Journal of Systems Architecture 119 (2021) 102257

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Accelerating DNNs from local to virtualized FPGA in the Cloud: A survey of trends

Chen Wu [a,*], Virginie Fresse [a], Benoit Suffran [b], Hubert Konik [a]

[a] Hubert Curien Laboratory, University of Saint-Etienne, University of Lyon, France
[b] ST Microelectronics, Grenoble, 38000, France

ARTICLE INFO

Keywords:
FPGA virtualization
Cloud computing
Deep neural network
Accelerator
Trends

ABSTRACT

Field-programmable gate arrays (FPGAs) are widely used locally to speed up deep neural network (DNN) algorithms with high computational throughput and energy efficiency. Virtualizing FPGA and deploying FPGAs in the cloud are becoming increasingly attractive methods for DNN acceleration because they can enhance the computing ability to achieve on-demand acceleration across multiple users. In the past five years, researchers have extensively investigated various directions of FPGA-based DNN accelerators, such as algorithm optimization, architecture exploration, capacity improvement, resource sharing, and cloud construction. However, previous DNN accelerator surveys mainly focused on optimizing the DNN performance on a local FPGA, ignoring the trend of placing DNN accelerators in the cloud's FPGA.

In this study, we conducted an in-depth investigation of the technologies used in FPGA-based DNN accelerators, including but not limited to architectural design, optimization strategies, virtualization technologies, and cloud services. Additionally, we studied the evolution of DNN accelerators, e.g., from a single DNN to framework-generated DNNs, from physical to virtualized FPGAs, from local to the cloud, and from single-user to multi-tenant. We also identified significant obstacles for DNN acceleration in the cloud. This article enhances the current understanding of the evolution of FPGA-based DNN accelerators.

1. Introduction

Deep neural networks (DNNs) have become a cutting-edge research topic owing to their excellent performance in image classification, detection, segmentation, and data prediction. Owing to the remarkable prediction capacity of datasets in a wide range of complex applications, researchers have proposed myriad networks, such as AlexNet [1], VGG16 [2], ResNet152 [3], Transformers [4], General Adversarial Networks (GANs) [5], and Variational Autoencoder (VAE) [6]. The success of DNNs has also attracted attention in the development of industrial platforms, such as Google Deepmind [7], Facebook AI [8], Amazon Alexa [9].

Traditionally, in academia and industry, graphics processing units (GPUs) are used to train DNNs, as they provide a high degree of parallelism to process these algorithms [10,11]. However, the execution of DNNs on GPU-based platforms encounters energy/power and throughput bottlenecks. In 2016, a tensor processing unit (TPU) was announced by Google [12], which runs DNNs 15 to 30 times faster than contemporary GPUs using similar technologies [13], and the energy efficiency is increased by a factor of 30–80. Despite its speedup and energy efficiency, the TPU has a high production cost, lacks reconfigurability, and cannot be adapted to the emergence of new network models with complex structures.

Field-programmable gate arrays (FPGAs) can achieve energy efficiency and high performance in the face of rapidly innovating DNN models and computational characteristics, as reported by Venieris et al. [14]. FPGAs can achieve up to 20 tera multiply accumulates per second (TMACs), and the power consumption does not exceed 25 W, incurring a less than 10% overhead in the overall power consumption [15]. Moreover, FPGAs can provide a flexible hardware architecture with a fine granularity and massive pipeline level. Therefore, FPGAs have become an alternative method for accelerating DNNs.

Early DNN accelerators (e.g., [16–19]) are typically implemented on a single local FPGA fabric. As the number of learnable parameters and operations in DNNs increases, the resources of a single FPGA may be insufficient for the entire DNN deployment. The challenges of designing DNNs on a single local FPGA are described below.

- Productivity: Owing to the complexity of DNN design, mapping a DNN onto an FPGA requires specific hardware expertise in

* Corresponding author.
E-mail address: chen.wu@univ-st-etienne.fr (C. Wu).

https://doi.org/10.1016/j.sysarc.2021.102257
Received 29 April 2021; Received in revised form 29 July 2021; Accepted 2 August 2021
Available online 16 August 2021

hardware description language programming and performance optimization, which have long learning curves. According to the complexity of the DNN algorithm, deploying the DNN on the FPGA may be time-consuming and may increase the programming burden of designers. In recent years, productivity has improved owing to the emergence of compilation frameworks that automatically map DNNs onto the FPGA.

- Scalability: DNNs are computation- and data-intensive applications that require enormous computational resources. For example, VGG-16 has up to 39 billion operations and more than 500 million parameters for 224 × 224 image classification [20]. In deeper DNNs, the resource requirements may exceed the available resources in a single FPGA, limiting the scalability of the DNN architecture. Even if technologies and strategies are adopted to optimize the DNN architecture, when a large-scale DNN is deployed in a single local FPGA, the resource bottleneck can easily be reached.
- Elasticity: The solution of deploying DNN accelerators on local FPGAs lacks resource elasticity because it assumes that DNN resource allocation must be fixed throughout the deployment lifecycle. Because different DNN algorithms require different computing resources, memory bandwidths, and storage resources [21], these solutions cannot flexibly provide and deprovision resources at runtime and hence fail to match different workloads of the DNN.
- Portability: The deployment of most DNN accelerators directly depends on the characteristics of the FPGA platform and is therefore restricted to a specific FPGA vendor. Owing to the lack of an abstraction layer that isolates DNN accelerators from specific FPGA platforms, these accelerators may face portability issues of DNN structures. They cannot adapt quickly to the current changing DNN algorithms.
- Multi-tasks: Generally, the execution mode of a DNN on a local FPGA is limited to a single user executing a single DNN within a given time. It remains difficult for a single local FPGA to support multiple users by executing execute multiple DNNs in parallel and satisfy each user's time, cost, and quality of service (QoS) requirements. Some frameworks (for example, [22]) successfully solve the problem of multiple DNN scheduling but can only execute DNNs sequentially in the form of time slices in a single-task environment.

Deploying FPGAs in the cloud and/or virtualizing FPGAs can resolve the aforementioned challenges, as shown in Fig. 1. The cloud paradigm enhances the computing capability of FPGAs with high throughput and low latency. It enables the sharing of single or multiple FPGA resources across multiple users, thereby efficiently scaling and accelerating on-demand DNNs. Virtualizing FPGAs abstracts low-level physical resources and hides hardware design and compilation flow from the software designers' view, providing a high-level application-dependent architecture. Owing to FPGA virtualization, software designers can deploy DNN accelerators according to different requirements (e.g., throughput, execution time, and accuracy) without relying on a specific FPGA platform. The collaboration between FPGA virtualization and the FPGA cloud can satisfy the resource requirement, thereby taking advantage of the "unlimited" cloud capability to flexibly scale DNN accelerators.

While exploring techniques to accelerate DNNs on local physical FPGAs, researchers have also attempted to adopt FPGA virtualization and the FPGA cloud to facilitate the implementation of multiple DNNs at a large scale and achieve flexible deployment in a multi-user environment. Although the FPGA cloud and virtualization have brought breakthroughs to the deployment of DNNs, previous surveys (e.g., [23–29]) have mainly focused on DNN optimization and the design of local FPGAs (e.g., architecture design, simplification, optimization strategies). These surveys ignore the trend of DNN implementation on the timeline, that is, from local to virtual FPGA in the cloud. Moreover, no in-depth analysis or comparison of the challenges faced by the DNN accelerators at different stages of deployment was conducted. Relying on previous surveys, we aim to

- Provide an overview of the main techniques of FPGA-based DNN accelerators. These techniques were initially proposed to optimize the performance of DNN accelerators in a local FPGA, but they can also be applied to the FPGA cloud environment.
- Present the evolution of DNN accelerator deployment from local to virtualized FPGAs through an in-depth introduction of virtualization techniques and the FPGA cloud.
- Perform an in-depth analysis and comparison of the challenges faced by the DNN accelerators at each stage.

The article is organized as follows: Section 2 provides an FPGA cloud definition and a general overview of FPGA virtualization. Section 3 discusses the crucial approaches for accelerating DNNs on the FPGA, which is also applicable to the FPGA cloud. Section 4 describes the use of virtualization technology in local DNNs and cloud-based DNNs. Section 5 highlights the trends and evolution of the FPGA-based DNN and compares the characteristics of these accelerators. Section 6 discusses the unresolved challenges of accelerating DNNs in the FPGA cloud and presents other directions for DNN acceleration. Section 7 gives the conclusion of the survey.

2. Background

This section presents an overview of the FPGA cloud and the available services in the cloud and introduces the FPGA virtualization technology from the viewpoints of the abstraction level and system architecture.

2.1. FPGA cloud

Deploying FPGAs in the cloud involves leasing a bundle of specific software tools, platforms, or FPGA resources remotely in a cost-effective manner. Such an FPGA-enabled cloud maintains the advantages of FPGAs (e.g., low power consumption and programmability) and establishes scalability, elasticity, and multi-tenancy.

Provisioning FPGA resources is similar to provisioning traditional central processing unit (CPU)- and GPU-based clouds. Regarding the service categories in traditional cloud computing, FPGA cloud providers offer FPGAs as infrastructure as a service (IaaS) or software as a service (SaaS) [30]. Fig. 2 presents an example of hierarchical mapping in the FPGA cloud. There is no standard definition or classification for FPGA clouds, and the hierarchical mapping may change over time.

2.1.1. FPGA in IaaS

The FPGA in IaaS provides access to the FPGA computing resource pool and memory storage in the cloud. This paradigm divides the FPGA into multiple independent virtual instances and supports high-bandwidth communication to collaborate between each resource instance. Per-FPGA or multiple-FPGA granularity can be supported in the IaaS for application deployment. Cloud users must manually map their applications to resources if their applications are deployed across multiple FPGAs.

As a commercial example, the Amazon F1 instance offers a collection of eight FPGA devices with a high bandwidth. Enabling FPGA in IaaS has also attracted attention in the academic field. Byma et al. [31] abstracted FPGAs into virtual regions and managed resources across multiple FPGAs through OpenStack. Asiatici et al. [32] provided a runtime management framework to map FPGA resources for different applications with limited overhead.

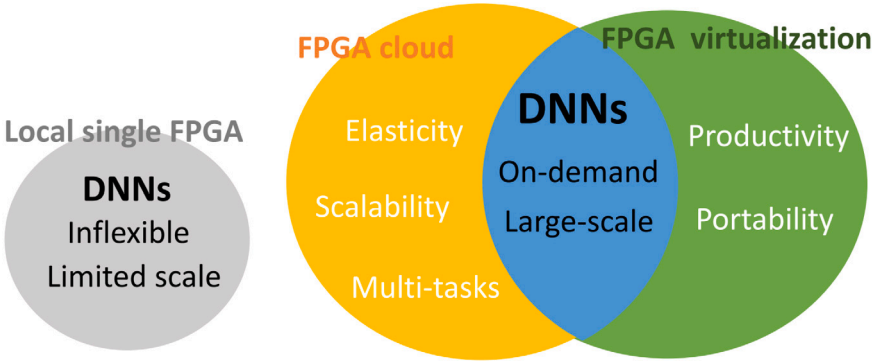

Fig. 1. Characteristics of deploying FPGAs in the cloud and FPGA virtualization for DNN deployment.

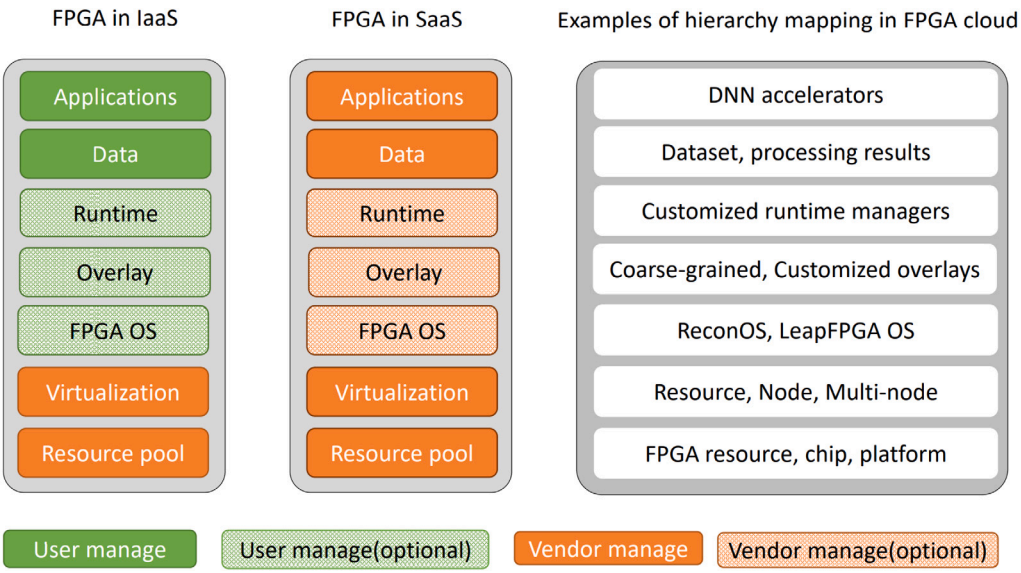

Fig. 2. IaaS and SaaS FPGA cloud. "Vendor manage (optional)" and "User manage (optional)" indicate that this hierarchy does not always exist in the FPGA cloud, and it is customized by each FPGA cloud vendor or user.

2.1.2. FPGA in SaaS

The FPGA in SaaS offers acceleration services for cloud users to execute applications and process data. Technical processes have been hidden in the cloud background, and cloud users do not need to be responsible for the hardware design flow and FPGA resource management. For example, Microsoft released the Catapult project [33], which puts Altera Stratix vFPGA per CPU in the cloud to accelerate the Bing web search engine, with a 95% improvement throughout. Moreover, Microsoft released the BrainWare project, where FPGAs are used to accelerate state-of-the-art DNNs in major services such as Bing and Azure [34].

2.2. FPGA virtualization

The objectives of FPGA virtualization are to (1) provide a virtual abstraction of resources and underly the low-level hardware design from users; (2) support FPGA sharing in the time and space domains to serve multiple tasks; and (3) facilitate the hardware design process and accelerate the program compilation [35–38]. We review FPGA virtualization according to the abstraction level [39] and system architecture [40]. The definition of FPGA virtualization has changed over time in different scenarios.

2.2.1. Abstraction level

According to the scale of resource computing, FPGA virtualization can be divided into three abstraction levels: resource, node, and multi-node levels.

- Resource level: The resource level contains reconfigurable resources (e.g., logic) and non-reconfigurable resources (e.g., Input/Output blocks). Several uniform architectures, such as coarse-grained overlays, have been proposed to support the portability of this level between different types of FPGAs [41,42].
- Node level: The node level considers a single FPGA as a node. Resource allocation and scheduling are concerned with a single FPGA at this level. Currently, time-division multiplexing (TDM) and space-division multiplexing (SDM) are the two principal methods for sharing a single FPGA resource [43,44].
- Multi-node level: The multi-node level is designed to assign resources in multiple FPGAs to multiple applications or multiple users. However, mainstream compilation tools only support application deployments on a single FPGA [45]. Therefore, application mapping across FPGAs requires specific frameworks to solve hardware problems, such as intercommunication, resource partitioning, and traversing the physical boundary.

2.2.2. System architecture

The system architecture refers to a structural view at the abstraction level. It usually covers the hardware, software stack, and overlay [46] but may be different at each level of abstraction. Here, we introduce the system architecture in a node-level abstract form, as shown in Fig. 3, which can also be applied to other levels of abstraction.

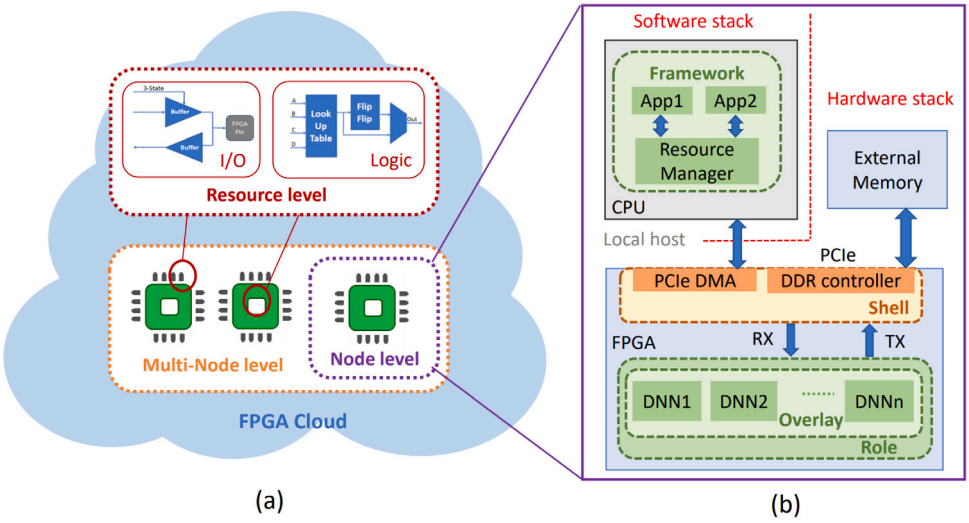

Fig. 3. Overall architecture of the FPGA-based DNN accelerators in the IaaS cloud. (a) Different levels of abstraction in the FPGA virtualization technique. (b) Example of the system architecture in node-level virtualization.

- Hardware stack: The hardware stack can vary in the host interface, shell, and role.
  - Host interfaces: (1) on-chip host inside the FPGA, which can be a soft core formed by programmable logic (PL) or a hard core in the processing system (PS) of a system-on-a-chip (SoC) FPGA; (2) local host, local CPU host, connected via high-bandwidth links (e.g., PCIe); (3) remote host placed remotely via the network.
  - Shell: The shell is a static region, usually comprising a system memory controller (e.g., DRAM adapter), interface controller (e.g., DMA controller), and network interface controller (e.g., Ethernet core). For instance, the shell in [47] includes the user PCIe, management PCIe, card management system, and DDR access channel.
  - Role: The role is a dynamic region in the FPGA, which can be regarded as a reserved region for deploying DNNs in our context. It runs independently of the shell and can be reconfigured every time for each application to satisfy user requirements.
- Software stack: The software stack runs on a host, provides users with an application programming interface, and enables the communication between the host and the FPGA. [40] introduces three types of software stacks: (1) Operating systems (e.g., LeapFPGA OS [48], Recon OS [49]), which are conceived to support multiple threads for runtime resource management. (2) The host application, which is written in OpenCL and C++, provides simultaneous access to a shared FPGA for multiple users. (3) Software frameworks (e.g., OpenStack), which can be used to share resources across multiple users and distribute several partial reconfigurations to one FPGA.
- Overlay: The overlay provides an intermediate layer between the hardware stack and the software stack to achieve program portability. It is considered a virtual reconfigurable architecture on top of a physical FPGA. Fine-grained granularity and coarse-grained granularity in overlays are used in various applications [50,51].

3. DNN implementation techniques

To enhance the performance of DNNs on the FPGA locally and in the cloud, several techniques have been extensively studied. This section presents implementation techniques that have been recently investigated.

3.1. Hardware architecture design

The widely used hardware architecture are streaming and single computation engine architectures.

- Streaming architecture: The streaming architecture (Fig. 4) implements an entire DNN on the side of the Programmable logic (PL) of the FPGA from the first convolutional layer to the final fully connected layer. On the PL side, it deploys a chain of sequential DNN intellectual property (IP) to process the dataset in the pipeline mode. The intermediate results (DNN feature maps) are stored on the chip on the PL side. This architecture enables an efficient data stream without frequent data exchange with external memory, significantly reducing the latency and obtaining throughput at a high frequency. A specific DNN model using a streaming architecture must be defined before generating the bitstream. Whenever the DNN model changes, architecture re-compilation and bitstream regeneration are inevitable. According to the selected DNN algorithm, the re-compilation of this architecture may be time- and resource-consuming.
- Single Computation Engine: Single computation engine (Fig. 5) implements a part or a layer of the DNN on the PL side. It is a universal fixed template, usually in the representation of a systolic array or multiple processing elements that can be configured as DNN layers of different scales [52,53]. The execution of the entire DNN is achieved sequentially by configuring this template in the program on the PS side. The intermediate results (DNN feature maps) are stored off the chip. The architecture significantly reduces resource usage and introduces possibilities to scale accelerators. It has been widely used in accelerators to enrich DNN diversity. However, because the DNN blocks are executed sequentially on the FPGA, the execution time is extended significantly. Each time the DNN architectures changes, it is necessary to reload a complete bitstream to realize the novel DNN deployments on the FPGAs.

Table 1 presents the major features of streaming and single computing engine architectures according to their performance (e.g., flexibility, reconfiguration, resource consumption).

3.2. Network compression

The increasing amounts of learnable parameters and arithmetic operations of DNNs lead to a computational burden and additional resource consumption of hardware devices. Network compression makes

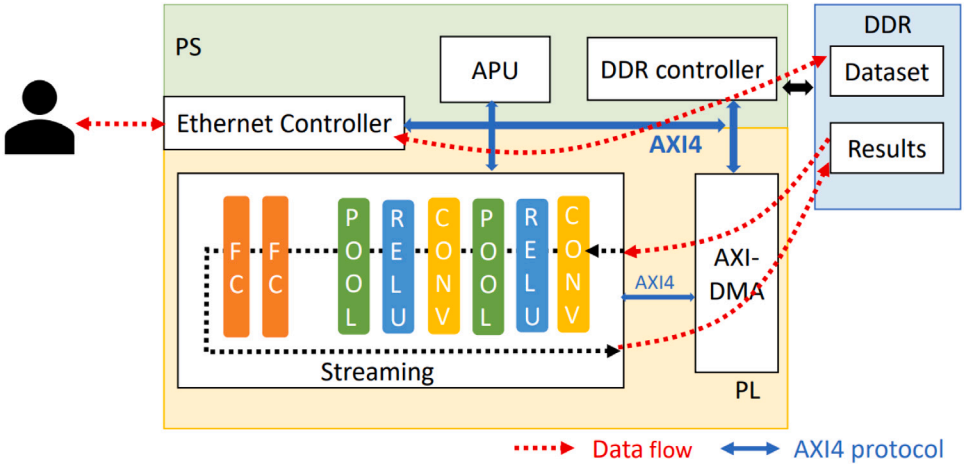

Fig. 4. Example of accelerating DNNs using the streaming architecture.

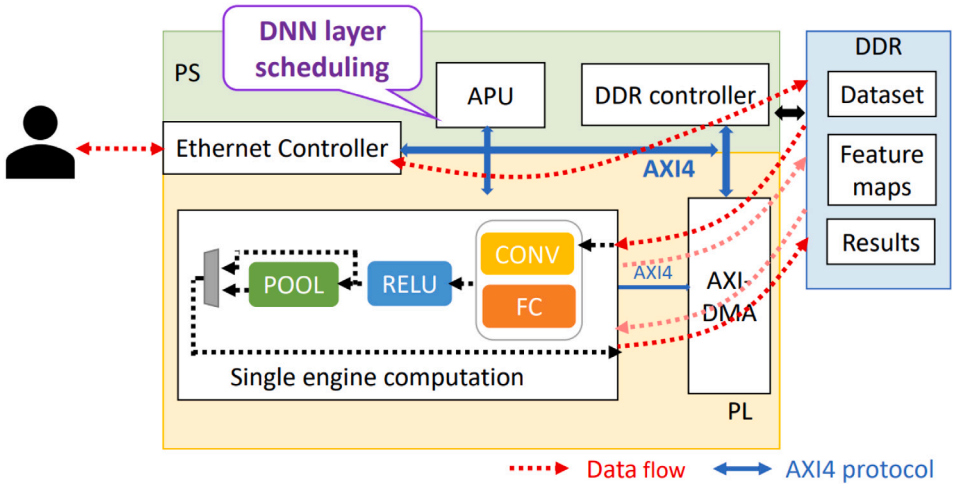

Fig. 5. Example of accelerating DNNs using the single computation engine accelerator architecture.

Table 1
Comparison of streaming and single computation architectures for DNN acceleration.

	Streaming	Single computation
Network implementation	Entire network	Function unit
Structure	Pipeline	Recurrent
Optimization mode	Layer-independent	One-optimization-fit-all
Recompilation time	Long	Short
Reconfiguration	Bitstream-level reconfiguration	Processor control configuration
Flexibility	Low	High
Resource usage	High	Low
Speedup	Fast	Low

DNNs more compact when the data width is limited, assisting in striking a balance between resource usage and accuracy. Thus far, quantization, pruning, and in-parallel pruning quantization have been successfully employed for network compression.

- Quantization: Network quantization converts floating-point data to fixed-point data with a selectable data width. Quantization includes uniform quantization with the same width for all network layers or dynamic quantization of each layer based on the layer characteristics. Researchers have widely adopted 16-bit fixed-point quantization (for example, [54,55]), and 4- and 8-bit uniform quantization [56,57] have already achieved good accuracy. Therefore, uniform quantization of a small width is promising owing to its ease of implementation on FPGA while maintaining accuracy.
- Pruning: Network pruning removes nonsignificant neurons to avoid overfitting. This is an efficient method, particularly in embedded systems, for reducing the network size and saving computing resources to fit the network to the memory size [58]. In [59], the authors compressed a trained DNN model and performed reverse pruning and peak pruning with fewer weights. Compared with the GPU, the compressed AlexNet on FPGA achieved 182.3× and 1.1× improvements in latency and throughput, respectively.

3.3. Optimization strategy

The scale of complex DNN structures introduces resource challenges. Moreover, the data (e.g., weights) stored in the external memory require enormous energy and latency. Because DNNs are composed

Table 2
Several examples of manual mapping and frameworks on the local FPGA.

	Works	Year	DNN models	Device	Data format	Architecture	Strategy	Perform. (GOPs)
Manual	[60]	2015	Costum CNN	Xilinx Virtex vc707	32-bit FP[a]	Single engine	Unrolling	61.62
	[61]	2015	CIFAR10	Xilinx Kintex 325 T	16-bit	Single engine	Quantization	260
	[53]	2016	VGG-16	Xilinx Zynq XC7Z045	16-bit	Single engine	Unrolling,tilling	137.3
	[62]	2016	AlexNet	Xilinx vc707	32-bit FP[a]	Single engine	Unrolling,tilling	75.16
	[63]	2016	AlexNet	xilinx vc709	16-bit	Streaming	Quantization, ping-pong buffer,batching	565.94
	[64]	2017	LRCN	Xilinx vc710	16-bit	Single engine	Pruning, quantization, unrolling, tilling	75.5
Framework	[65]	2017	VGG16 YOLO	Xilinx Zynq xc7z020	8-bit	Single engine	Per-layer quantization	84.3 62.9
	[66]	2016	LeNet-5 MPCNN CNP CFF	Xilinx Zynq xc7z02	32-bit FP[a]	Streaming	Pipeline	185.81 100.23 150.91 159.22
	[67]	2017	BNN-SFC BNN-LFC BNN-CNV	Xilinx Zynq ZC706	1-bit	Streaming	Pipeline, binarized network	8265 908 246
	[54]	2017	VGG-19 LSTM-LM REsNet-152	Altera StratixV SGSMD5	16-bit	Single engine	Tilling, batching	364.36 315.85 226.47
	[68]	2017	AlexNet GoogLeNet ResNet-50	Xilinx Zynq XC7Z045	16-bit	Single engine	Quantization, loop removal, rearrangement	120.3 116 122.3
	[69]	2018	AlexNet VGG16	Xilinx UltraScale KU060 Xilinx Virtex vc709 Xilinx UltraScale KU060	16-bit	Single engine	Unrolling, pipeline	163 354 266
	[70]	2018	AlexNet NiN	Altera Stratix V GXA7	8-bit	Single engine	Unrolling	114.5 117.3
	[71]	2019	VGG ResNet50 GoogLeNet	Xilinx ZU2	8-bit	Streaming	Quantization, tilling	334 228.7 231.5

[a]FP = Floating point format.

of massive repeated loop operations, unrolling and tiling can be used to weaken off-chip communication and deal with parallel computation problems. A more detailed optimization was presented in [23].

- Loop unrolling: Unrolling executes a network or multiple layers in parallel—particularly convolutional layers. The network can be fully expanded to achieve massively parallel processing or apply appropriate unrolling factors (iterations in the loop) across different layers for partial unrolling in the for-loop to optimize the datapath and maximize the throughput [65,70]. Ma et al. [72] adopted four types of loop unrolling in kernel maps and feature maps to determine the parallelism scheme and maximize data reuse. In an experiment involving VGG-16 on an Arria 10 FPGA, a throughput of 645.25 GOPS was achieved.
- Loop tiling: Constrained by limited on-chip memories, the data to be processed are tiled into multiple tiles and stored in on-chip buffers. Selecting a suitable tiling size factor can determine the trade-off between resources and the required external memory bandwidth. For example, Ma et al. [73] designed an auto-compilation process based on RTL, which uses intra-block and inter-block strategies to divide the layer execution into multiple sequential tiles. The process designed in [74] supports both unrolling and tiling of input and output feature maps on binarised networks. A 2× area efficiency improvement was achieved compared with existing binarised networks.

4. Accelerating DNNs from local to virtualized FPGAs in the cloud

The work of accelerating DNNs on FPGAs in our surveys covers local to the cloud and integrates the virtualization technique. The metrics used to evaluate these methods usually include throughput, power, and accuracy. Additionally, the adoption of virtualization techniques introduces additional characteristics such as portability and productivity, and in the cloud environment, QoS and isolation are regarded as new characteristics.

4.1. DNNs on local FPGA

Early studies (e.g., [63,75,76]) were dedicated to manually mapping a DNN model to a local FPGA with a streaming architecture. These studies take full advantage of DNNs parallelism and apply layer-independent optimization strategies to fit the entire network into the FPGA.

Benefiting from the well-defined structure of modern DNNs, which contain similar layers with repetitive operations, researchers have proposed frameworks with a single-engine computation structure [54,69,70,77,77–79], as shown in Fig. 6. These frameworks take advantage of both software programmability and flexible hardware structures, making DNN implementation more diversified and achieving high performance with reduced resource consumption. More frameworks that automatically map single DNNs to local FPGAs were presented in [52]. Another new type of framework is a toolchain that includes a compiler [65,71,80]. The compiler is a DNN architecture-aware tool that can map a wide range of DNN applications to the instruction set architecture (ISA) and control signals [65]. Fig. 7 presents an example of a compiler-inspired toolchain. Wang et al. [71] proposed a compiler that transforms a DNN deployment into a graph-level problem. The compiler first takes the software description as input and then transforms the description into directed acyclic graphs of computational operations. The networks generated by the compiler on Xilinx ZU9 reach throughputs of 2.82 TOPs/s (VGG), 1.38 TOPs/s (ResNet50), and 1.41 TOPs/s (GoogleNet).

More works are presented in Table 2.

4.2. DNNs on local virtualized FPGA

FPGA virtualization bridges the gap between the hardware stack and the software stack with the abstraction layer, enhancing the productivity and portability of DNN applications. Virtualization also enables

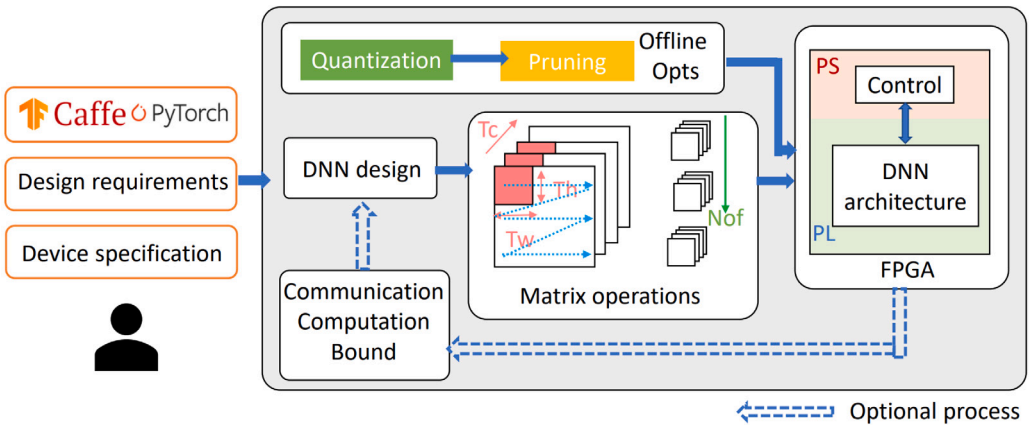

Fig. 6. Generic frameworks for DNN accelerators.

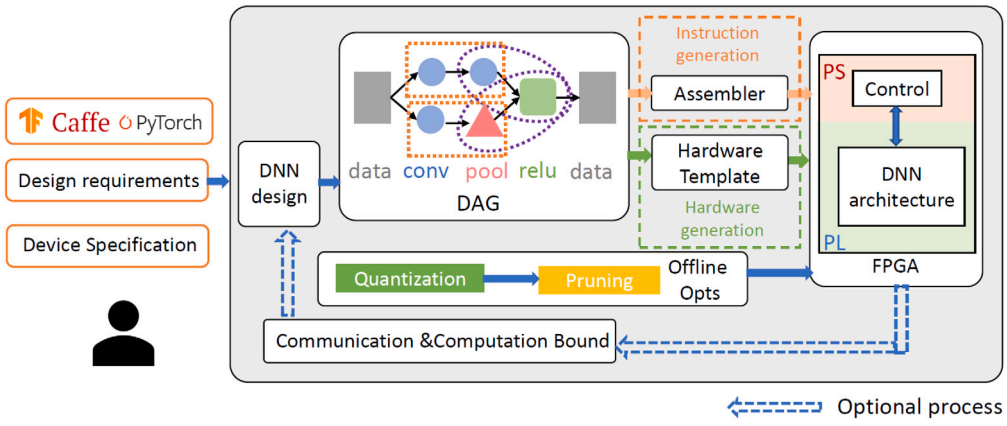

Fig. 7. Generic compiler-inspired frameworks for DNN accelerators.

resource sharing among multiple FPGAs with flexible resource management to support a wide range of DNNs. Fig. 8 shows an example of virtualization at the node level.

At the resource level of virtualization, Tong et al. [81] propose a coarse-grained overlay-based framework for quantizing and accelerating a DNN with any data width on an FPGA. The coarse-grained array comprises a reconfigurable NoC, a scheduler, and network computation components and is configured as DNN models according to instructions generated by the compiler of the framework. Such an overlay is independent of FPGA features and can be flexibly adapted to FPGAs provided by different vendors. To satisfy the metrics in virtualization, e.g., reducing the time and complexity of DNN reconfiguration, this framework reconfigures the coarse-grained array from the rightmost column to the leftmost column. The results indicate that the inference of AlexNet and VGG-16 on Xilinx UltraScale+ VCU118 takes only 0.13 and 2.63 ms, respectively.

Similarly, Struharik et al. [82] designed a coarse-grained overlay-based accelerator consisting of a set of processing blocks, which enabled on-the-fly reconfiguration for different DNNs. The accelerator can implement mainstream DNN families, such as VGG, Inception, ResNet, MobileNet, and NASNet, with a frame rate up to 6.05 times higher than that of Nullhop [83]. Other methods [84,85] also employ a coarse-grained overlay on top of the FPGA to enable dynamic datapath reconfiguration of DNN applications at runtime.

In contrast to previous studies where DNNs were deployed on FPGAs using the coarse-grained overlay, several researchers adopted a fine-grained overlay as an abstraction level to achieve higher flexibility. Venieris et al. [86] proposed an automated framework for implementing multiple DNNs on a target FPGA platform with fast space exploration. The framework adopts a streaming architecture to allocate resources at a fine-grained granularity for exploring a wide range of resource and bandwidth allocations. The authors tested their framework in a multi-DNN system (ZFNet, VGG16, SceneLabelCNN) on Xilinx ZC706, and the results indicated that the framework achieved an improvement of up to 6.8× in performance/W over Nvidia Tegra X1. Table 3 presents the features of the overlays used in the previous studies.

Table 3
Comparison of coarse- and fine-grained overlays on FPGAs for DNN acceleration.

	Coarse-grained overlay	Fine-grained overlay
Logic level	RT level	Gate level
Data width	Up to 32 bit	1 bit
Example logic	DNN function unit (e.g., Conv)	Control instruction (e.g., Load)
Goals	Opitimise DNN datapath switch	Enable DNN variability
Advantage	Area-efficiency	Higher flexibility

In node-level virtualization, the resource of a single FPGA can be allocated to a single DNN application or multiple DNN applications in TDM or SDM [87]. Zhang et al. [88] developed an end-to-end framework called a DNN builder to build DNNs with high performance using a design space exploration strategy. The DNN builder enables virtualization on a single physical FPGA by allocating resources to several small accelerating engines. The resource allocator can generate parallel schemes and data buffering guidelines for each layer. The tool deploys AlexNet, ZF, VGG16, and YOLO on Xilinx XC7Z045 and KU115 and achieves up to 5.15× better performance than that reported in [89].

At a multi-node level, allocating resources from multiple FPGAs to the DNN application may result in performance degradation owing to insufficient off-chip bandwidth. Therefore, it is essential to employ optimized resource mapping and efficient communication for this virtualization level. Zhang et al. [90] enabled large-scale DNN application implementation across up to 16 FPGAs with resource- and bandwidth-aware mapping methods. Taking the FPGA topology, resource conditions, and neural-network specifications as the inputs, this method can partition the DNN application to each FPGA depending on the statuses of the FPGAs (busy or free) and the estimation throughput of layer mapping. Results indicated that ResNet-152 on a multi-FPGA architecture outperformed a single-FPGA deployment by a factor of 16.4. Geng et al. [91] developed a framework that adopts a pipelined architecture to train DNNs on multiple FPGAs with a one-dimensional topology. The pipelined architecture with the fine-grained inter- and intra-layer methodology minimizes the time required for storing the feature map in the memory during training. The authors evaluated their framework by training AlexNet on 10 Xilinx VC709 Connectivity Kits. The results indicated that compared with other frameworks [21], the throughput obtained by this framework was increased by a factor of 5; compared with Titan X, the energy efficiency of the framework was up to 7.6 times higher. Moreover, the framework exhibits good scalability, as it can scale up to 60 FPGAs to accelerate DNNs.

However, such multiple-FPGA platforms adopting pipeline models gain high throughput while sacrificing latency. Jiang et al. [92] developed a general framework called Super-LIP to support concurrent processing for both single- and multi-layer deployment on FPGAs. To achieve communication between two FPGAs, the authors employ a novel methodology in Super-LIP to achieve linear speedup by balancing computation workloads and distributing the shared data across FPGAs to avoid traffic heaviness on the FPGA memory bus. Compared with the existing single-FPGA design [60], this method achieved a 3.48× speedup of AlexNet, VGG, and YOLO on two Xilinx ZCU102 kits.

4.3. DNNs on virtualized FPGA in the cloud

Zeng et al. [44] proposed a framework using FPGA virtualization, which is applicable to any DNN accelerator based on the ISA in a cloud environment. This principle divides a large resource pool into multiple virtualised cores to share FPGA resources at the node level. By introducing a novel two-level instruction (dispatch module and tiling-based instruction package design), virtualized multi-core resources can be dynamically allocated to each block in one DNN (single-task mode) or each DNN for multiple users (multi-task mode) at runtime. Compared with previous methods, this technique solves physical resource isolation and performance among multiple users by sharing FPGA resources in the SDM method. Experiments on VGG-16, ResNet50, Inception V3, and MobileNet indicated that compared with a single non-virtualized core design, the throughput of the proposed virtualization method with multiple cores was 1.07–1.69 times higher overall.

A similar method called ViTAL was developed by Zha et al. [93] to enable FPGA virtualization in a cloud environment for deploying DNNs. This method supports resource sharing at both the node and multi-node levels. ViTAL provides an abstraction layer between DNN applications and physical resources, which abstracts heterogeneous resources into homogeneous resources and provides a view of virtual blocks. The abstraction layer divides a DNN application into virtual blocks and then maps these virtual blocks to an FPGA or multiple FPGAs without impacting other running DNNs. By using a latency-insensitive interface, virtual blocks can be mapped across FPGAs at the multi-node level to achieve timing closure and match communication delays. Additionally, isolation in the cloud environment is achieved by avoiding the sharing of physical resources among different virtual blocks. The authors evaluated ViTAL by implementing LeNet, AlexNet, and VGG-16 on a Xilinx UltraScale+ FPGA. The experimental results indicated that ViTAL achieved good DNN mapping quality with a short compilation time (1.6% of the total). Furthermore, ViTAL can dynamically relocate the DNNs to different positions in the FPGA. The experimental results also indicated that with FPGA virtualization methods, ViTAL significantly shortened the response time (by 82%) in the cloud environment.

Fowers et al. [94] proposed a full-system architecture with virtualization at a multi-node level to serve DNN inferences in a cloud environment. The critical feature of the architecture is the dedicated neural processing units (NPUs), which implement an SIMD ISA containing a matrix–vector multiplier. This DNN-specific ISA offers a high-level abstraction between the underlying FPGA infrastructure and DNN software development, thereby simplifying FPGA programming for software developers. The authors validated the architecture by running RNNs and compared it with the NVIDIA Titan GPU, and it gained more than 36 effective teraflops (10 instances NPU). Moreover, the authors evaluated ResNet-50 on the Arria 10 GX 1150, which achieved 559 inferences per second (IPS), whereas ResNet-50 on the Nvidia P40 GPU achieved only 461 IPS.

4.4. DNN deployment in commercial cloud

In recent years, companies such as Amazon F1 [95], Tentent [96], Huawei FACs [97], and Microsoft [98] have launched cloud projects that provide FPGA IaaS for users to rent FPGA resources. Researchers have begun to accelerate DNN workloads in these commercial clouds to improve performance. The framework of deploying DNNs on FPGAs with a commercial cloud as the backend is similar to local deployment, but virtualization and physical connections of FPGAs are often charged by cloud vendors and hidden in the backend.

Several frameworks [99–101] have been proposed to implement DNNs on a single physical FPGA in the cloud with Caffe and TensorFlow as a frontend. Later, the research focus of DNN deployments moved from per-FPGA granularity to multiple FPGAs. Because the mainstream compilation tools do not support application implementation among multiple FPGASs, particular mapping algorithms or customized tools designed by the researchers are needed. Shan et al. [102] proposed an effective solution for implementing DNNs among multiple FPGAs in an AWS instance. The solution, which is based on the characteristics of FPGAs in the AWS, uses a heuristic method to find the global execution throughput between the CPU and the connected FPGA and then uses an allocation algorithm (including group kernel allocation and individual kernel allocation) to assign DNN workloads to various FPGAs with resource constraints. It is suitable for deploying any DNN to the AWS F1. Compared with the traditional mixed-integer nonlinear programming solution, this solution achieved faster DNN implementations on multiple FPGAs: 16-bit fixed-point AlexNet on two FPGAs, 32-bit floating-point AlexNet on four FPGAs, 16-bit fixed-point VGG-16 on four or six FPGAs, and ResNet on five FPGAs.

Table 4 presents studies on virtualization technology and the cloud environment.

5. Trends of DNN accelerators

As shown in Fig. 9, the first stage in the evolution of DNN accelerators involved manually mapping a single DNN to a single local FPGA with low energy consumption. DNN accelerators were designed for implementation on specific FPGA families. The optimization strategies are customized for a particular DNN and are not compatible with other networks. Therefore, DNN deployment has disadvantages, such as poor portability, time-consuming deployment, complex optimization, and inflexibility. Efforts have been made to automatically generate DNN hardware structures according to the requirements of different FPGA families. Therefore, researchers have proposed several frameworks to support a generic DNN accelerator and to offer customized DNN implementations by analysing requirements and platform-specific constraints. These frameworks are usually integrated with an RTL compiler with full exploitation of low-level structures to achieve high

Table 4
Several examples of DNNs based on local virtualized FPGAs and DNNs in the cloud.

Works	Years	DNN		Virtualization			Cloud	Multi-tenent	Task and resources Manager	FPGA platform	
		Model	Training	Abstraction	Host	Shell				Device	Num.
[21]	2016	AlexNet, VGG-16	N/A	Multi-node level	RH	AXI, Network access	N/A	N/A	system controller	Xilinx Virtex VC709	6
[34]	2018	LSTM, RNN	N/A	Multi-node level	RH	Network communication, PCIe controller	N/A	N/A	Model parallelism, On-chip pinning	Altera Stratix 10 280	1
[91]	2018	AlexNet	16-bit	Multi-node level	LH	Communication, I/O component	N/A	N/A	Partitioning, Memory subsystem	Xilinx Virtex VC709	10
[103]	2018	VGG-16, AlexNet, SqueezeNet, YOLO		Multi-node level	LH	Communication, PCIe, Aurora	N/A	N/A	Mixed integer linear programming	Xlinx XC7Z015 /XC7Z045 /XCZU9EG	1
[104]	2019	AlexNet, DispNet, ResNet, GoogLe-Net	N/A	Resource level	OC	AXI, Memory controller	N/A	N/A	Tuning algorithm	Xilinx ZC706, ZCU102	1
[92]	2019	AlexNet, Squeeze-Net, YOLO, VGG-16	N/A	Multi-node level	OC	Host communication, clock generator	N/A	N/A	Hypervisor	Xilinx ZCU102	2
[90]	2019	ResNet-152	N/A	Multi-Node level	RH	Network communication	N/A	N/A	Dynamic partitioning	Xilinx UltraScale	16
[105]	2019	Custom-ized 3D CNN	N/A	Multi-Node level	LH	Network interface, PCIe controller	N/A	N/A	Hardware monitor, mapping table	Xilinx VCU118	4
[106]	2019	Squeeze-Net, GoogLe-Net, VGG-16	32-bit FP*	Resource level	LH	DDR controller, Global memory interconnection	N/A	N/A	Partial reconfiguration manager	Intel Stratix 10 SX SoC	1
[81]	2020	VGG-16, AlexNet	N/A	Resource level	OC	DDR controller	N/A	N/A	Coarse-grained NoC, Parameter scheduler	Xilinx VCU118	1
[107]	2018	DNN Weaver	N/A	Node level	LH	PCIe controller, DMA, MMIO	Cloud environment	N/M	Zone manager on host CPU	Altera Stratix V GS, Xilinx Ultrascale+	1
[34]	2018	LSTM, RNN	N/A	Multi-node level	RH	Network communication, PCIe controller	Cloud environment	SM, TM	Resource runtime manager	Altera Stratix 10 280	1
[108]	2020	CIFAR-Net	N/A	Node level	OC	Memory controller, AXI	Cloud environment	SM, TM	Runtime task manager, scheduling decision	Xilinx ZCU104	1
[93]	2020	NiN, AlexNet, OverFeat, Vgg-16	N/A	Multi-node level	RH	Latency-intensive interface, address translation	Cloud environment	TM	Hypervisor and system controller	Xilinx UltraScale+	4
[44]	2020	ResNet-50, Inception V3, MobileNet	N/A	Node level	LH	Virtualization infrastructure	Cloud environment	SM	Multi-level instructions, virtualization manager	Xilinx Alveo U200	1
[54]	2017	VGG-19, ResNet-152, LSTM	32-bit FP*, 16-bit	N/M, Charged by cloud vendors			IaaS (Per-FPGA)	N/A	Symbolic compiler	Catapult	1
[109]	2018	MLP, YOLO, DoReFa-Net	N/A	N/M, Charged by cloud vendors			IaaS (Per-FPGA)	N/A	Data Flow Balancing algorithm	AWS F1	1
[99]	2018	LeNet, VGG-16	N/A	N/M, Charged by cloud vendors			IaaS (Per-FPGA)	N/A	Datamover, system controller	AWS F1	1
[100]	2019	AlexNet, VGG-16, ResNet-50	N/A	N/M, Charged by cloud vendors			IaaS (Per-FPGA)	N/A	Model split, task allocation manager	AWS F1	1
[110]	2020	AlexNet, VGG-16	N/A	N/M, Charged by cloud vendors			IaaS (multi-FPGAs)	N/A	MINLP Solver	AWS F1 and F2	1
[111]	2015	DenseNet-121, ResNet-152, etc	N/A	N/M			SaaS (Per-FPGA)	SM, TM	Web service API	Arria10	1, 2, 4

* FP = floating point format; OC = On-chip host; RH = Remoted host; LH = Local host; (Host in Section 2.2.2); N/A = Not applied;
N/M = Not mentioned; SM = Spatial multiplexing; TM = Time multiplexing.

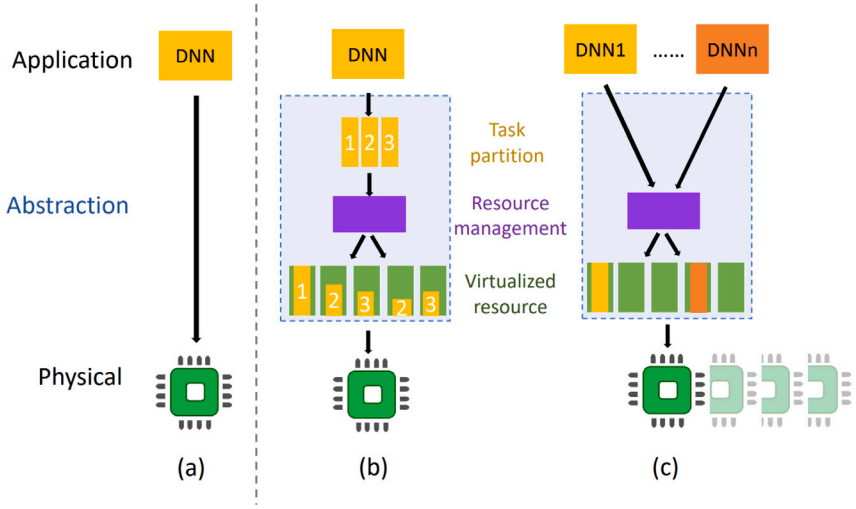

Fig. 8. (a) DNN deployment without virtualization. (b) Example of FPGA virtualization at the node level for deploying one DNN on a local FPGA. (c) Example of FPGA virtualization at the node level for deploying several DNNs in the cloud environment.

performance. Moreover, instruction-driven compiler frameworks have been developed in recent years to simplify the control flow of DNNs.

Despite enjoying energy efficiency and acceleration, DNN deployment on FPGAs faces complexity, resulting in reduced productivity. The framework of the previous stage mainly reduces the programmable complexity at the single-FPGA level without multiple tasks, and researchers have not yet determined how to improve the productivity of DNN implementation at the multiple-FPGA or resource level. Accordingly, DNN accelerators with virtualization techniques are being developed. Resource-level virtualization provides portability of DNN deployment for various families of FPGAs from different vendors. Node-level and multi-node-level virtualization enables resource sharing among FPGAs. Multi-node level virtualization exhibits the advantages of scaling up DNNs and training DNNs.

Subsequently, several works proposed cloud-based accelerators for deploying DNNs on-demand. These studies can be divided into two categories. The first category involves building an end-to-end cloud environment for DNN acceleration. These works not only require the development of a framework or a solution for DNN deployment but are also responsible for providing FPGA devices, virtualizing FPGAs, managing FPGA resources, scheduling tasks, and supporting multi-tenant scenarios with resources and data isolation. However, these works are still in their infancy and face obstacles, such as runtime overhead. Few researchers have performed studies in this area, but it will be an appealing field owing to the growing focus on cloud computing. The other category involves using the commercial FPGA cloud as a backend to develop DNN frameworks or solutions. These frameworks usually cannot consider multi-tenant solutions and cannot support DNN deployment at runtime. Additionally, FPGA management and virtualization are handled by the cloud provider and hidden in the background. Studies have mainly focused on deploying DNNs at per-FPGA granularity because this does not require resource-mapping algorithms or compilation tools across multiple FPGAs. DNN development can only be completed by using cloud integrators provided by cloud vendors and mainstream compilation tools.

At each stage, DNN deployment exhibits various characteristics, as shown in Fig. 10. Most DNN implementations are based on streaming or a single computation engine, along with the compression and optimization strategies mentioned in Section 3. Compared with a single computation engine, the streaming architecture gains efficiency by pipelining the network and activating concurrent executions between layers. However, this efficiency leads to a resource burden and a long recompilation time because obtaining a new DNN model requires regenerating the bitstream. According to the different requirements (e.g., resource constraints or speedup), researchers can choose different hardware structures in both the local and cloud FPGAs.

6. Discussion

In the history of deploying DNNs on FPGAs, new requirements have been proposed at different stages, which has led to different challenges. With the development of a novel generation of platforms, technologies, and concepts, challenges have been resolved.

6.1. Unresolved challenges

Some challenges of using FPGAs in the cloud have not been fully resolved owing to their complexity. Here, we describe two major challenges: isolation and diversity.

6.1.1. Isolation

With the increasing efforts to provide a cloud environment for multiple tenants to deploy DNNs on the shared FPGAs, resources and performance isolation have become a concern in the cloud.

DNN accelerators on the FPGA usually run under full hardware access and may share resources. Therefore, malicious code can attack the entire platform for other tenancies [112,113]. Additionally, dataset collection can be time-consuming and expensive—particularly in industrial cases where datasets are of significant commercial value. Providing strict data and resource isolation for multiple tenants can prevent unauthorized access to the dataset and avoid data leakage [114,115].

Additionally, a DNN application may affect the performance of other DNN applications during concurrent execution [112,116], which causes unreliable performance. However, few works [44,93] discuss performance isolation problems, and their isolation remains underexplored.

6.1.2. Diversity

Diversity of DNN functions: Owing to resource limitations and development difficulties, the networks reported in the literature are standard (such as AlexNet and VGG) with common functions (such as convolution and pooling). With the continuous emergence of DNNs, the current DNN functions that can be implemented on FPGAs lack consistency with the development of DNN algorithms. However, the cloud environment provides more possibilities for exploring the deployment of DNNs with a rich set of functions on FPGAs by providing more resources and abstraction layers and can promote the diversity of DNN IP development.

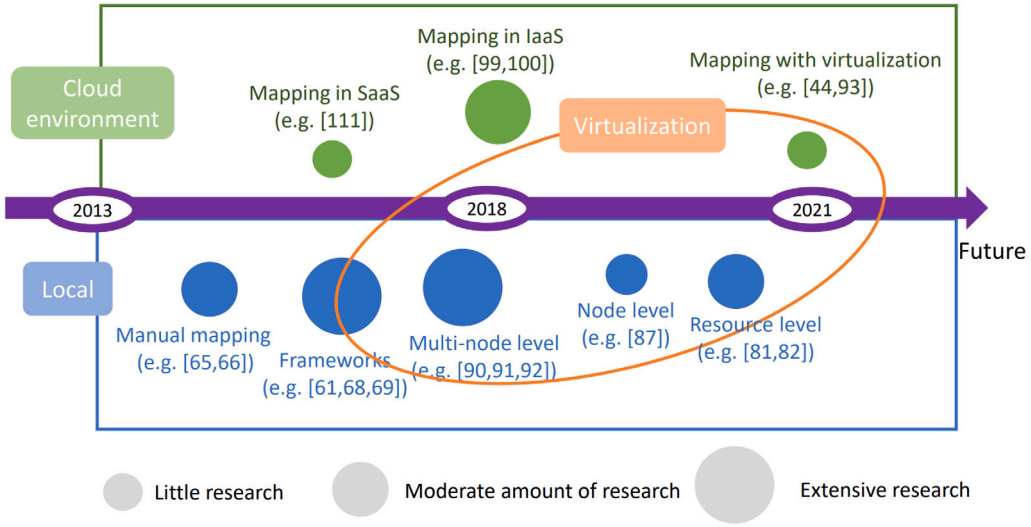

Fig. 9. Evolution of DNN accelerators at each time node: from manual mapping to frameworks, from a single node to a cluster, from physical to virtual resources, and from local to cloud.

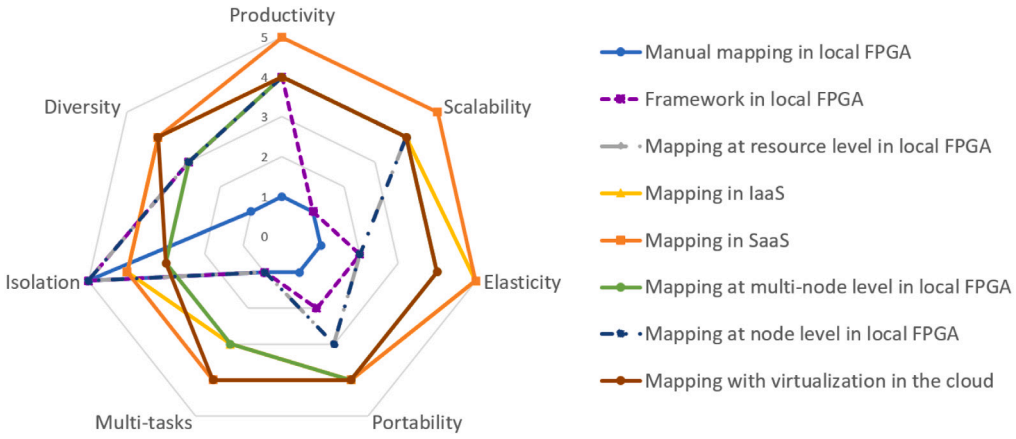

Fig. 10. Comparison of related methods with different characteristics.

Diversity of DNN usage: Training is a difficult phase to be performed on the FPGA, because all the features must be stored in memory until the corresponding errors are backpropagated, which requires more storage than inference. Existing works mainly focus on performing DNN inferences with relatively simple functions on the FPGA. Benefiting from the "unlimited" capacity and resources provided by the FPGA cloud, DNN training, fine-tuning, transfer learning, and the support of new functions in DNNs will be more feasible.

6.2. Industrial solution

To keep pace with the development of DNN accelerator design, novel platforms have been used in industry to enhance the hardware computing power. In 2019, Xilinx proposed a new SoC family called Versal, which is based on an adaptive compute acceleration platform, for accelerating applications such as DNNs. Versal tightly integrates software-programmable accelerators through the NoC structure, making accelerators scalable with flexible connections and achieving a high level of software abstraction for the rapid development of accelerators.

Xilinx also proposed a novel framework called Vitis AI [117]. The framework can be interfaced with Caffe and TensorFlow and provides a unified solution, e.g., quantization, optimization, and pruning. Moreover, it allows the deployment of DNNs based on the ISA and can compile the latest DNNs into deep-learning processor unit instruction codes. Vitis AI can enhance the productivity and portability of DNN deployment, allowing software engineers to deploy DNNs without hardware expertise.

6.3. Roadblocks of FPGA cloud

Solutions of FPGA-based accelerators in the cloud have been proposed for several years [95,97]. Nevertheless, FPGAs have achieved less success compared to GPU and TPU architectures in the cloud. Deploying FPGA devices as easy-to-use resources in the cloud faces the following major roadblocks.

First, FPGA programming requires cloud users to have extensive hardware skills and expertise to deploy their applications in the cloud, which is a considerable challenge for software engineers and data scientists. Cloud providers must provide well-developed virtualization techniques for abstracting FPGAs [118]. As discussed in Section 5, virtualizing FPGAs in the cloud for artificial-intelligence applications still has issues, such as runtime overhead, multi-user support, user isolation, and data privacy. Additionally, the FPGA cloud provides users with high permissions to access the resources, where users can upload their bitstreams for application deployment, leading to malicious attacks and

security problems [119]. Such problems hinder the success of FPGAs in cloud computing.

7. Conclusion

This paper summarizes several techniques to promote DNN deployments on FPGAs, including architectural design and optimization strategies. We reviewed related works based on FPGA virtualization and cloud deployment. Our study involved an in-depth analysis of the evolution of DNN deployment on FPGAs, from local FPGAs to virtualized FPGAs in the cloud. This topic was ignored by previous surveys.

With the rising concern regarding the adoption of FPGAs at the edge and in the cloud, porting DNNs onto FPGAs in cloud services will continue to attract attention in the years to come.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The first author is funded by China Scholarship Council (Grant number, 201708070009).

References

[1] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, http://dx.doi.org/10.1145/3065386, https://doi.org/10.1145/3065386.

[2] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015, http://arxiv.org/abs/1409.1556.

[3] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778, http://dx.doi.org/10.1109/CVPR.2016.90.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2017, https://arxiv.org/pdf/1706.03762.pdf.

[5] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, 2014, arXiv preprint arXiv:1406.2661.

[6] D.P. Kingma, M. Welling, Auto-encoding variational bayes, 2013, arXiv preprint arXiv:1312.6114.

[7] J. Powles, H. Hodson, Google DeepMind and healthcare in an age of algorithms, Health Technol. 7 (4) (2017) 351–367, http://dx.doi.org/10.1007/s12553-017-0179-1.

[8] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, X. Wang, Applied machine learning at facebook: A datacenter infrastructure perspective, in: 2018 IEEE International Symposium on High Performance Computer Architecture, HPCA, 2018, pp. 620–629, http://dx.doi.org/10.1109/HPCA.2018.00059.

[9] I. Lopatovska, K. Rink, I. Knight, K. Raines, K. Cosenza, H. Williams, P. Sorsche, D. Hirsch, Q. Li, A. Martinez, Talk to me: Exploring user interactions with the amazon alexa, J. Librarianship Inf. Sci. 51 (4) (2019) 984–997, http://dx.doi.org/10.1177/0961000618759414, https://doi.org/10.1177/0961000618759414.

[10] M. Song, Y. Hu, H. Chen, T. Li, Towards pervasive and user satisfactory CNN across GPU microarchitectures, in: 2017 IEEE International Symposium on High Performance Computer Architecture, HPCA, 2017, pp. 1–12, http://dx.doi.org/10.1109/HPCA.2017.52.

[11] S. Potluri, A. Fasih, L.K. Vutukuru, F.A. Machot, K. Kyamakya, CNN Based high performance computing for real time image processing on GPU, in: Proceedings of the Joint INDS'11 ISTET'11, 2011, pp. 1–7, http://dx.doi.org/10.1109/INDS.2011.6024781.

[12] N.P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T.V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C.R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, D.H. Yoon, In-datacenter performance analysis of a tensor processing unit, SIGARCH Comput. Archit. News 45 (2) (2017) 1–12, http://dx.doi.org/10.1145/3140659.3080246, https://doi.org/10.1145/3140659.3080246.

[13] N.P. Jouppi, C. Young, N. Patil, D. Patterson, A domain-specific architecture for deep neural networks, Commun. ACM 61 (9) (2018) 50–59, http://dx.doi.org/10.1145/3154484, https://doi.org/10.1145/3154484.

[14] S.I. Venieris, I. Panopoulos, I. Leontiadis, I.S. Venieris, How to reach real-time AI on consumer devices? Solutions for programmable and custom architectures, 2021, arXiv preprint arXiv:2106.15021.

[15] Intel® Stratix® 10 variable precision DSP blocks user guide.

[16] Y. Zhou, J. Jiang, An FPGA-based accelerator implementation for deep convolutional neural networks, in: 2015 4th International Conference on Computer Science and Network Technology, ICCSNT, vol. 01, 2015, pp. 829–832, http://dx.doi.org/10.1109/ICCSNT.2015.7490869.

[17] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, Y. Cao, Throughput-optimized opencl-based FPGA accelerator for large-scale convolutional neural networks, in: Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 16–25, http://dx.doi.org/10.1145/2847263.2847276, https://doi.org/10.1145/2847263.2847276.

[18] L.B. Saldanha, C. Bobda, An embedded system for handwritten digit recognition, J. Syst. Archit. 61 (10) (2015) 693–699, Special section on Architecture of Computing Systems edited by Editors: Wolfgang Karl, Erik Maehle, Kay Römer, Eduardo Tovar, Martin Danek Special section on Testing, Prototyping, and Debugging of Multi-Core Architectures edited by Editors: Frank Hannig and Andreas Herkersdorf Special section on Embedded Vision Architectures and Applications edited by Editors: Christophe Bobda, Walter Stechele, Ali Ahmadinia and Miaoqing Huang, https://doi.org/10.1016/j.sysarc.2015.07.015, URL https://www.sciencedirect.com/science/article/pii/S1383762115000867.

[19] T. Fanni, L. Li, T. Viitanen, C. Sau, R. Xie, F. Palumbo, L. Raffo, H. Huttunen, J. Takala, S.S. Bhattacharyya, Hardware design methodology using lightweight dataflow and its integration with low power techniques, J. Syst. Archit. 78 (2017) 15–29, http://dx.doi.org/10.1016/j.sysarc.2017.06.003, URL https://www.sciencedirect.com/science/article/pii/S1383762116302831.

[20] F. Li, B. Liu, Ternary weight networks, 2016, arXiv arXiv:abs/1605.04711.

[21] C. Zhang, D. Wu, J. Sun, G. Sun, G. Luo, J. Cong, Energy-efficient CNN implementation on a deeply pipelined FPGA cluster, in: Proceedings of the 2016 International Symposium on Low Power Electronics and Design, ISLPED '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 326–331, http://dx.doi.org/10.1145/2934583.2934644, https://doi.org/10.1145/2934583.2934644.

[22] R. DiCecco, G. Lacey, J. Vasiljevic, P. Chow, G. Taylor, S. Areibi, Caffeinated FPGAs: FPGA framework for convolutional neural networks, in: 2016 International Conference on Field-Programmable Technology, FPT, 2016, pp. 265–268, http://dx.doi.org/10.1109/FPT.2016.7929549.

[23] S. Mittal, A survey of FPGA-based accelerators for convolutional neural networks, Neural Comput. Appl. 32 (4) (2020) 1109–1139, http://dx.doi.org/10.1007/s00521-018-3761-1.

[24] K. Guo, S. Zeng, J. Yu, Y. Wang, H. Yang, [Dl] a survey of FPGA-based neural network inference accelerators, ACM Trans. Reconfigurable Technol. Syst. 12 (1) (2019) http://dx.doi.org/10.1145/3289185, https://doi.org/10.1145/3289185.

[25] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access 6 (2018) 64270–64277, http://dx.doi.org/10.1109/ACCESS.2018.2877890.

[26] Z. Li, Y. Zhang, J. Wang, J. Lai, A survey of FPGA design for AI era, J. Semicond. 41 (2020) 021402, http://dx.doi.org/10.1088/1674-4926/41/2/021402.

[27] A.G. Blaiech, K. Ben Khalifa, C. Valderrama, M.A. Fernandes, M.H. Bedoui, A survey and taxonomy of FPGA-based deep learning accelerators, J. Syst. Archit. 98 (2019) 331–345, http://dx.doi.org/10.1016/j.sysarc.2019.01.007, URL https://www.sciencedirect.com/science/article/pii/S1383762118304156.

[28] D. Moolchandani, A. Kumar, S.R. Sarangi, Accelerating CNN inference on ASICs: A survey, J. Syst. Archit. 113 (2021) 101887, http://dx.doi.org/10.1016/j.sysarc.2020.101887, URL https://www.sciencedirect.com/science/article/pii/S1383762120301612.

[29] O. Djedidi, M.A. Djeziri, Power profiling and monitoring in embedded systems: A comparative study and a novel methodology based on NARX neural networks, J. Syst. Archit. 111 (2020) 101805, http://dx.doi.org/10.1016/

j.sysarc.2020.101805, URL https://www.sciencedirect.com/science/article/pii/S1383762120300953.

[30] S. Salamat, B. Khaleghi, M. Imani, T. Rosing, Workload-aware opportunistic energy efficiency in multi-FPGA platforms, 2019, CoRR abs/1908.06519, arXiv:1908.06519.

[31] S. Byma, J.G. Steffan, H. Bannazadeh, A. Leon-Garcia, P. Chow, FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack, in: 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, 2014, pp. 109–116, http://dx.doi.org/10.1109/FCCM.2014.42.

[32] M. Asiatici, N. George, K. Vipin, S.A. Fahmy, P. Ienne, Virtualized execution runtime for FPGA accelerators in the cloud, IEEE Access 5 (2017) 1900–1910, http://dx.doi.org/10.1109/ACCESS.2017.2661582.

[33] A. Putnam, A.M. Caulfield, E.S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G.P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P.Y. Xiao, D. Burger, A reconfigurable fabric for accelerating large-scale datacenter services, IEEE Micro 35 (3) (2015) 10–22, http://dx.doi.org/10.1109/MM.2015.42.

[34] E. Chung, J. Fowers, K. Ovtcharov, M. Papamichael, A. Caulfield, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, M. Abeydeera, L. Adams, H. Angepat, C. Boehn, D. Chiou, O. Firestein, A. Forin, K.S. Gatlin, M. Ghandi, S. Heil, K. Holohan, A. El Husseini, T. Juhasz, K. Kagi, R.K. Kovvuri, S. Lanka, F. van Megen, D. Mukhortov, P. Patel, B. Perez, A. Rapsang, S. Reinhardt, B. Rouhani, A. Sapek, R. Seera, S. Shekar, B. Sridharan, G. Weisz, L. Woods, P. Yi Xiao, D. Zhang, R. Zhao, D. Burger, Serving DNNs in real time at datacenter scale with project brainwave, IEEE Micro 38 (2) (2018) 8–20, http://dx.doi.org/10.1109/MM.2018.022071131.

[35] Q. Ijaz, E.-B. Bourennane, A.K. Bashir, H. Asghar, Revisiting the high-performance reconfigurable computing for future datacenters, Future Internet 12 (4) (2020) http://dx.doi.org/10.3390/fi12040064, URL https://www.mdpi.com/1999-5903/12/4/64.

[36] A. Vaishnav, K.D. Pham, D. Koch, J. Garside, Resource elastic virtualization for FPGAs using OpenCL, in: 2018 28th International Conference on Field Programmable Logic and Applications, FPL, 2018, pp. 111–1117, http://dx.doi.org/10.1109/FPL.2018.00028.

[37] K. Vipin, S.A. Fahmy, FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications, ACM Comput. Surv. 51 (4) (2018) http://dx.doi.org/10.1145/3193827, https://doi.org/10.1145/3193827.

[38] R. Skhiri, V. Fresse, et al., From FPGA to support cloud to cloud of FPGA: State of the art, Int. J. Reconfigurable Comput. (2019).

[39] A. Vaishnav, K.D. Pham, D. Koch, A survey on FPGA virtualization, in: 2018 28th International Conference on Field Programmable Logic and Applications, FPL, 2018, pp. 131–1317, http://dx.doi.org/10.1109/FPL.2018.00031.

[40] M. Quraishi, E. Tavakoli, F. Ren, A survey of system architectures and techniques for FPGA virtualization, IEEE Trans. Parallel Distrib. Syst. 32 (09) (2021) 2216–2230, http://dx.doi.org/10.1109/TPDS.2021.3063670.

[41] O. Knodel, P.R. Genssler, R.G. Spallek, Virtualizing reconfigurable hardware to provide scalability in cloud architectures, in: International Conference on Advances in Circuits, Electronics and Micro-Electronics, CENICS, 2017.

[42] J. Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, Enabling FPGAs in hyperscale data centers, in: 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops, UIC-ATC-ScalCom, 2015, pp. 1078–1086, http://dx.doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.199.

[43] X. Li, D.L. Maskell, Time-multiplexed FPGA overlay architectures: A survey, ACM Trans. Des. Autom. Electron. Syst. 24 (5) (2019) http://dx.doi.org/10.1145/3339861, https://doi.org/10.1145/3339861.

[44] S. Zeng, G. Dai, K. Zhong, H. Sun, G. Ge, K. Guo, Y. Wang, H. Yang, Enable efficient and flexible FPGA virtualization for deep learning in the cloud, in: Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 317, http://dx.doi.org/10.1145/3373087.3375346, https://doi.org/10.1145/3373087.3375346.

[45] C. Xu, G. Liu, R. Zhao, S. Yang, G. Luo, Z. Zhang, A parallel bandit-based approach for autotuning FPGA compilation, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 157–166, http://dx.doi.org/10.1145/3020078.3021747, https://doi.org/10.1145/3020078.3021747.

[46] H.K.-H. So, C. Liu, FPGA overlays, in: D. Koch, F. Hannig, D. Ziener (Eds.), FPGAs for Software Programmers, Springer International Publishing, Cham, 2016, pp. 285–305, http://dx.doi.org/10.1007/978-3-319-26408-0_16, https://doi.org/10.1007/978-3-319-26408-0_16.

[47] C. Lu, K. Ye, G. Xu, C.-Z. Xu, T. Bai, Imbalance in the cloud: An analysis on alibaba cluster trace, in: 2017 IEEE International Conference on Big Data, Big Data, 2017, pp. 2884–2892, http://dx.doi.org/10.1109/BigData.2017.8258257.

[48] K. Fleming, H.-J. Yang, M. Adler, J. Emer, The LEAP FPGA operating system, in: 2014 24th International Conference on Field Programmable Logic and Applications, FPL, 2014, pp. 1–8, http://dx.doi.org/10.1109/FPL.2014.6927488.

[49] A. Agne, M. Happe, A. Keller, E. Lübbers, B. Plattner, M. Platzner, C. Plessl, ReconOS: An operating system approach for reconfigurable computing, IEEE Micro 34 (1) (2014) 60–71, http://dx.doi.org/10.1109/MM.2013.110.

[50] S.A. Chin, K.P. Niu, M. Walker, S. Yin, A. Mertens, J. Lee, J.H. Anderson, Architecture exploration of standard-cell and FPGA-overlay CGRAs using the open-source CGRA-ME framework, in: Proceedings of the 2018 International Symposium on Physical Design, ISPD '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 48–55, http://dx.doi.org/10.1145/3177540.3177553, https://doi.org/10.1145/3177540.3177553.

[51] X. Li, K. Vipin, D.L. Maskell, S.A. Fahmy, A.K. Jain, High throughput accelerator interface framework for a linear time-multiplexed FPGA overlay, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, 2020, pp. 1–5, http://dx.doi.org/10.1109/ISCAS45731.2020.9181072.

[52] S.I. Venieris, A. Kouris, C.-S. Bouganis, Toolflows for mapping convolutional neural networks on FPGAs: A survey and future directions, ACM Comput. Surv. 51 (3) (2018) http://dx.doi.org/10.1145/3186332, https://doi.org/10.1145/3186332.

[53] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, H. 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, FPGA '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 26–35, http://dx.doi.org/10.1145/2847263.2847265, https://doi.org/10.1145/2847263.2847265.

[54] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, J. Cong, FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates, in: 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, 2017, pp. 152–159, http://dx.doi.org/10.1109/FCCM.2017.25.

[55] Q. Xiao, Y. Liang, L. Lu, S. Yan, Y.-W. Tai, Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on FPGAs, in: 2017 54th ACM/EDAC/IEEE Design Automation Conference, DAC, 2017, pp. 1–6, http://dx.doi.org/10.1145/3061639.3062244.

[56] P. Gysel, J. Pimentel, M. Motamedi, S. Ghiasi, Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks, IEEE Trans. Neural Netw. Learn. Syst. 29 (11) (2018) 5784–5789, http://dx.doi.org/10.1109/TNNLS.2018.2808319.

[57] C. Louizos, M. Reisser, T. Blankevoort, E. Gavves, M. Welling, Relaxed quantization for discretized neural networks, 2018, arXiv preprint arXiv:1810.01875.

[58] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient transfer learning, 2016, CoRR abs/1611.06440, arXiv:1611.06440.

[59] M. Zhang, L. Li, H. Wang, Y. Liu, H. Qin, W. Zhao, Optimized compression for implementing convolutional neural networks on FPGA, Electronics 8 (3) (2019) http://dx.doi.org/10.3390/electronics8030295, URL https://www.mdpi.com/2079-9292/8/3/295.

[60] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, J. 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, FPGA '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 161–170, http://dx.doi.org/10.1145/2684746.2689060, https://doi.org/10.1145/2684746.2689060.

[61] S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, JMLR.org, 2015, pp. 1737–1746.

[62] A. Rahman, J. Lee, K. Choi, Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array, in: 2016 Design, Automation Test in Europe Conference Exhibition, DATE, 2016, pp. 1393–1398.

[63] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, L. 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, 2016, pp. 1–9, http://dx.doi.org/10.1109/FPL.2016.7577308.

[64] X. Zhang, X. Liu, A. Ramachandran, C. Zhuge, S. Tang, P. Ouyang, Z. Cheng, K. Rupnow, D. Chen, High-performance video content recognition with long-term recurrent convolutional network for FPGA, in: 2017 27th International Conference on Field Programmable Logic and Applications, FPL, 2017, pp. 1–4, http://dx.doi.org/10.23919/FPL.2017.8056833.

[65] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, H. Yang, Angel-eye: A complete design flow for mapping CNN onto embedded FPGA, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (1) (2018) 35–47, http://dx.doi.org/10.1109/TCAD.2017.2705069.

[66] S.I. Venieris, C.-S. Bouganis, fpgaConvNet: A framework for mapping convolutional neural networks on FPGAs, in: 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, IEEE, 2016, pp. 40–47.

[67] Y. Umuroglu, N.J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, K. Vissers, FINN: A framework for fast, scalable binarized neural network inference, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 65–74, http://dx.doi.org/10.1145/3020078.3021744, https://doi.org/10.1145/3020078.3021744.

[68] V. Gokhale, A. Zaidy, A.X.M. Chang, E. Culurciello, Snowflake: An efficient hardware accelerator for convolutional neural networks, in: 2017 IEEE International Symposium on Circuits and Systems, ISCAS, 2017, pp. 1–4, http://dx.doi.org/10.1109/ISCAS.2017.8050809.

[69] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, J. Cong, Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 38 (11) (2019) 2072–2085, http://dx.doi.org/10.1109/TCAD.2017.2785257.

[70] Y. Ma, N. Suda, Y. Cao, S. Vrudhula, J. sun Seo, ALAMO: FPGA acceleration of deep learning algorithms with a modularized RTL compiler, Integration 62 (2018) 14–23, http://dx.doi.org/10.1016/j.vlsi.2017.12.009, URL https://www.sciencedirect.com/science/article/pii/S0167926017304777.

[71] Y. Xing, S. Liang, L. Sui, X. Jia, J. Qiu, X. Liu, Y. Wang, Y. Shan, Y. Wang, Dnnvm: End-to-end compiler leveraging heterogeneous optimizations on FPGA-based CNN accelerators, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (10) (2019) 2668–2681.

[72] Y. Ma, Y. Cao, S. Vrudhula, J.-s. 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, FPGA '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 45–54, http://dx.doi.org/10.1145/3020078.3021736, https://doi.org/10.1145/3020078.3021736.

[73] Y. Ma, Y. Cao, S. Vrudhula, J.-s. Seo, An automatic RTL compiler for high-throughput FPGA implementation of diverse deep convolutional neural networks, in: 2017 27th International Conference on Field Programmable Logic and Applications, FPL, 2017, pp. 1–8, http://dx.doi.org/10.23919/FPL.2017.8056824.

[74] E. Wang, J.J. Davis, P.Y.K. Cheung, G.A. Constantinides, LUTNet: Learning FPGA configurations for highly efficient neural network inference, IEEE Trans. Comput. 69 (12) (2020) 1795–1808, http://dx.doi.org/10.1109/TC.2020.2978817.

[75] K. ABDELOUAHAB, M. Pelcat, J. Sérot, C. Bourrasset, F. Berry, Tactics to directly map CNN graphs on embedded FPGAs, IEEE Embedded Syst. Lett. 9 (4) (2017) 113–116, http://dx.doi.org/10.1109/LES.2017.2743247, URL https://hal.archives-ouvertes.fr/hal-01626462.

[76] W. Ding, Z. Huang, Z. Huang, L. Tian, H. Wang, S. Feng, Designing efficient accelerator of depthwise separable convolutional neural network on FPGA, J. Syst. Archit. 97 (2019) 278–286, http://dx.doi.org/10.1016/j.sysarc.2018.12.008, URL https://www.sciencedirect.com/science/article/pii/S1383762118304612.

[77] H. Sharma, J. Park, D. Mahajan, E. Amaro, J.K. Kim, C. Shao, A. Mishra, H. Esmaeilzadeh, From high-level deep neural models to FPGAs, in: 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2016, pp. 1–12, http://dx.doi.org/10.1109/MICRO.2016.7783720.

[78] Z. Xu, R.C. Cheung, Binary convolutional neural network acceleration framework for rapid system prototyping, J. Syst. Archit. 109 (2020) 101762, http://dx.doi.org/10.1016/j.sysarc.2020.101762, URL https://www.sciencedirect.com/science/article/pii/S1383762120300564.

[79] L.-C. Hsu, C.-T. Chiu, K.-T. Lin, H.-H. Chou, Y.-Y. Pu, ESSA: An energy-aware bit-serial streaming deep convolutional neural network accelerator, J. Syst. Archit. 111 (2020) 101831, http://dx.doi.org/10.1016/j.sysarc.2020.101831, URL https://www.sciencedirect.com/science/article/pii/S1383762120301235.

[80] M.S. Abdelfattah, D. Han, A. Bitar, R. DiCecco, S. O'Connell, N. Shanker, J. Chu, I. Prins, J. Fender, A.C. Ling, et al., DLA: Compiler and FPGA overlay for neural network inference acceleration, in: 2018 28th International Conference on Field Programmable Logic and Applications, FPL, IEEE, 2018, pp. 411–4117.

[81] T. Geng, C. Wu, C. Tan, B. Fang, A. Li, M. Herbordt, CQNN: a CGRA-based QNN framework, in: 2020 IEEE High Performance Extreme Computing Conference, HPEC, 2020, pp. 1–7, http://dx.doi.org/10.1109/HPEC43674.2020.9286194.

[82] R.J. Struharik, B.Z. Vukobratović, A.M. Erdeljan, D.M. Rakanović, CoNNa–Hardware accelerator for compressed convolutional neural networks, Microprocess. Microsyst. 73 (2020) 102991, http://dx.doi.org/10.1016/j.micpro.2020.102991, URL https://www.sciencedirect.com/science/article/pii/S0141933119300158.

[83] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M.B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, T. Delbruck, NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps, IEEE Trans. Neural Netw. Learn. Syst. 30 (3) (2019) 644–656, http://dx.doi.org/10.1109/TNNLS.2018.2852335.

[84] S. Hadjis, K. Olukotun, TensorFlow to cloud FPGAs: Tradeoffs for accelerating deep neural networks, in: 2019 29th International Conference on Field Programmable Logic and Applications, FPL, 2019, pp. 360–366, http://dx.doi.org/10.1109/FPL.2019.00064.

[85] A. Arora, S. Mehta, V. Betz, L.K. John, Tensor slices to the rescue: Supercharging ML acceleration on FPGAs, in: The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 23–33, http://dx.doi.org/10.1145/3431920.3439282, https://doi.org/10.1145/3431920.3439282.

[86] S.I. Venieris, C.-S. Bouganis, f-CNNx: A toolflow for mapping multiple convolutional neural networks on FPGAs, in: 2018 28th International Conference on Field Programmable Logic and Applications, FPL, 2018, pp. 381–3817, http://dx.doi.org/10.1109/FPL.2018.00072.

[87] J. Ma, G. Zuo, K. Loughlin, X. Cheng, Y. Liu, A.M. Eneyew, Z. Qi, B. Kasikci, A hypervisor for shared-memory FPGA platforms, in: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 827–844, http://dx.doi.org/10.1145/3373376.3378482, https://doi.org/10.1145/3373376.3378482.

[88] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, D. Chen, DNNBuilder: an automated tool for building high-performance dnn hardware accelerators for FPGAs, in: 2018 IEEE/ACM International Conference on Computer-Aided Design, ICCAD, 2018, pp. 1–8, http://dx.doi.org/10.1145/3240765.3240801.

[89] H. Zeng, R. Chen, C. Zhang, V. Prasanna, A framework for generating high throughput CNN implementations on FPGAs, in: Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 117–126, http://dx.doi.org/10.1145/3174243.3174265, https://doi.org/10.1145/3174243.3174265.

[90] W. Zhang, J. Zhang, M. Shen, G. Luo, N. Xiao, An efficient mapping approach to large-scale DNNs on multi-FPGA architectures, in: 2019 Design, Automation Test in Europe Conference Exhibition, DATE, 2019, pp. 1241–1244, http://dx.doi.org/10.23919/DATE.2019.8715174.

[91] T. Geng, T. Wang, A. Sanaullah, C. Yang, R. Xu, R. Patel, M. Herbordt, FPDeep: Acceleration and load balancing of CNN training on FPGA clusters, in: 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM, 2018, pp. 81–84, http://dx.doi.org/10.1109/FCCM.2018.00021.

[92] W. Jiang, E.H. Sha, X. Zhang, L. Yang, Q. Zhuge, Y. Shi, J. Hu, Achieving super-linear speedup across multi-FPGA for real-time DNN inference, 2019, CoRR abs/1907.08985, arXiv:1907.08985.

[93] Y. Zha, J. Li, Virtualizing FPGAs in the cloud, in: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 845–858, http://dx.doi.org/10.1145/3373376.3378491, https://doi.org/10.1145/3373376.3378491.

[94] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S.K. Reinhardt, A.M. Caulfield, E.S. Chung, D. Burger, A configurable cloud-scale DNN processor for real-time AI, in: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture, ISCA, 2018, pp. 1–14, http://dx.doi.org/10.1109/ISCA.2018.00012.

[95] Amazon, Amazon EC2 F1, https://aws.amazon.com/fr/ec2/instance-types/f1/.

[96] Tencent, Tencent Cloud: Instance Type FPGA FX2, https://intl.cloud.tencent.com/document/product/213/11518FX2.

[97] Huawei, Huawei acceleration cloud server(FACS), https://www.huaweicloud.com.

[98] Microsoft, Microsoft Catapult, https://www.microsoft.com/en-us/research/project/project-catapult/.

[99] N. Raspa, G. Natale, M. Bacis, M.D. Santambrogio, A framework with cloud integration for CNN acceleration on FPGA devices, in: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops, 2018.

[100] Y. Chen, J. He, X. Zhang, C. Hao, D. Chen, Cloud-DNN: An open framework for mapping DNN models to cloud FPGAs, in: Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 73–82, http://dx.doi.org/10.1145/3289602.3293915, https://doi.org/10.1145/3289602.3293915.

[101] S. Tridgell, M. Kumm, M. Hardieck, D. Boland, D. Moss, P. Zipf, P.H.W. Leong, Unrolling ternary neural networks, ACM Trans. Reconfigurable Technol. Syst. 12 (4) (2019) http://dx.doi.org/10.1145/3359983, https://doi.org/10.1145/3359983.

[102] J. Shan, M.T. Lazarescu, J. Cortadella, L. Lavagno, M.R. Casu, CNN-on-AWS: Efficient allocation of multikernel applications on multi-FPGA platforms, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (2) (2021) 301–314, http://dx.doi.org/10.1109/TCAD.2020.2994256.

[103] W. Jiang, E.H.-M. Sha, Q. Zhuge, L. Yang, X. Chen, J. Hu, Heterogeneous FPGA-based cost-optimal design for timing-constrained CNNs, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (11) (2018) 2542–2554, http://dx.doi.org/10.1109/TCAD.2018.2857098.

[104] Q. Xiao, Y. Liang, Fune: An FPGA tuning framework for CNN acceleration, IEEE Des. Test 37 (1) (2020) 46–55, http://dx.doi.org/10.1109/MDAT.2019.2908549.

[105] J. Shen, D. Wang, Y. Huang, M. Wen, C. Zhang, Scale-out acceleration for 3D CNN-based lung nodule segmentation on a multi-FPGA system, in: 2019 56th ACM/IEEE Design Automation Conference, DAC, 2019, pp. 1–6.

[106] K. He, B. Liu, Y. Zhang, A. Ling, D. Gu, FeCaffe: FPGA-enabled caffe with opencl for deep learning training and inference on intel stratix 10, in: Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 314, http://dx.doi.org/10.1145/3373087.3375389, https://doi.org/10.1145/3373087.3375389.

[107] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, C.J. Rossbach, Sharing, protection, and compatibility for reconfigurable fabric with amorphos, OSDI'18, USENIX Association, USA, 2018, pp. 107–127.

[108] H.-Y. Ting, T. Giyahchi, A.A. Sani, E. Bozorgzadeh, Dynamic sharing in multi-accelerators of neural networks on an FPGA edge device, in: 2020 IEEE 31st International Conference on Application-Specific Systems, Architectures and Processors, ASAP, 2020, pp. 197–204, http://dx.doi.org/10.1109/ASAP49362.2020.00040.

[109] M. Blott, T.B. Preusser, N.J. Fraser, G. Gambardella, K. O'brien, Y. Umuroglu, M. Leeser, K. Vissers, FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks, ACM Trans. Reconfigurable Technol. Syst. (TRETS) 11 (3) (2018) 1–23.

[110] J. Shan, et al., Power-optimal mapping of CNN applications to cloud-based multi-FPGA platforms, IEEE Trans. Circuits Syst. (2020).

[111] J. Barnes, Azure machine learning, in: Microsoft Azure Essentials, Microsoft, 2015.

[112] P.R. Genssler, O. Knodel, R.G. Spallek, Securing virtualized FPGAs for an untrusted cloud, in: Proceedings of the International Conference on Embedded Systems, Cyber-Physical Systems, and Applications, ESCS, The Steering Committee of The World Congress in Computer Science, Computer ..., 2018, pp. 3–9.

[113] S. Yazdanshenas, V. Betz, The costs of confidentiality in virtualized FPGAs, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 27 (10) (2019) 2272–2283, http://dx.doi.org/10.1109/TVLSI.2019.2919644.

[114] F. Yao, A.S. Rakin, D. Fan, DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips, 2020, arXiv arXiv:abs/2003.13746.

[115] A. Shafee, T.A. Awaad, Privacy attacks against deep learning models and their countermeasures, J. Syst. Archit. 114 (2021) 101940, http://dx.doi.org/10.1016/j.sysarc.2020.101940, URL https://www.sciencedirect.com/science/article/pii/S138376212030196X.

[116] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, T. Dumitraș, Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks, in: Proceedings of the 28th USENIX Conference on Security Symposium, SEC'19, USENIX Association, USA, 2019, pp. 497–514.

[117] Xilinx, Xilinx Vitis AI, https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.

[118] A. Iordache, G. Pierre, P. Sanders, J.G. de F. Coutinho, M. Stillwell, High performance in the cloud with FPGA groups, in: Proceedings of the 9th International Conference on Utility and Cloud Computing, UCC '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1–10, http://dx.doi.org/10.1145/2996890.2996895, https://doi.org/10.1145/2996890.2996895.

[119] K. Matas, T. La, N. Grunchevski, K. Pham, D. Koch, Invited tutorial: FPGA hardware security for datacenters and beyond, in: Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 11–20, http://dx.doi.org/10.1145/3373087.3375390, https://doi.org/10.1145/3373087.3375390.

Chen Wu is a Ph.D. student at Hubert Curien Laboratory, Lyon University, France. She obtained her M.Sc. degrees at École des Mines de Saint-Etienne, France. Her research interests include Network on Chip on FPGA, Cloud computing, image processing applications.

Virginie Fresse received her Ph.D. in INSA Rennes in 2001 in Electrical Engineering. After a 2 years — post doc position in the University of Strathclyde in Scotland, she became an associate professor in the Jean Monnet University in Saint Etienne, France. Her research interests are on the design and implementation of real time image processing applications on FPGA platform, Design Space exploration of hardware architecture and NoC structure and CNNs on embedded systems integrating GPU or FPGA devices. She has also projects on deploying IA architectures on Cloud or Edge computing infrastructures.

Benoit Suffran is a Technical Staff Member of STMicroelectronics in FPGA Prototyping. He received M.S. degree in electronics from ISTASE in 2003. He joined STMicroelectronics in 2007. After being the technical leader to FPGA prototyping for various generations of SoC on mobile platform or Setup Box, he is currently responsible for prototyping activities of RF devices and EEPROM.

Hubert Konik received the Ph.D. degree in computer science from Universite Jean Monnet, in 1995. He is currently an Associate Professor with Telecom Saint-etienne and a member of Image Science and Computer Vision team, Laboratory Hubert Curien, Saint-etienne, France. His research interests are focused on image processing and analysis, more particularly content aware image processing for new services and usages.