Characterizing CNN Throughput and Energy Under Multithreaded and Multiaccelerator Execution

M A Muneeb, *Member, IEEE*, and Rajesh Kedia, *Senior Member, IEEE*

***Abstract*—Emerging applications and batch processing convolutional neural network (CNN) workloads require executing multiple CNNs concurrently. A wide variety of CNN accelerators are available today and we characterize the support for concurrency for CNNs in such accelerators. We use a commercial-off-the-shelf CNN accelerator in multithreading and multiaccelerator modes and identify that upto 3.98× improvement in throughput and 3.20× improvement in energy per inference can be obtained even with just a single accelerator. Our detailed characterization of 104 CNN models, for three different sizes of accelerator, reveals many insights that connect CNN characteristics to improvement in throughput and energy. We also present a design space and a low error throughput estimation model to explore such a design space.**

***Index Terms*—Accelerator, convolutional neural network (CNN), embedded system, field programmable gate array (FPGA).**

I. Introduction

THE SUCCESS of convolutional neural networks (CNNs) in applications like object detection, classification, and segmentation has led to a wide variety of CNN accelerators [1], [2], [3]. Most CNN accelerators for embedded applications are designed for executing one CNN at a time. However, emerging applications like autonomous driving assistant systems [4] and batch processing workloads require executing multiple CNNs concurrently. Thus, it is important to evaluate CNN accelerators for concurrent execution which has not been considered in prior CNN characterization [5], [6].

Typically, a host CPU invokes the CNN execution on such accelerators and also executes any nonstandard CNN layer not supported by the accelerator. Concurrent execution can be realized in two ways: 1) when only one accelerator is available, but multiple CNNs execute as separate threads on the host processor invoking the accelerator in-turns, or 2) when multiple accelerator instances are available and multiple CNNs executing as separate threads use them concurrently. We refer to these scenarios as multithreaded and multiaccelerator mode of executions, respectively. We use a commercial-off-the-shelf (COTS) platform for experiments (Section III-A1) which has multiple host CPU cores to realize multithreading and is configurable to implement multiple accelerator instances. It also supports different sizes of accelerator.

While our characterization is specific to CNNs and for a specific platform, many of the observations are generic in nature and applicable to other systems as well. To support design space exploration (DSE), we also propose an analytical model to quickly estimate the throughput under concurrent execution. Specifically, we claim the following key contributions.

1) We perform a detailed experimentation and measure the throughput and energy consumption for 104 different CNNs, in various multithreading and multiaccelerator modes, for three different sizes of accelerator.
2) We develop various insights into functioning of CNN accelerators leading to further research problems.
3) We propose an analytical model (< 1% average error) to estimate the throughput for the multithreading scenario.

Manuscript received 5 August 2024; accepted 9 August 2024. This work was supported by the Science and Engineering Research Board (SERB) under Project SRG/2022/001083. This manuscript was recommended for publication by A. Shrivastava. *(Corresponding author: Rajesh Kedia.)*

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Hyderabad, Hyderabad 502285, India (e-mail: muneeb.bagdali@cse.iith.ac.in; rkedia@cse.iith.ac.in).

Digital Object Identifier 10.1109/LES.2024.3446896

II. Related Work

Many research works have proposed CNN accelerators [1], [2], [3]. However, most of their design and analysis is focused on standalone performance of the accelerator without any consideration for concurrent CNNs. Recently, a few works analysed and improved the speed of concurrent CNN accelerators [7], [8], [9]. However, their concurrency implies use of more than one accelerator and they do not consider using only one accelerator in a multithreaded context. Another work considers time multiplexing of an field programmable gate array (FPGA)-based accelerator [10] in a server–client setup but focuses on improving scheduling and utilization of FPGA accelerator across various users. PERSEUS [11] evaluates performance and cost for multitenant CNN execution for CPU-GPU servers but uses only two CNN models. Shafi et al. [5] characterized CNNs for GPU-based devices and frameworks while Hadidi et al. [6] characterized CNN execution on different edge devices. Both these works do not consider the concurrent execution scenarios. We characterize and analyze the behavior of a wide range of CNN models for concurrent execution (multithreading and multiaccelerator) scenarios.

III. Characterizing CNN Execution

A. Experimental Setup

1) Hardware Platform: We use Xilinx ZCU102 board having four ARM CPU cores and a programmable FPGA logic which is used to implement a COTS CNN accelerator named deep learning processor unit (DPU). We use v4.1.0 of the DPU IP (DPUCZDX8G) and the Vitis AI 3.0 framework for compiling standard CNN models for execution on DPU. We use three different sizes of DPUs—B4096, B2304, and B512; which we refer as large, medium, and small sized accelerator, respectively. ZCU102 board can support three instances of large, four instances of medium, or eight instances of small sized accelerator. Based on the CNN characteristics, the Vitis AI compiler maps a layer on DPU if it is supported by the DPU variant or otherwise maps it to the CPU.

2) Workloads: We use 104 standard CNN models from Model Zoo [12] to form the workloads. A workload

TABLE I
SELECTED 10 CNNS DISCUSSED IN DETAIL (OUT OF 104 CNNS CHARACTERIZED)

Model Name (as in Model Zoo repository)	Wokload Name	Single thread CPU time (ms)			Single thread Acc. time (ms)		
		Large	Med.	Small	Large	Med.	Small
pt_fadnet_sceneflow_576_960_0.65_154G_3.0	fad	8.0	8.0	7.9	16.2	27.2	120.3
pt_psmnet_sceneflow_576_960_0.68_696G_3.0	psm	106.0	106.8	105.5	62.9	83.5	191.5
pt_salsanextv2_semantic-kitti_64_2048_0.75_33.27G_3.0	salsa	9.5	9.2	9.4	120.4	143.2	260.8
tf_superpoint_mixed_480_640_52.4G_3.0	super	454.3	456.4	445.4	67.4	116.8	365.4
tf_yolov4_coco_416_416_60.3G_3.0	yolo	3.5	3.7	3.0	71.9	119.7	441.4
tf_vgg16_imagenet_224_224_0.43_17.67G_3.0	vgg16	0.2	0.2	42.7	24.9	36.9	119.9
tf_vgg19_imagenet_224_224_0.24_29.79G_3.0	vgg19	0.2	116.6	123.2	51.5	52.9	206.4
tf_inceptionresnetv2_imagenet_299_299_26.35G_3.0	in2	0.4	0.4	1474.9	48.6	64.5	191.7
tf_inceptionv3_imagenet_299_299_0.2_9.1G_3.0	in3	0.3	0.4	0.4	15.7	22.6	74.1
tf_mobilenetv2_1.0_imagenet_224_224_602M_3.0	mob2	0.3	0.3	0.3	3.9	4.0	9.9

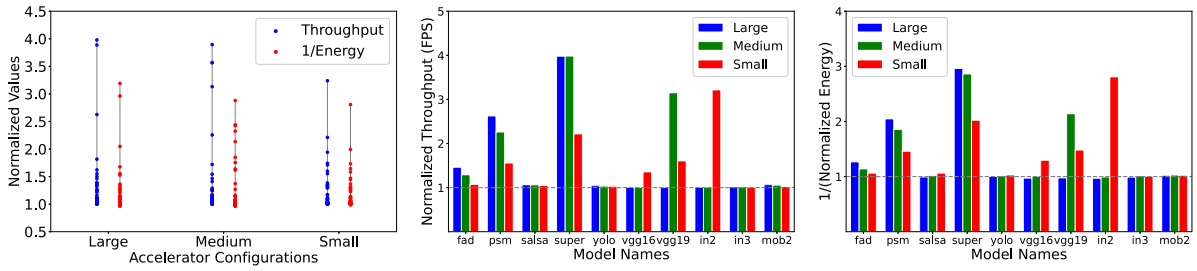


Fig. 1. Throughput and energy improvement with 4 threads, 1 accelerator (normalized to single thread). (a) Throughput and energy improvement. Each point refers to one out of 104 workloads. (b) Throughput improvement for selected CNNs. (c) Energy improvement for selected CNNs.

consists of one CNN being continuously executed back to back for a fixed duration. These models span various CNNs like FadNet, Inception (v1, v2, v3, v4), ResNet (v1, v2), SalsaNext, VoxelNet, YOLO (v4), RefineDet, VGG16, VGG19, EfficientNet, SqueezeNet, and MobileNet (v1, v2, v3) and include their varying resolutions and pruning ratios. While we characterize all CNNs, we choose ten of them (Table I) based on their CPU and accelerator times to illustrate important observations.

3) Execution Flow and Measurement Setup: We generate three different platform configurations, each containing one instance of the large, medium, and small sized accelerator, to be used for the multithreaded scenario. Three additional configurations containing three, four, and eight instances of the large, medium, and small sized accelerators are created for the multiaccelerator scenario. We execute each workload for 60 s which allows a large number of executions of the CNN. We specify the number of threads when invoking the workloads. Based on the platform configuration, the Vitis runtime system uses the available accelerator instances, thereby supporting multithreading and multiaccelerator scenarios.

We obtain the average throughput in the form of frames per second (FPS) and the average times taken by the CPU and the accelerator using the *xdputil* utility and *profiler* provided within Vitis AI framework. The power is measured using the software accessible on-board current sensors at every 1 ms and accumulated to calculate the energy consumption.

B. Observations and Insights

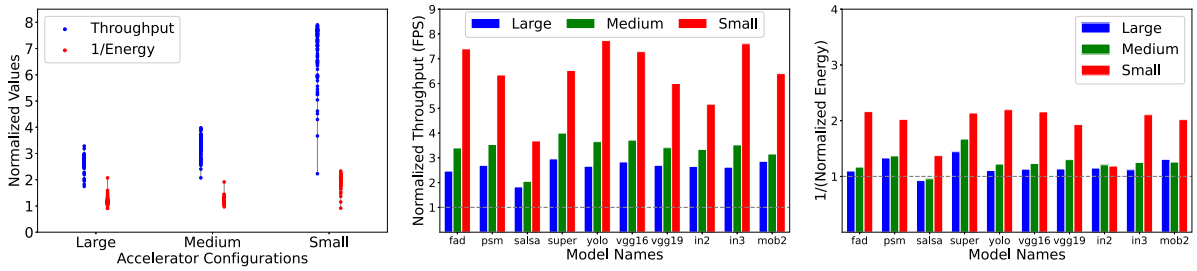
1) Multithreading With Single Accelerator Instance: Since the number of CPU cores on this platform is fixed as four, we execute various workloads with a multithreading level of four to answer the following question—How much throughput can be improved with only a single CNN accelerator instance?

Fig. 1(a) shows the normalized throughput for all the 104 workloads for different sized accelerators. The throughput increases by upto 3.98× even with only one accelerator instance. CNNs having a larger fraction of the execution time on CPU core compared to the accelerator show larger improvements. This improvement is primarily because when different CNN instances are executing as different threads, one CNN might be using the DPU while some other can execute its CPU portion. Additionally, multiple CPU cores can enable upto four CNN instances to execute their CPU portions concurrently.

Fig. 1(b) shows the normalized performance under multithreaded execution for ten selected workloads (Table I). The workload *super* achieves a speedup close to 4 for large and medium size of the accelerator because of a significantly larger CPU time than accelerator time. For most of the workloads, the speedup decreases when moving from a bigger to a smaller accelerator. This is because the accelerator time increases from bigger to smaller accelerator causing the fraction of CPU time to decrease. However, for *vgg16, vgg19, and in2* workloads, smaller accelerators depict more speedup. As shown in Table I, the CPU time is drastically higher for these workloads for smaller sized accelerators as some layers that cannot fit on the smaller accelerator are executed on the CPU core.

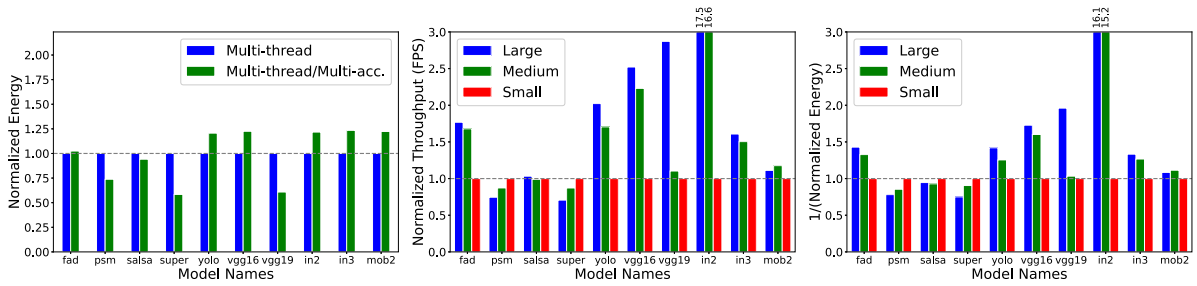
Multithreading not only improves the throughput, but also improves (reduces) the energy consumption per inference.[1] Across the workloads, Fig. 1(a) shows that multithreading reduces the energy consumption per inference by a factor of upto 3.20×. Such a reduction happens because the power consumption does not increase much with multithreading and therefore, more inferences can be executed for the same energy due to increased throughput. However, for a few CNNs having a negligible CPU time compared to the accelerator time, the energy per inference increases slightly (about 1%–3%). This is most likely due to switching overheads associated with multithreading. Fig. 1(c) shows the energy reduction normalized

[1]We plot energy reduction as an energy improvement (>1) by taking the inverse so that it is easier to visually correlate throughput and energy.



(a) (b) (c)

Fig. 2. Throughput and energy improvement with multi-accelerators (normalized to single thread). (a) Throughput and energy improvement. Each point refers to one out of 104 workloads. (b) Throughput improvement for selected CNNs. (c) Energy improvement for selected CNNs.



(a) (b) (c)

Fig. 3. Throughput and energy comparison across different implementations. (a) Energy comparison of multi-accelerator and multi-threading for Medium-sized accelerator. (b) Throughput improvement with multi-accelerators (normalized to Small-sized). (c) Energy improvement with multi-accelerators (normalized to Small-sized).

to a single-thread execution for the ten selected CNNs. The energy reduction follow a similar trend as the throughput improvement, which was discussed earlier.

2) Using Multiple Instances of CNN Accelerator: We now consider platform configurations with multiple accelerator instances implemented and accessed by different threads executing on the CPU cores. While the previous multithreading scenario improved throughput and energy for CNNs having considerable CPU time in comparison to the accelerator time, the multiaccelerator mode improves throughput for almost all workloads, as shown in Fig. 2(a). The improvement depends upon the number of accelerator instances, which is three, four, and eight for the large, medium, and small sized accelerators, respectively.

Fig. 2(b) shows that the throughput improves by a factor close to, but smaller than the number of accelerator instances. This is because multiple accelerators share internal buses and memory bandwidth which might slow down their execution. Fig. 2(c) shows the improvement in energy for the ten selected models. There are two contending factors determining the energy per inference: 1) the power consumption, which increases due to multiple accelerator instances and 2) the time per inference, which decreases due to improved throughput. In comparison to the throughput, the energy per inference improves (reduces) only by a small amount.

3) Comparing Multithreading With Multiaccelerator: While multiaccelerator mode provides similar or better throughput than the multithreaded mode, it does not apply to energy consumption. Fig. 3(a) shows the relative energy consumption for the medium sized accelerator. Some workloads have lesser energy for multithreaded mode while others have lesser energy for multiaccelerator mode depending upon their time on CPU core and accelerator. This opens up a DSE problem to identify appropriate level of multithreading and number of accelerators to use.

4) Effect of Accelerator Size: A larger accelerator executes a workload faster, but only a fewer instances can be implemented compared to a smaller accelerator which takes longer to execute a workload but allows more number of concurrent instances. We study the effect of accelerator size with workloads executing with three, four, or eight instances of large, medium, or small sized accelerator, respectively.

Fig. 3(b) and (c) shows the throughput and energy per inference for each workload normalized to that of the small sized accelerator. For most of the workloads, throughput increases and energy decreases with an increase in accelerator size (similar to prior works [9]). This is primarily because the number of concurrent multiply and accumulate (MAC) operations within a larger accelerator is much larger than the ratio of number of instances compared to a smaller accelerator. However, our characterization identifies that a few workloads deviate from this trend which we discuss in detail.

For the *psm* and *super* workloads, the highest throughput as well as lowest energy is obtained for the small sized accelerator (instead of the large). This is because apart from the accelerator, these models execute on the CPU core for a significant amount of time (see Table I). Hence, they benefit more from larger number of accelerator instances than from faster accelerators. The *mob2* workload achieves the best throughput and energy consumption with the medium sized accelerator. This CNN has execution time of 3.9, 4.0, and 9.9 ms on the large, medium, and small sized accelerators, respectively (Table I). Since the time for large and medium accelerators is almost similar, *mob2* gets benefit from having more instances of medium sized accelerator. However, its execution time is drastically higher for the small sized accelerator, making the medium sized accelerator to be best for it. While prior works [9] proposed a larger accelerator to be always used if available, we observe that a medium or small accelerator could also be better based on the workload characteristics.

This presents another DSE problem to identify a suitable size of accelerator for a given workload.

The *vgg19* and *in2* workloads experience a very high improvement for larger accelerators as some of the layers cannot execute on a smaller DPU and get mapped to CPU. This reduces the throughput for smaller accelerators and hence higher improvement is seen for larger accelerators. This can be considered as one limitation of the end-to-end framework associated with this accelerator (DPU). Ideally, the compiler framework should have split the layer into smaller units to execute on accelerator rather than on the CPU core. However, with fast emergence of newer types of layers and CNN architectures, it is also a practical approach followed by most frameworks to migrate unsupported layers to the host CPU.

IV. System Modeling for Throughput

Having seen that multithreading and multiaccelerator execution of CNNs can improve the throughput, and based on the need for DSE to identify appropriate level of multithreading and count of accelerators, we propose an analytical model to estimate the throughput for multithreading scenario. Multiaccelerator scenario involves complex modeling to estimate contention among accelerators [8] and is left as future work.

For a system with n CPU cores, we want to estimate the throughput for a multithreading level of n and with only 1 instance of the DPU. We assume that the CPU time (c) and DPU time (d) for a single-thread execution are already known. DPU execution can completely hide the CPU execution time if $c <= (n-1) \times d$, as illustrated in Fig. 4 for $n = 4$ and $c = 3 \times d$. Therefore, in such cases, the estimated throughput is determined by only the DPU time (d).

However, when CPU time becomes further long, i.e., $c > (n-1) \times d$; then, the total time for DPU execution for other threads is not sufficient for a CPU to complete its processing to again invoke the DPU. Therefore, DPU's utilization reduces and even a single DPU instance can be available to all threads on need. In such cases, the single-thread throughput can increase by a factor n. Therefore, a simple analytical model (under ideal conditions) for the throughput (T) is

$$T = \begin{cases} \frac{1}{d}, & \text{if } c <= (n-1) \times d \\ \frac{n}{c+d}, & \text{if } c > (n-1) \times d. \end{cases} \quad (1)$$

Evaluating Throughput Estimation: Fig. 5 shows the percentage error between the measured and estimated throughput. The error values are positive indicating that our predictions are an upper bound. Even with a simple model, the estimation error is within 5% (except for 6 outlier points out of 312). The mean error is < 1% (0.65%, 0.56%, and 0.67% for the large, medium, and small sized accelerators, respectively). The high error for some workloads (e.g., *vgg16, vgg19* for medium sized accelerator) is because their CPU time is very large and concurrent execution on four CPU cores experiences slowdown due to shared resources (e.g., LLC).

V. Conclusion

We presented detailed characterization of 104 different CNN workloads executing on an FPGA-based COTS CNN accelerator named DPU. Throughput and energy improves with multithreading and multiple instances of the accelerator. Our study also included three different sizes of accelerator and derived key insights: 1) how different CNNs are affected with multithreading and multiaccelerators; 2) how the size of accelerator affects throughput and energy; and 3) need for DSE as per the workload. We also developed a model to estimate the throughput improvements due to the use of multithreading, which matches the measurements quite closely.

In the future, we would like to develop estimation models for multiaccelerator mode, evaluate other platforms and architectures, and expand the framework to support DSE.

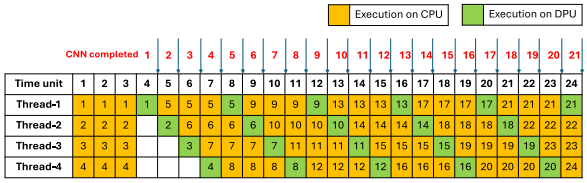


Fig. 4. CNN execution with four threads and one accelerator with $c = 3 \times d$.

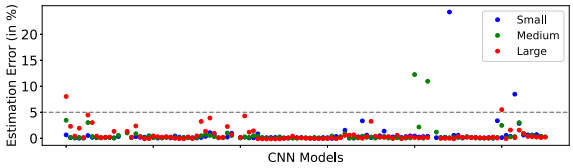


Fig. 5. Error in throughput estimation for 104 CNNs and 3 sizes.

References

[1] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

[2] K. Guo et al., "Angel-eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.

[3] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of FPGA-based neural network inference accelerators," *ACM Trans. Reconfig. Technol. Syst.*, vol. 12, no. 1, pp. 1–26, 2019.

[4] J. Peng et al., "Multi-task ADAS system on FPGA," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Hsinchu, Taiwan, 2019, pp. 171–174.

[5] O. Shafi, C. Rai, R. Sen, and G. Ananthanarayanan, "Demystifying TensorRT: Characterizing neural network inference engine on Nvidia edge devices," in *Proc. IEEE Int. Symp. Workload Character. (IISWC)*, Storrs, CT, USA, 2021, pp. 226–237.

[6] R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, and H. Kim, "Characterizing the deployment of deep neural networks on commercial edge devices," in *Proc. IEEE Int. Symp. Workload Character. (IISWC)*, 2019, pp. 35–48.

[7] Z. Du, W. Zhang, Z. Zhou, Z. Shao, and L. Ju, "Accelerating DNN inference with heterogeneous multi-DPU engines," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.

[8] S. Goel, R. Kedia, M. Balakrishnan, and R. Sen, "INFER: INterFerence-aware estimation of runtime for concurrent CNN execution on DPUs," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, 2020, pp. 66–71.

[9] R. Kedia, S. Goel, M. Balakrishnan, K. Paul, and R. Sen, "Design space exploration of FPGA-based system with multiple DNN accelerators," *IEEE Embed. Syst. Lett.*, vol. 13, no. 3, pp. 114–117, Sep. 2021.

[10] T. Nakamura, S. Saito, K. Fujimoto, M. Kaneko, and A. Shiraga, "Spatial- and time- division multiplexing in CNN accelerator," *Parallel Comput.*, vol. 111, Jul. 2022, Art. no. 102922.

[11] M. LeMay, S. Li, and T. Guo, "PERSEUS: Characterizing performance and cost of multi-tenant serving for CNN models," in *Proc. Int. Conf. Cloud Eng. (IC2E)*, 2020, pp. 66–72.

[12] (Xilinx Inc., San Jose, CA, USA). *Vitis AI Model Zoo*. Accessed: Jan. 31, 2024. [Online]. Available: https://xilinx.github.io/Vitis-AI/3.0/html/docs/workflow-model-zoo