all approximated by binarization, and the convolution can be estimated by XNOR and bit counting operations. In Binary-Weight Networks, the filters are approximate with binary values resulting in 32 memory saving. And XNOR-Networks makes convolution 58× faster and 32× memory savings.

Currently, 32-bit floating p oint d ata d oes n ot p erform p erfectly o n D NNS F PGA accelerators, so most of the advanced accelerators replace 32-bit floating p oint d ata w ith lower fixed-point r epresentations. I n [5] P odili e t a l. p roposed t o r eplace t he 3 2-bits floating-point data with 32-bits fixed-point d ata. A ccording t o [6], Q iu e t a l. p roposed t o u se 1 6-bits fixed-point data to replace the 32-bits floating-point d ata. A nd i n [7] G uo e t a l. p roposed that data quantization strategy helps reduce the bit-width down to 8-bit with negligible accuracy loss. These improvements on floating p oint b its g reatly i mprove t he e fficiency o f computation without decreasing accuracy.

By removing the insignificant c hannels o f t he n etwork a nd q uantizing t he w eights a nd bias expressed by floating p oint numbers (high precision) t o b e low precision integers, t he size of the models and computation demand can be greatly reduced, which are effective means for DNN model compression.

2.2 Knowledge of distillation

Knowledge distillation is to transfer dark knowledge-what deep learning methods actually learn, from complex (teacher) model to simple (student) model by minimizing a loss function. Generally speaking, the teacher model has strong ability and performance, while the student model is more compact. Through knowledge distillation, it is hoped that the student model can approach or surpass the teacher model as much as possible, so as to obtain similar prediction accuracy with less complexity.

Hiton [8] adopted the strategy of feature matching within the Softmax layer for processing. Its essence was to use Softmax output as supervision. But in order to make the score vector softer, distillation temperature T is added to the Softmax layer to improve the performance of distillation. The model trains the teacher at $T = 1$, uses the output probability of the teacher softmax as the soft label at high temperature to fuse with the hard label to supervise the student, and weighs the loss of the two. With 61 specialist models, there is a 4.4 percent relative improvement in test accuracy overall.

Zagoruyko [9] thought that direct transferring of feature map as knowledge from teacher to student is too rigid and the effect is poor. He hoped that the student could pay attention to the areas that the teacher takes care. Therefore, he took the absolute values of feature planes of different channels in the feature map for power operation and then added them together, narrowing the distance between teacher and student's attention map. It will have better effects than direct transfer feature map.

Yang [10] thought that hard labels would lead to overfitting of the model, but soft labels would contribute to the generalization ability of the model. In this regard, he proposed a method that did not calculate the additional loss of all classes, but selected several classes with the highest confidence score. During the training of teacher in the experiment, a constraint was added to teacher's loss for selection. And during the training of students, the teacher's soft labels obtained previously are combined with the hard labels. Experiments show that this method improves the classification efficiency of data sets by 3 to 8 percent.

2.3 low-rank matrix factorization

While DNNs have achieved tremendous successes for many tasks, the training process of these networks is time and resource expensive. One of the major reasons is that DNNs are trained in a large number of parameters. Meanwhile, Low-rank factorization is a very effective method to reduce the number of parameters.

Sainath et al. [11] proposed a low-rank matrix factorization of the final weight layer, and applied this low-rank technique to DNNs for both acoustic modeling and language modeling. This method reduced the number of parameters of the network by 30-50 percent.

For some simple DNN models, a few low-rank approximation and clustering schemes for the convolutional kernels were proposed in [12]. They exploited the redundancy present within the convolutional filters to derive approximations that significantly reduce the required computation, and their method achieved 2× speedup for a single convolutional layer with 1 percent drop in classification accuracy.

The work in [13] proposed using different tensor decomposition schemes. This is achieved by exploiting cross-channel or filter redundancy to construct a low rank basis of filters that are rank-1 in the spatial domain. Reporting a 4.5× speedup with 1 percent drop in accuracy in text recognition.

2.4 Filter dimension reduction

Most of the advanced DNN models, such as Googlenet, and ResNet [14], use a large convolutional filter size in the first convolution layer, thus giving the DNN model a larger acceptance area for better performance. However, larger filter sizes tend to be computationally expensive.

Karpathy [15] proposed that 7×7 filter can be replaced by 3×3 stacked filters. In this way the network has smaller costs, and requires only about 50 percent the MACC operations required by a 7×7 filter.

Gschwend [16], another researcher working on this study replaced the 7×7 filter with a 3×3 filter, and proved that the accuracy decreased by less than 1 percent after the replacement. It proved that the use of a smaller filter can be applied without compromising the accuracy.

3 Hardware-level acceleration on FPGA

There are multiple hardware platforms for DNNs, such as GPU, CPU, ASI, FPGA, etc. It is hard to say which works best for all deep learning applications. FPGA just offers some distinct advantages for DNNs. In this section, FPGA-based accelerations of DNNs are introduced in this section.

3.1 Acceleration based on sparsity

The high percentage of sparsity causes a serious problem of computation resource underutilization in sparse CNN accelerators, especially for the irregularity of sparsity. However, FPGA provides an effective solution on hardware level.

Yijin Guan et al. [17]. proposed an accelerator named Crane. In the accelerator, DMA can only obtain non-zero activation data and weights, and store them on the chip for convolution processing. The output RAM stores all generated results and transmits them to the output unit for convolution post-processing, including activation functions, pooling, and encoding. Experimental results show that Crane improves performance by 27 - 88 percent and reduces energy consumption by 16 - 48 percent, respectively, compared to the counterparts.

Zhang et al. [18]. proposed a software-based coarse-grained pruning technique to significantly reduce the irregularities of sparse synapses. He combines coarse-grained pruning techniques with local quantization techniques, which can significantly reduce the index size and improve the network compression ratio. They further designed a hardware accelerator, Cambricon-X, to efficiently address the remaining sparse synapses and neuronal irregularities. Experimental results over a number of representative sparse networks show that the accelerator achieves, on average, 7.23× speedup and 6.43× energy saving against the state-of-the-art NN accelerator.

Zhou et al. [19], proposed an accelerator featuring processing elements (PE) -based architecture consisting of multiple PEs. Indexing modules efficiently select and transmit the desired neurons to the connected PE, reducing bandwidth requirements, while each PE stores irregular and compressed synapses in an asynchronous manner for local computation. Compared with a state-of-the-art sparse neural network accelerator, the accelerator is 1.71× and 1.37× better in terms of performance and energy efficiency, respectively.

3.2 Structure specialized for DNNs

With the feature of hardware reprogrammable and reconfigurable, DNNs can be accelerated on FPGA by elaborately designing the implementation method.

Sina Ghaffari et al. [20] designed two kinds of specialized hardware architectures for DNNs. The first architecture is suitable for the small DNNs of applications. The researchers designed specific hardware for each individual layer. The second architecture has one hardware designed for each layer that is used several times as we need different layers. There is a control loop deciding when to use each hardware. With this technique, the network can have as many layers as needed with the same resources. This architecture is extensive and can be easily used for large networks.

Although PE parallel computation improves the computation speed, there may be time delay inside PEs while FPGA is carrying out convolution calculations. Eyi Wang et al. [21] realizes the pipeline optimization of convolution operations by adding registers between two data processing nodes. During the process of data flow, each register stores the calculated data of the node in each clock cycle and will cache the data in the clock cycle to the next calculated node.

3.3 Resources utilization

One of the key issues for FPGA-based DNN accelerators is that the computational throughput might not match well for the memory bandwidth provided by the FPGA platform. And many methods have failed to achieve optimal performance without making full use of memory bandwidth and logical resources. As a result, making full use of FPGA resources has been a very important research direction.

In [22], Zhang proposed an analytical design scheme using the roofline model. For the solution of a CNN design, the research quantitatively analyzes its computing throughput and required memory bandwidth using various optimization techniques, such as loop tiling and transformation. Then, with the help of roofline model, we can identify the solution with best performance and lowest FPGA resource requirement.

In [5], Huimin proposed an end-to-end FPGA based CNN accelerator with all the layers mapped on one chip, so that different layers can work concurrently in a pipelined structure to increase the throughput. And a methodology which can find the optimized parallelism strategy for each layer is proposed to achieve high throughput and high resource utilization.

3.4 Data flow optimization

For DNNs accelerator based on FPGA without fully studying the convolution loop optimization before the hardware design phase, the resulting accelerator can hardly exploit the data reuse and manage data movement efficiently. Therefore, the optimization of data flow is very important in our opinion.

Yufei Ma et al. [23] put forward through quantitative analysis and optimization method based on many design variables to optimize the convolution cycle, through the search design variable configuration, they put forward CNN hardware accelerators clear data flow to minimize memory access and data movement. At the same time, data flow is also used to maximize resource utilization in order to obtain high performance.

In [24], Ding proposed an FPGA-based depthwise separable CNN accelerator with all the layers working concurrently in a pipelined fashion to improve the system throughput and performance. To implement the accelerator, The paper presented a custom computing engine architecture to handle the dataflow between adjacent layers by using double-buffering-based memory channels. This method achieved up to 17.6× speed up and 29.4× low power than CPU and GPU implementations respectively.

3.5 DNNs implementation on FPGA

The complexity and development overhead of the HDL(Hardware Description Language) make it difficult to implement the algorithms on FPGA-based platforms efficiently, especially for DNNs. There have been multiple tools to bridge the gap between DNNs and FPGA, which liberates researchers to concentrate on the study of DNN algorithms. For example:

Vitis AI, the AI development environment of Xilinx for AI inference on Xilinx hardware platforms, supports mainstream frameworks such as Caffe, PyTorch, TensorFlow, and latest models capable of diverse deep learning tasks. The Xilinx also provides the Vitis HLS tool to synthesize a C or C++ function into RTL code for acceleration in programmable logic device.

TF2FPGA [25], a framework that extends the well known TensorFlow system with automic FPGA acceleration capabilities, enables automatic and transparent generation of high throughput DNN accelerators implemented on FPGA.

4 Conclusion

As presented here, a survey on the DNN acceleration technologies provides an illustration of an ideal FPGA accelerator as the embodiment of a high level of hardware and software cooperation. At the software level, this review summarizes the existing techniques for DNN acceleration, which are prerequisites for them to be applied on FPGAs. And concerning hardware, the featured approaches further optimize acceleration while focusing on different aspects. Due to FPGA-based DNN accelerators being the vital feature for embedded application implementation, this study forms a comprehensive reference for future research.

References

[1] Lyu Bing, Hiroyuki Tomiyama, and Lin Meng. Frame detection and text line segmentation for early japanese books understanding. In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods - ICPRAM,*, pages 600–606. INSTICC, SciTePress, 2020.

[2] Lin Meng, Bing Lyu, Zhiyu Zhang, C.V.Aravinda, Naoto Kamitoku, and Katsuhiro Yamazaki. Oracle bone inscription detector based on ssd. *ICIAP2019*, pages 126–136, 2019.

[3] Hengyi Li, Zhichen Wang, Xuebin Yue, Wenwen Wang, Tomiyama Hiroyuki, and Lin Meng. A comprehensive analysis of low-impact computations in deep learning workloads. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, GLSVLSI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-net: ImageNet classification using binary convolutional neural networks.

[5] Huimin Li, Xitian Fan, Li Jiao, Wei Cao, Xuegong Zhou, and Lingli Wang. A high performance FPGA-based accelerator for large-scale convolutional neural networks. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–9. IEEE.

[6] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. Going deeper with embedded FPGA platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 26–35. ACM.

[7] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. Angel-eye: A complete design flow for mapping CNN onto embedded FPGA. 37(1):35–47.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network.

[9] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer.

[10] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan Yuille. Knowledge distillation in generations: More tolerant teachers educate better students.

[11] Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659. IEEE.

[12] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation.

[13] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE.

[15] A Karpathy. Cs231n neural networks part 3: learning and evaluation, 2016.

[16] Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, Mohammad Sabokrou, and Ehsan Adeli. Towards principled design of deep convolutional networks: Introducing SimpNet.

[17] Yijin Guan, Guangyu Sun, Zhihang Yuan, Xingchen Li, Ningyi Xu, Shu Chen, Jason Cong, and Yuan Xie. Crane: Mitigating accelerator under-utilization caused by sparsity irregularities in CNNs. 69(7):931–943.

[18] Shijin Zhang, Zidong Du, Lei Zhang, Huiying Lan, Shaoli Liu, Ling Li, Qi Guo, Tianshi Chen, and Yunji Chen. Cambricon-x: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE.

[19] Xuda Zhou, Zidong Du, Qi Guo, Shaoli Liu, Chengsi Liu, Chao Wang, Xuehai Zhou, Ling Li, Tianshi Chen, and Yunji Chen. Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 15–28. IEEE.

[20] Sina Ghaffari and Saeed Sharifian. FPGA-based convolutional neural network accelerator design using high level synthesize. In *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, pages 1–6. IEEE.

[21] Enyi Wang and Dehui Qiu. Acceleration and implementation of convolutional neural network based on FPGA. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 321–325. IEEE.

[22] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing FPGA-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 161–170. ACM.

[23] Yufei Ma, Yu Cao, Sarma Vrudhula, and Jae-sun Seo. Optimizing loop operation and dataflow in FPGA acceleration of deep convolutional neural networks. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 45–54. ACM.

[24] Wei Ding, Zeyu Huang, Zunkai Huang, Li Tian, Hui Wang, and Songlin Feng. Designing efficient accelerator of depthwise separable convolutional neural network on FPGA. 97:278–286.

[25] Spyridon Mouselinos, Vasileios Leon, Sotirios Xydis, Dimitrios Soudris, and Kiamal Pekmestzi. Tf2fpga: A framework for projecting and accelerating tensorflow cnns on fpga platforms. In *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4. IEEE, 2019.