





# Out-of-the-Box Performance of FPGAs for ML Workloads Using Vitis AI

Deepak Kumar Athur<sup>1</sup>, Rutuparn Pawar<sup>2</sup>, and Aman Arora<sup>1</sup>

<sup>1</sup> Arizona State University, Tempe, AZ 85281, USA  
{dathur, aman.kbm}@asu.edu

<sup>2</sup> The University of Texas at Austin, Austin, TX 78712, USA  
rutuparn.pawar@utexas.edu

**Abstract.** Field Programmable Gate Arrays (FPGAs) are an attractive choice for accelerating Machine Learning (ML) workloads due to the flexible fabric of configurable logic blocks, interconnects, and embedded memory. However, programming FPGAs is difficult for ML developers as it requires intricate hardware knowledge. Even though high-level implementation solutions such as HLS are available, they come with their own challenges and a steep learning curve. To address this issue, FPGA vendors have raised the level of abstraction by providing ready-to-deploy frameworks for ML. In this paper, we present an evaluation of the out-of-the-box performance of FPGAs using AMD/Xilinx Vitis AI, a development environment for deploying ML models on FPGAs. The study aims to assess the inference performance of Vitis AI for both edge and cloud platforms. We benchmark various popular and standard pre-trained models focusing on latency, throughput, and power efficiency. Since Google Tensor Processing Units (TPUs) are a platform for out-of-the-box acceleration of ML, we compare these results with cloud TPU and edge TPU in terms of performance, ease of use, and tool support. We discuss the experience of working with Vitis AI, the strengths and limitations of Vitis AI as a plug-and-play solution for FPGA-based ML acceleration, providing insights for developers looking to leverage FPGAs for their inference workloads.

AQ1

AQ2

## 1 Introduction

**Platforms for Running ML Workloads:** Deep Neural Networks (DNNs) inference has become a fundamental workload in modern applications. The high computational complexity of DNN algorithms and the stringent cost-energy-latency constraints for data center workloads necessitate efficient domain-specific hardware. With several hardware platform choices available from Graphic Processing Units (GPUs) to reconfigurable Field Programmable Gate Arrays (FPGAs) to highly specialized ASICs (Application Specific Integrated Circuits) it is important to make a meaningful hardware platform choice. GPUs are power-hungry and are optimized for throughput, hence they are less suitable for inference at the edge [8, 12]. Google TPUs, on the other hand, are ASICs that must be

Author Proof

redesigned when models evolve, costing a lot of time and money. However, these platforms have achieved success because they are easy to program. The vendors of TPUs and GPUs have spent considerable resources developing software stacks to easily use their hardware for DNN acceleration. Nvidia, for example, has created libraries like cuDNN and cuBLAS, which are natively supported by popular ML frameworks like Pytorch and Tensorflow, making it easy to run ML workloads optimally without any GPU programming. For the TPUs, Google provides Accelerated Linear Algebra (XLA), a domain-specific compiler that takes operations written Pytorch or Tensorflow and optimizes them for execution on the TPU. Also, the pre-configured cloud-based environments make it easy to work with TPUs.

**FPGAs, Flexible and Efficient:** The field of AI is rapidly evolving, with new algorithms and models being developed every week. Hardware capable of effectively running these algorithms must likewise advance at the same rate to keep up with the fast pace of algorithm development. This has led to the widespread use of reconfigurable hardware architectures such as Field Programmable Gate Arrays (FPGAs) that can easily adapt to changing requirements. Moreover, AI’s computational complexity also comes with an energy cost which has become an important factor for both cloud and edge platforms. FPGA solutions are known to be more energy-efficient [7, 8], by designing customized dataflows specific to the model being deployed and exploiting the inherent parallelism in DNNs. Furthermore, they avoid the overhead of fetching/decoding instructions and can use smaller data precisions.

**FPGAs have a Steep Learning Curve:** While FPGAs offer appealing advantages, most ML engineers are not accustomed to designing at lower abstraction levels like RTL, which involves hardware description languages like Verilog or VHDL and requires an understanding of hardware-specific concepts. High-Level Synthesis (HLS) addresses this concern by allowing engineers to describe hardware using languages like C/C++. However, it still requires some hardware knowledge and often yields less optimal results compared to expert-written RTL [14].

**Out-of-the-Box DNN Deployment Frameworks for FPGAs:** Overlays are intermediate hardware designs that abstract the complexity of low-level FPGA fabric, providing developers with a simpler, programmable interfaces. This abstraction gap helps users to interact at higher levels. The Deep Learning Processing Unit (DPU) [4], Deep Learning Accelerator (DLA) [2] are such overlays offered by AMD and Intel respectively for easy DNN acceleration. Vendors also provide software tools and frameworks for these overlays. AMD’s Vitis AI [4], Intel’s FPGA AI suite [3], and Lattice AI Suite [21] and Matlab’s DNN toolbox [18] are such tools and frameworks that help users optimize and accelerate workloads on FPGAs. The goal is to make FPGAs easy to use, allowing an end-user, such as an ML practitioner, to utilize them with minimal effort-essentially achieving an “out-of-the-box” experience.

**Contributions:** Our intent in this work is to understand how easy it is to use modern FPGAs for DL acceleration. We wish to use this as a case study to see the maturity of FPGA tool flows for realistic deployment scenarios that do not involve experienced computer engineers who can write RTL or use HLS. We also compare the out-of-the-box performance and experiences with TPUs. To that end, we use AMD FPGAs and Vitis AI suite in this work. There is a knowledge gap about Vitis AI’s capabilities and limitations, which our work tries to fill. Also, our analysis is helpful for companies to improve their solutions thereby expanding the user base of FPGAs. Specifically, our contributions are:

- Performance analysis of ML workloads on FPGAs deployed using AMD/Xilinx Vitis AI, for both edge and cloud devices.
- Comparison with Google TPUs (Edge and Cloud platforms) for out-of-the-box performance, deployment flow, and ease of use.
- Highlighting our experiences with AMD/Xilinx Vitis AI - strengths, weaknesses, and issues faced.

While some FPGA domain experts may know the experience and limitations of tools like Vitis AI, we believe it is useful for people from other domains (i.e., ML engineers) to understand the capabilities that exist with FPGA tools to make informed decisions.

## 2 Related Work

There are a few recent works that have focused on Vitis AI. In [24], the authors implement three different variants of ResNet that vary in number of layers, which are implemented on Zynq-7000 SoC. They compare the throughput and power consumption with GPU. In [25], the authors implement Yolov3 on ZCU104 and compare results with GPU. In [17], a course-grained accelerator using SystemVerilog, a fine-grained FINN accelerator, and a sequential Vitis AI accelerator are implemented and their performance is compared using the Zynq ZCU3EG device. In [11], the performance of FINN, Vitis AI and Jetson Nano GPU are compared for a single Resnet Model. In [6], the performance of CNNs implemented on U280 using HLS4ML and Vitis AI is studied.

FPGA overlays have grown in popularity for DNN acceleration [5]. Microsoft Brainwave provides low latency inference for neural networks in the cloud [7]. Recent works have evaluated FPGAs and GPUs for DNNs [19].

Although academic frameworks like HLS4ML [1], DNNWeaver [22], FPDNN [9], HPIPE [10], etc. exist, they have several drawbacks. These lack support for some frameworks or only limited support, inadequate profiling tools, support for multiple devices, thorough documentation, and technical support, which renders them less attractive than FPGA vendor tool options. As ML is a rapidly developing field, academic solutions might be unable to adapt and optimize architectures for these advances quickly. FPGA vendors have more extensive resources and financial incentives to overcome these challenges. Intel and Lattice have their Intel FPGA AI Suite and Lattice AI suite for out-of-the-box acceleration [3, 21].

**Table 1.** Resource utilization and throughput for the DPUs used for our edge and cloud FPGAs

Platform	LUTs	FFs	BRAM	DSP	FPGA Peak TOPS
Edge	52161	98249	255	710	3.36
Cloud	592600	1013608	616	6569	26

In our work, we use a range of standard models that are commonly used and evaluate AMD’s Vitis AI tool for implementing them on both Edge and Cloud platforms on devices marketed for ML acceleration. We also compare it with the deployment flow for Google TPUs in terms of features, performance, and ease of use.

### 3 Out-of-the-Box Deployment of DNNs on FPGAs

**Vitis AI Overview:** AMD Xilinx’s Vitis AI is an open-source, comprehensive framework for the end-to-end deployment of ML inference on AMD hardware suitable for both cloud and edge applications [4]. This tool has a wide range of hardware support including Alveo data center cards, Zynq Ultrascale+ MPSoCs, Xilinx Versal Adaptive Compute Acceleration Platform (ACAP) cards, and Kria System on Modules. This tool is developed keeping the ML algorithm developers in mind who would like to develop and deploy ML applications to speed up inference deployment by abstracting the underlying hardware’s intricacies. Vitis AI consists of set of IP cores, tools, libraries, models and example designs.

**DPU FPGA Overlay:** The DPU is an optimized overlay for DNN execution on AMD/Xilinx FPGAs. Vitis AI uses this as the core accelerator for inference. The DPU fetches and executes an optimized DPU instruction set for model inference from system memory. This helps abstract away the complex underlying hardware.

The architecture of the DPU [4] is shown in Fig. 1. The DPU is on the Programmable Logic (PL) and communicates with the host. The DPU can consists of several DPU cores which again contain several PEs which are convolution Engines. To handle interrupts and manage data transfers, an application on the host is also necessary. The instructions are fetched from off-chip memory, decoded and dispatched by the instruction scheduler to control the DPU. To attain high throughput and efficiency, input activations, intermediate feature maps, and output meta-data are buffered using on-chip memory.

There are several variants of DPUs, each targeting a particular hardware. Some features of the DPUs are parameterized to enable optimizing the design based on the user’s requirements. Features such as number of cores, PEs, RAM usage etc. can be configured. However, the parameters that can be configured depend on the DPU. The two DPU architectures we use in our work for edge and cloud platforms are DPUCZDX8G and DPUCAHX8H-DWC respectively.

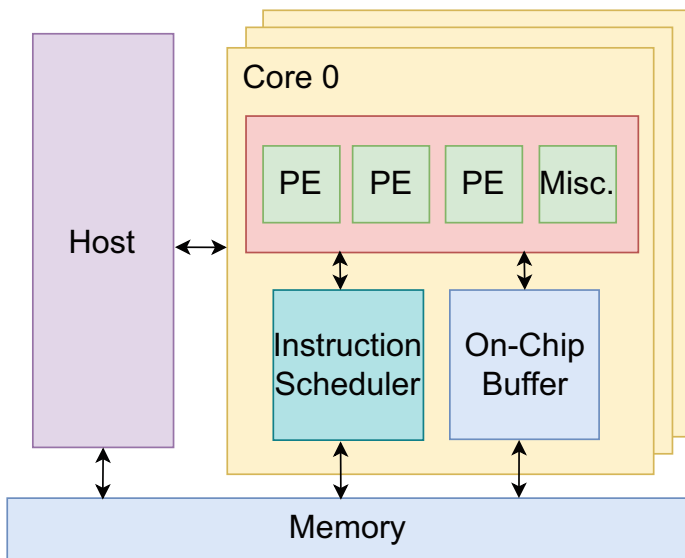


Fig. 1. High-level architecture of the DPU overlay

DPUCZDX8G is the DPU designed for Zynq Ultrascale+ MPSoC FPGAs. It comes in different configurations with varying throughput and resource usage. For our **edge FPGA platform**, we use the B4096 variant which has the highest peak throughput and resource utilization which are described in Table 1. The DPU has one core with four Processing Elements (PEs) for fundamental computation and runs at a clock frequency of 325 MHz.

DPUCAHX8H is the DPU designed for Alveo cards with HBM memory. It comes in throughput-optimized and latency-optimized variants. For our **cloud FPGA platform**, we use the latency-optimized variant DPUCAHX8H-DWC. It consists of 3 cores with 11 PEs in total and runs at a clock frequency of 300 Mhz with a peak performance of 14745 GOPs/s. To attain high throughput and efficiency, HBM is utilized to buffer weights, biases etc. The resource utilization of this DPU is mentioned in Table 1.

**Vitis AI Flow:** The overall flow that we follow for deploying DNNs on FPGAs is summarized in Fig. 2. To deploy a model on the DPU, it first needs to be quantized using Vitis AI quantizer. This tool quantizes the model’s floating point precision weights and activations to fixed point (8-bit) with minimal loss in prediction accuracy. This can help reduce the model’s memory footprint by reducing the overall size and the bandwidth for loading and unloading the data. This is particularly crucial for the deployment of these demanding models on resource-constrained edge devices. Then the binary instruction code that can be deployed on the DPU is generated using Vitis AI Compiler. Vitis AI Run Time (VART) provides APIs which are used to manage and execute AI models on the DPU. The instructions are then loaded into the DRAM.

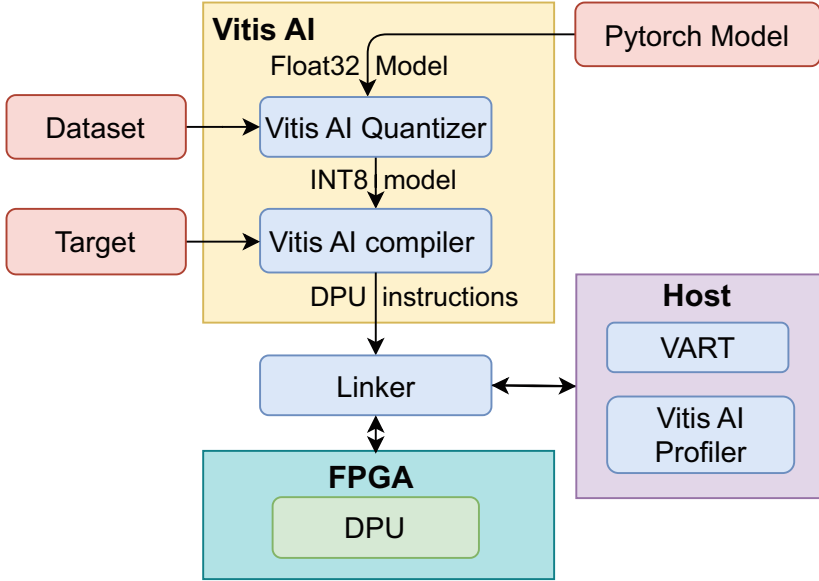


Fig. 2. Summary of DNN acceleration flow using Vitis AI

**Model Zoo:** Xilinx also provides a Model Zoo that contains several models several pre-trained models for different applications that are ready to be deployed on the DPU. Various versions of the same model for various hardware platforms are available. This out-of-the-box support reduces development time further. Examples of some models that are available are UNet, Inception, Yolo, and Mobilenet. Transformer-based models are not supported as of this writing.

## 4 Methodology

In this section, we discuss the details for our approach to evaluate Vitis AI for ML acceleration on FPGAs. This involves empirical analysis to evaluate performance.

### 4.1 FPGA Platforms

We sampled an edge and a cloud FPGA from the extensive number of FPGA platforms available, thereby covering both common scenarios. Additionally, these FPGAs come close in terms of specs to the Google TPUs (mentioned in the next section). Furthermore, these FPGAs are marketed for ML applications. While the experiments depicted in this paper can be performed on more FPGAs, our conclusions regarding the performance gap between out-of-the-box solutions and hand-coded solutions and showcasing the experiences of using out-of-the-box solutions, will stay the same. For the edge scenario, we use **KRIA KV260**. It

**Table 2.** Specifications of edge and cloud FPGA

	KV260	U55C
INT8 TOPS/s	3.36	26
DDR Bandwidth (GB/s)	9.2	460
Max power (W)	15	150
Tech node	16 nm	16 nm
On-chip memory (MB)	2.88	23
Availability	2021	2021

**Table 3.** Specifications of edge and cloud TPU

	TPU v3	Edge TPU
TOPS/s	123 (BF16)	4 (INT8)
DDR Bandwidth	900 GB/s	-
Max power	262	-
Tech node	16 nm	-
On-chip memory	32 MB	8 MB
Availability	2018	2019
Clock frequency	940 Mhz	500 Mhz

targets ML application engineers and developers who have strong time to market constraints allowing them to quickly prototype and deploy AI-driven systems. For the cloud scenario, we use **U55C**, which is a high-performance accelerator card for workloads like machine learning, fintech, and data search. It is equipped with 16GB of high bandwidth memory (HBM) and fast networking capabilities. The specifications for these FPGAs are described in Table 2. **INT8** precision is used for executing the DNNs on both FPGA platforms.

## 4.2 TPU Platforms

We use the TPU for comparison because we want to compare out-of-the-box performance and ease of deployment. The **Edge TPU** is an ASIC developed by Google for the edge. It is designed to execute TensorFlow Lite models efficiently, providing high-speed ML inference while maintaining low power consumption. The Edge TPU only supports **INT8** precision. On the other hand, for the cloud scenario, we use **TPU v3**, which is a large multi-die accelerator card for training, fine-tuning and inference. We refer to this TPU as ‘Cloud TPU’ in this paper. This TPU is closer to the U55C than TPU v2 in terms of max power. One TPU v3 has 4 chips, and each chip has 2 cores which equals to 8 cores per card. For our experiments, we are only using 1 core of the TPU. This version of the TPU does not support INT8 precision. Hence, we use Google’s Brain Float 16 (**BF16**) precision for our experiments. Specifications of the TPUs are mentioned

in Table 3. Some specifications are not available for the Edge TPU. Furthermore, the cloud TPU and Edge TPU do not have an inbuilt method to monitor power.

### 4.3 Benchmarks

The benchmarks chosen for evaluating the performance of our FPGA and TPU platforms are shown in Table 4 in descending order of the total number of compute operations. These popular DNNs have diverse computational demands that provide varied insights. *Note that these benchmarks are all Convolutional Neural Networks (CNNs). Unfortunately, Transformer-based models that form the backbone of recent DNN applications like ChatGPT are not supported by the Vitis AI flow, for FPGA devices (they are supported only for devices with AI engines). Hence, we are not able to deploy them.* Furthermore, YoloX-Nano could not be deployed on our cloud FPGA due to lack of support. All evaluations are performed at a batch size of 1 for edge devices and a batch size of 3 for cloud devices. A batch size of 1 for edge inference is a common usecase, but a higher batch size would have been more practical for cloud inference. However, the out-of-the-box DPU that we use does not support other batch sizes than a batch size of 1 for the KV260 and a batch size of 3 for U55C. Hence, our experiments were limited to these batch sizes.

**Table 4.** Benchmarks used in our experiments

Model	Operations	Parameter size		Input Sizes
	GOPs	M	MB	
VGG19	39.28	143.667	548.09	224 * 224 * 3
VGG16	30.96	138.358	527.79	224 * 224 * 3
Resnet-152	21.83	60.193	229.62	224 * 224 * 3
Resnet-50	8.2	25.557	97.49	224 * 224 * 3
MobileNetV1	1.14	3.505	13.37	224 * 224 * 3
YOLOX-Nano	1	0.91	2	416 * 416 * 3
EfficientNet b0	0.78	5.289	20.17	224 * 224 * 3

### 4.4 Software Platforms

For the edge TPU, we take the Pytorch model and convert it to Tensorflow Lite. Then, this model is run on the Edge TPU using the TPU compiler. For the Cloud TPU, we deploy the Pytorch model using XLA compiler. When it comes to both the Cloud and Edge FPGA, we go through the Vitis AI flow where the DPU instructions are generated by Vitis AI compiler executed on the DPUs. We use Vitis AI version 3.0 for KV260, but we are forced to use Vitis AI version 2.5 for U55C as the later versions have dropped support for that device.

## 4.5 Performance Measurement

The key metrics that we focus for evaluating the platforms are latency, throughput and energy efficiency. We measure latency using the TPU profiler and APIs for the Cloud TPU and the Edge TPU respectively, and Vitis AI Profiler and APIs the Edge FPGA and the Cloud FPGA respectively. We measure the latency of the DPU or TPU execution only. We ignore any overhead from the host. We measure utilization as a percentage ratio of peak TOPS and achieved TOPS for the TPU, and peak TOPS of the FPGA (not the DPU peak TOPS mentioned in Table 2) and achieved TOPS for the FPGA. We measure the power of the System on Module (ARM core, FPGA, etc.) for the edge FPGA, and for the Cloud FPGA, we measure the power of the FPGA board (PCIE card). For the edge FPGA we use Xilinx’s xutil and for Cloud FPGA we use xbutil to measure power. Hence we obtained Inferences/J as shown below.

$$\text{Inferences per Joule} = \frac{\text{Inferences per second}}{\text{Power in Watts}} \quad (1)$$

Since XLA compiler works in ‘lazy’ mode by default (i.e. it builds the whole graph, optimizes it and then executes it), we use XLA APIs to break the graph and force the compiler to execute ‘greedily’ for performance debug.

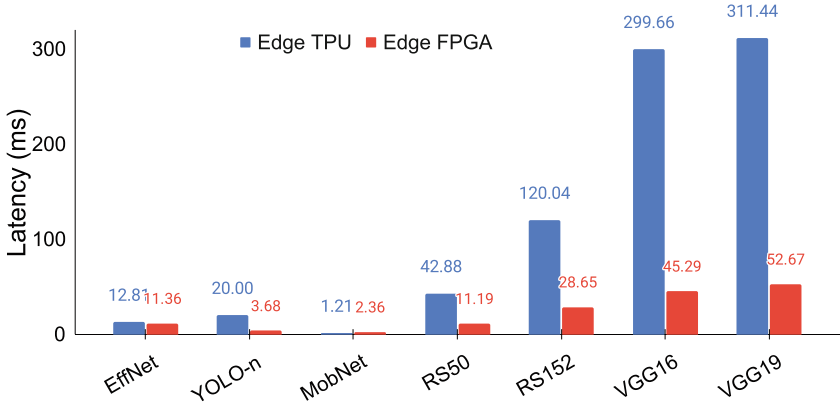


Fig. 3. Latency of edge devices for various DNNs

## 5 Results and Discussion

In this section, we present the performance of the various benchmarks in terms of latency, utilization, and throughput efficiency. We also present a roofline analysis of these benchmarks. *Specifically, we compare the performance achieved on FPGA platforms using Vitis AI with the performance achieved on TPU*

platforms, as a way to demonstrate the out-of-the-box performance of DNNs on FPGAs. The underlying assumption here is that similar efforts are spent on deploying the ML models on both the FPGA and the TPU. No extensive customizations are performed on the FPGA platforms, other than those easily achieved through the Vitis AI framework. Since this effort is difficult to quantify, we qualitatively compare our experiences deploying the DNNs on FPGAs and TPUs in the last subsection (Sect. 5.6). Note that our goal is not to compare the achievable performance of FPGAs with that of ASICs, which is done in prior work [13].

### 5.1 Edge Platforms: Latency and Utilization

Starting with the edge devices Kria KV260 and Edge TPU, the inference latency of benchmarks are shown in Fig. 3. The single-batch inference of **edge FPGA outperforms the edge TPU** by 5.2X on average. The largest gap is seen for VGG19. Note that we calculate utilization using peak performance of the FPGA not the DPU. From Fig. 4, we also see a trend that the Edge TPU is underutilized for most benchmarks. However, the Edge TPU seems to be optimized for MobileNet, the only benchmark where it beats Edge FPGA. Even though the peak throughput of the edge TPU is higher than that of the edge FPGA, the edge TPU being optimized for power and the lack of sufficient on-chip memory for larger models are the likely reasons for this unexpected result. Also, EfficientNet has extremely poor utilization on both the devices.

### 5.2 Cloud Platforms: Latency and Utilization

For the cloud devices TPUv3 and U55c, from the batch 3 inference metrics shown in Fig. 5, **the TPUv3 outperforms cloud FPGA** by an average of 1.37X. We observe that the gap is small for VGG166 and is the largest for MobileNetV1. Comparing their power consumption would have given a better

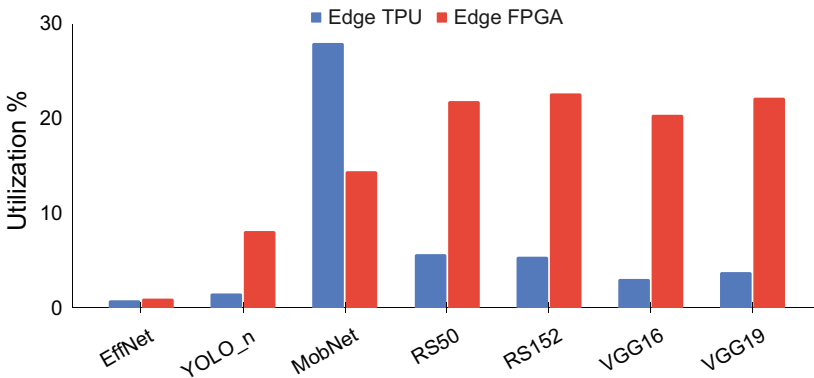


Fig. 4. Utilization of edge devices for various DNNs

insight. However, there is no means of measuring cloud TPU's power accurately. Regarding utilization (Fig. 6), for a batch size of 3, both the devices do not cross 35%. Based on Fig. 5, the DPU architecture appears to be more rigid than the TPU. We must note that this DPU supports only this batch size while the TPU supports several. When we run a Resnet50 inference with a higher batch size on the TPU, for example, a batch size of 512, the utilization of the TPU reaches 55%. An interesting point to note is that while the edge FPGA is worse in terms of latency on average (Fig. 3) compared to the cloud FPGA (Fig. 5), its performance for edge-optimized models like MobileNet and EfficientNet is very close to that of the cloud FPGA. **The cloud FPGA has much more room for optimizations.**

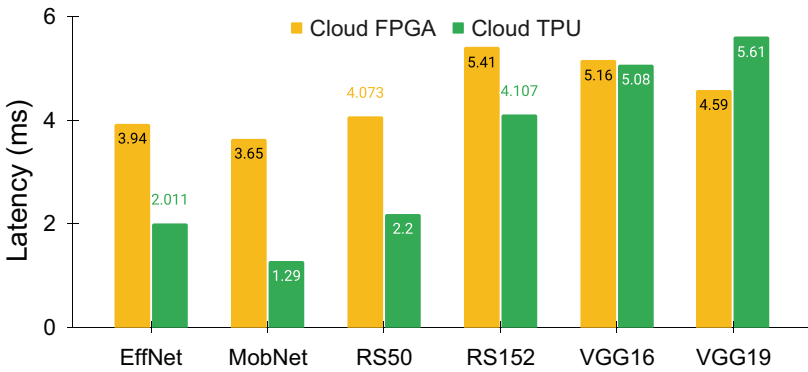


Fig. 5. Latency of cloud devices for various DNNs

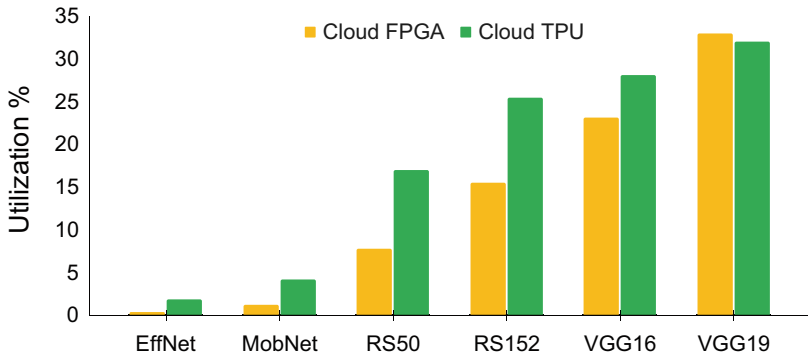


Fig. 6. Utilization of cloud devices for various DNNs

### 5.3 Energy Efficiency

Figure 7 shows the energy efficiency in terms of Inferences/Joule for the various benchmarks on both edge and cloud FPGAs. On average, the Edge FPGA is 4.19X more efficient for these benchmarks. But this is misleading. Although the edge FPGA provides excellent efficiency for MobileNet (17.3X) and EfficientNet(1.98x), it quickly falls off, performing much worse for more compute-intensive workloads in comparison to the cloud FPGA.

### 5.4 Roofline Analysis

The performance results are shown in a roofline plot in Fig. 8 for the cloud platforms. All the models except VGG16 are bandwidth-limited on the TPU. For the FPGA, EfficientNet and MobileNet are bandwidth limited, whereas others are compute bound. We see a trend that models that are optimized for edge (e.g., Efficient and MobileNet, as seen in Fig. 3) are the most memory bound in the cloud scenario. For the edge platforms, to plot the roofline chart, we could not find the peak bandwidth because of lack of documentation of these platforms.

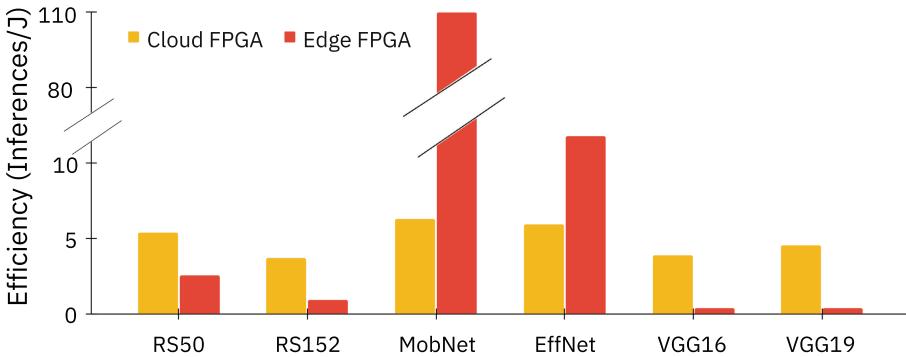
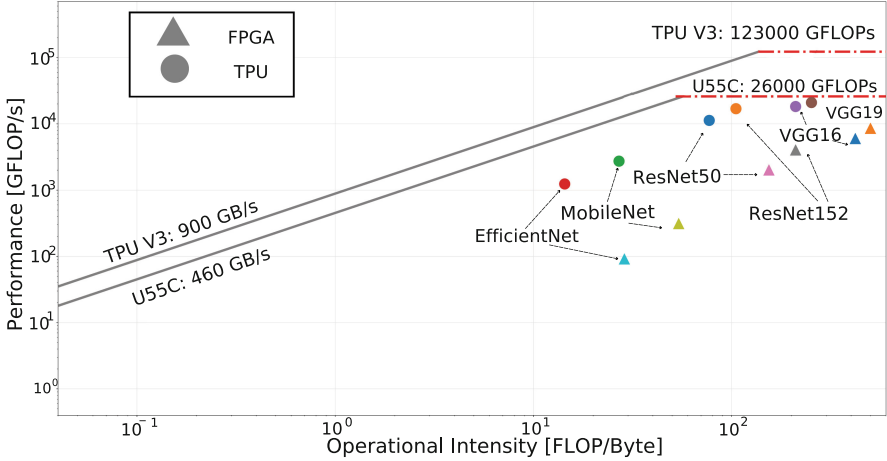


Fig. 7. Energy Efficiency for cloud and edge FPGAs

### 5.5 Comparison with Non Out-of-the-Box Performance

To understand how well the out-of-the-box solutions work, we compare their performance with that achieved by prior works which use hand-coded RTL-based accelerators, for the same DNNs. However, it is difficult to obtain the raw performance results for same DNNs, same FPGA devices, same precision, and same technology node from prior non-Vitis AI works. Hence, for fair comparison, we use the efficiency metric (GOPS/W) to compare our results with results from any FPGA device, for the same DNNs. For differences in technology node, we scale the results using [23] to a common value (16nm). The efficiency of non-Vitis AI works implemented using hand-coded accelerators is listed in Table 5.



**Fig. 8.** Roofline plot for cloud TPU and cloud FPGA

In comparison to Vitis AI, these works achieve  $2.65\times$ ,  $3.52\times$ ,  $1.89\times$  higher efficiency compared to our edge platform and  $1.6\times$ ,  $6.4\times$ ,  $16.85\times$  compared to our cloud platform, for Vgg16, Resnet50 and MobileNetv1 respectively. We conclude that, at present, the **out-of-the-box solution provides inferior efficiency compared to hand-coded solutions**. Note that the precision of these works is larger than or equal to INT8 (the precision supported by Vitis AI). These prior works obtain a higher efficiency even with larger precision, implying that the efficiency will be even higher with a lower INT8 precision. Thus, our conclusion about the inferior efficiency of out-of-the-box solutions will not change. Since only INT8 precision is supported in Vitis AI, these results also highlight that hand-coded solutions offer higher flexibility.

**Table 5.** Comparison with non Vitis AI solutions

Work	Device	Model	Precision	GOPS/W
[16]	Virtex7 VX690T	VGG16	8-bit BFP	195
[20]	ZC706	Resnet50	8–16 bit	283.5
[15]	XCVU37P	MobileNetv1	8-bit	121.30

## 5.6 Experiences with Vitis AI

Throughout our experience with Vitis AI, we identified various strengths and challenges associated with the deployment of machine learning models utilizing this platform. Overall, Vitis AI is a great effort towards making FPGA acceleration available out-of-box. The experience was a mix. The flow shines in many

places, but there is significant scope for improvement. Based on our experience, to develop an efficient ML accelerator on an FPGA using hand-coded Verilog, it would take 5–6 months of design and debug, assuming we heavily use existing IP blocks for common parts of the design. But with Vitis AI, this can be reduced to 3–4 weeks of time that includes setup, customization, deployment and debug. However, our experience was worse than this because of the issues we mention below in this section. On the other hand, for a TPU, it takes about 3–4 days to set up and run DNNs.

(1) Vitis AI repository contains a **large number of examples for different scenarios to help developers** to learn the flow quickly. However, the Vitis AI tool has a large number of capabilities and even with this large number of examples, developing some projects might be challenging. The Vitis AI model zoo consists of a diverse set of ready-to-deploy models. Several variants with various pruning ratios are present for the user to choose from, and new models are added with each release. However, **some models are available only to a particular DPU or device for a specific version of Vitis AI.**

(2) Vitis AI optimizer is a valuable tool for developers to optimize the performance of the deployed models. It provides several pruning features and techniques like Neural Architecture Search (NAS), Once-for-All (OFA), iterative and one-step pruning. The optimizer’s paid license requirement has only been removed for version 3.5 at the time of this work. Also, **not all these features of the optimizer are supported by all the ML frameworks** (PyTorch, TensorFlow, etc.).

(3) Vitis Model Inspector is a handy tool that provides insights into compiled models. It provides information on which layers on CPU and which ones run on the DPU. It can present the compiled models visually similar to Netron. This can **help developers optimize and confirm DNN compatibility with a specific DPU.**

(4) Since Vitis AI runs on a docker container for cloud FPGAs, it can cause issues. We have tried running the Vitis AI flow on academic compute clusters and on Xilinx’s Heterogeneous Accelerated Compute Cluster (HACC) that have Xilinx FPGAs. However, these **public clusters do not provide root access which makes it very hard or impossible to run Vitis AI flow.** The documentation also mentions that users can utilize the FPGAs in Azure or AWS. However, the documentation for this has not been updated.

(5) All the tools involved in Vitis AI flow have **strict version dependencies.** This, combined with the availability of several versions for Vitis AI and all its tools, along with documentation gaps with each release, makes it **quite confusing to set up the environment.** For example, each Vitis AI version only supports specific versions of XRT (Xilinx Run Time), DPU IP, shells (necessary infrastructure on the FPGA), OS, OS kernel, Vivado, and Vitis. It will be a great help to the developers if a dashboard could be developed, which documents models, supported FPGAs, versions of all supported tools, DPU capabilities, and download links.

(6) With each version of Vitis AI release, AMD seems to drop support for some devices. **Transformer-based models are only supported on devices with AI engines.** Recurrent Neural Networks (RNN) are not supported by Vitis AI flow but by a separate flow using the Vitis AI RNN compiler. Besides Vitis AI version 3.5, older versions **do not support precisions other than INT8.**

(7) Users can integrate custom logic along with the DPU, such as custom pre-processing blocks for specific image transformations (e.g., resizing or normalization) or custom post-processing blocks to handle the output of the DPU for additional operations like encoding. Even though Xilinx provides some parameters to customize the edge DPU, **users need to rebuild the operating system with the required firmware. They cannot just load a new bitstream and run their applications with that new DPU.** When the DPU configuration is customized, the underlying hardware design is changed. This modified hardware design requires corresponding firmware components, such as device tree files, kernel modules, and drivers, which describe the hardware configuration to the Linux operating system.

(8) **The cloud TPU environment is much easier to use than the FPGA overall.** Since cloud TPU is available in Google Cloud Platform (GCP), users can get started with just a few commands making it highly convenient. There is no hassle of setting up the card and environment as it is for the FPGA. Google TPUs in GCP come pre-configured, eliminating the time and technical expertise required to set up such environments. Porting your workloads to the TPU can be as easy as writing one line of code. Similar to Vitis AI's Profiler, the TPU profiler can help users identify bottlenecks and improve performance. **The Vitis AI DPU is not as flexible as the TPU.** For example, transformer-based models can be easily run on the TPU, and, in terms of batch size, the user can choose any arbitrary batch size. Though the TPUs support several data precisions, some versions, however, do not support INT8 precision.

## 6 Conclusion

In this paper, we evaluated the experience of out-of-the-box deployment of DNN workloads on FPGAs, using AMD/Xilinx Vitis AI suite. We focused on performance as well as ease-of-use, and compared them with TPUs. We believe that while such deployment can provide high performance, even surpassing TPUs with higher theoretical throughput in some cases, the experience leaves a lot to be desired in terms of ease-of-use. Particularly, we believe that the following efforts are essential for widespread usage of such platforms: (1) making the environment setup easier, (2) ensuring all the parts of the flow (models, tool versions, overlay versions, devices) are well supported and documented, (3) supporting state-of-the-art DNNs (such as transformer-based models) and (4) supporting all batch sizes and precisions for a wide range of devices.

## References

1. Aarrestad, T., et al.: Fast convolutional neural networks on FPGAs with hls4ml. *Mach. Learn. Sci. Tech.* **2**(4), 045015 (2021). <https://doi.org/10.1088/2632-2153/ac0ea1>
2. Abdelfattah, M.S., et al.: DLA: compiler and FPGA overlay for neural network inference acceleration. In: 2018 28th International Conference on Field Programmable Logic and Applications (FPL), pp. 411–4117 (2018). <https://doi.org/10.1109/FPL.2018.00077>
3. Ahmad, J., Jervis, M., Venkata, R.: Intel® FPGAs and SOCs with intel® FPGA AI suite and OpenVino toolkit drive embedded/edge AI/machine learning applications
4. AMD: AMD Vitis AI (2020). <https://www.amd.com/en/developer/resources/vitis-ai.html>
5. Boutros, A., Nurvitadhi, E., Betz, V.: Specializing for efficiency: customizing AI inference processors on FPGAs. In: 2021 International Conference on Microelectronics (ICM), pp. 62–65. IEEE, New Cairo City, Egypt (2021). <https://doi.org/10.1109/ICM52667.2021.9664938>
6. Carini, D.: Comparing hls4ml and VITIS AI for CNN synthesis and evaluation on FPGA: a comprehensive study (2022)
7. Chung, E., et al.: Serving DNNs in real time at datacenter scale with project brainwave. *IEEE Micro* **38**(2), 8–20 (2018). <https://doi.org/10.1109/MM.2018.022071131>
8. Dhilleswararao, P., Boppu, S., Manikandan, M.S., Cenkeramaddi, L.R.: Efficient hardware architectures for accelerating deep neural networks: survey. *IEEE Access* **10**, 131788–131828 (2022). <https://doi.org/10.1109/ACCESS.2022.3229767>
9. Guan, Y., et al.: 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), pp. 152–159. IEEE (2017)
10. Hall, M., Betz, V.: HPIPE: heterogeneous layer-pipelined and sparse-aware CNN inference for FPGAs. arXiv preprint [arXiv:2007.10451](https://arxiv.org/abs/2007.10451) (2020)
11. Hamanaka, F., Odan, T., Kise, K., Van Chu, T.: An exploration of state-of-the-art automation frameworks for FPGA-based DNN acceleration. *IEEE Access* **11**, 5701–5713 (2023)
12. Jouppi, N.P., et al.: In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th Annual International Symposium on Computer Architecture, pp. 1–12 (2017)
13. Kuon, I., Rose, J.: Measuring the gap between FPGAs and ASICs. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **26**(2), 203–215 (2007). <https://doi.org/10.1109/TCAD.2006.884574>
14. Lahti, S., Sjövall, P., Vanne, J., Hämäläinen, T.D.: Are we there yet? A study on the state of high-level synthesis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **38**(5), 898–911 (2019). <https://doi.org/10.1109/TCAD.2018.2834439>
15. Li, Z., et al.: A high-performance pixel-level fully pipelined hardware accelerator for neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 1–14 (2024). <https://doi.org/10.1109/TNNLS.2024.3423664>
16. Lian, X., Liu, Z., Song, Z., Dai, J., Zhou, W., Ji, X.: High-performance FPGA-based CNN accelerator with block-floating-point arithmetic. *IEEE Trans. Very Large Scale Integr. (VLSI) Systems* **27**(8), 1874–1885 (2019)

17. Machura, M., Danilowicz, M., Kryjak, T.: Embedded object detection with custom LittleNet, FINN and VITIS AI DCNN accelerators. *J. Low Power Electron. Appl.* **12**(2), 30 (2022)
18. Miller, R.: Meta Previews New Data Center Design for an AI-Powered Future. <https://www.datacenterfrontier.com/data-center-design/article/33005296/meta-previews-new-data-center-design-for-an-ai-powered-future/>. Accessed 12 Sept 2024
19. Nurvitadhi, E., et al.: Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In: *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 5–14 (2017)
20. Ou, Y., Yu, W.H., Un, K.F., Chan, C.H., Zhu, Y.: A 119.64 GOPs/W FPGA-based resnet50 mixed-precision accelerator using the dynamic DSP packing. *IEEE Trans. Circuits Syst. II: Express Briefs* **71**(5), 2554–2558 (2024). <https://doi.org/10.1109/TCSII.2024.3377356>
21. Semiconductors, L.: Accelerating implementation of low power artificial intelligence at the edge. In: *A Lattice Semiconductor White Paper* (2018)
22. Sharma, H., et al.: DNNweaver: from high-level deep network models to FPGA acceleration. In: *The Workshop on Cognitive Architectures* (2016)
23. Stillmaker, A., Baas, B.: Scaling equations for the accurate prediction of CMOS device performance from 180nm to 7nm. *Integration* **58**, 74–81 (2017). <https://doi.org/10.1016/j.vlsi.2017.02.002>, <https://www.sciencedirect.com/science/article/pii/S0167926017300755>
24. Ushiroyama, A., Watanabe, M., Watanabe, N., Nagoya, A.: Convolutional neural network implementations using VITIS AI. In: *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0365–0371 (2022). <https://doi.org/10.1109/CCWC54503.2022.9720794>
25. Wang, J., Gu, S.: FPGA implementation of object detection accelerator based on VITIS-AI. In: *2021 11th International Conference on Information Science and Technology (ICIST)*, pp. 571–577. IEEE (2021)