Memory-Centric Benchmarking of Neural Network Workloads on AMD Versal AI Engines

Leonardo Fazzini[1], Valerio Rughetti[2], Marco Santic[1] and Giacomo Valente[1]

***Abstract*— Heterogeneous system-on-chip platforms integrating specialized accelerators are increasingly central to next-generation industrial and healthcare applications driven by the Industry 5.0 paradigm. Among these, AMD Versal devices combine programmable logic, general-purpose cores, and tightly coupled AI Engines, enabling neural network deployment through model-driven frameworks such as Vitis AI. While these frameworks reduce deployment effort, they provide limited guidance on achievable system-level performance, particularly with respect to external memory bandwidth. This paper addresses memory-centric benchmarking for neural network workloads on AMD Versal platforms by proposing a custom architecture, referred to as *NN memory bomb*, explicitly designed to stress the external memory subsystem and expose the maximum sustainable memory bandwidth. We evaluate the approach on a Versal AI Core XCVC1902 on the VCK190 board and compare the proposed *NN memory bomb* against LeNet, miniGoogleNet, and U-Net. The benchmark sustains up to 23,585 MB/s, approaching the peak theoretical DDR bandwidth of 25.6 GB/s, and provides a practical upper bound for evaluating memory sustainability of neural network deployments in industrial and medical embedded environments.**

I. Introduction

Addressing increasingly stringent performance requirements under power and cost constraints in embedded systems has led to the adoption of a wide range of heterogeneous computing platforms. Representative examples include embedded platforms such as NVIDIA Jetson devices [1], AMD Versal [2] and Zynq UltraScale+ [3] *systems-on-chip* (SoCs). These platforms typically integrate multiple computing engines and memory hierarchies, making efficient deployment and performance assessment non-trivial [4]. In this respect, their role is becoming increasingly central in next-generation industrial and medical environments, where AI-driven applications must operate reliably under tight resource constraints. Within this context, the *AMD Versal platform* (Versal platform hereinafter) has emerged as a prominent platform for high-performance embedded computing, supporting a wide range of data-intensive workloads, including neural network inference [5], [6]. As such, it represents a relevant candidate for Industry 5.0 and Healthcare 5.0 scenarios, where embedded AI inference is required to sustain high-throughput data processing while guaranteeing predictable and transparent system-level behavior. Versal devices integrate heterogeneous computing resources, including a *programmable logic* (PL) fabric, general-purpose processing cores, and a tightly coupled array of AI Engines, enabling a wide spectrum of acceleration strategies for data-intensive workloads. At the same time, current deployment workflows for artificial intelligence increasingly rely on tools that enable a rapid transition from high-level models to efficient hardware implementations [7]–[9]. By abstracting low-level architectural details and automating large portions of the design process, these tools allow domain experts to interact more directly with the implementation phase. Representative industrial examples include MathWorks toolchains for deep learning deployment, such as the Deep Learning Toolbox [10], as well as Intel-provided solutions for neural network inference acceleration, e.g., OpenVINO [11]. In parallel, the academic community has proposed methodologies and frameworks targeting the reduction of deployment time for embedded platforms, a challenge that has become particularly critical as system complexity and performance demands continue to grow [12].

To support the deployment of neural networks on Versal platforms, AMD provides Vitis AI [13]. Vitis AI takes as input a trained neural network model and target-specific constraints, and produces an optimized implementation mapped onto the available hardware accelerators. Notably, the framework is not limited to Versal devices, but is provided for Zynq UltraScale+ and Zynq-7000 families. While Vitis AI reduces the effort required to deploy neural networks on embedded hardware, it provides limited guidance on how close a deployment may operate to the fundamental performance limits of the target platform. As a result, designers often **lack a clear understanding of the system-level corner case**, that is, the maximum level of memory pressure and performance that the platform can sustain under neural network workloads. In this context, benchmarking allows the characterization of architectural limits and bottlenecks [14], but existing studies either assume direct control over hardware resources [15], [16] or do not explicitly target system-level corner cases under neural network workloads [6], [17].

In this work, we address this gap by focusing on **memory-centric benchmarking** of neural network deployments on the Versal AI Engine through the Vitis AI framework. Specifically, we introduce a custom neural network architecture explicitly designed as a memory bomb, referred to as *NN memory bomb*, to stress the external memory subsystem and to empirically determine the maximum average memory bandwidth achievable under realistic deployment conditions. The characterization provides an upper bound and a reference point for reasoning about the scalability of neural network deployments before committing to full system integration.

[1]Leonardo Fazzini, Marco Santic and Giacomo Valente are with the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy

[2]Valerio Rughetti is with LUMSA University, Rome, Italy

After providing a related work analysis (Section 2), the following contributions are presented: **(i)** the design of a custom neural network architecture, referred to as *NN memory bomb*, explicitly conceived as a memory bomb to stress the memory subsystem of the Versal AI Engine, enabling the characterization of the maximum external memory bandwidth under Vitis AI deployment constraints (Section 3); **(ii)** a comparative evaluation against representative neural network architectures, including LeNet, miniGoogleNet, and U-Net, with the goal of positioning the proposed *NN memory bomb* as a worst-case, memory-intensive reference workload and of assessing how close realistic neural network deployments operate with respect to the maximum external memory bandwidth of the Versal platform (Section 4); **(iii)** a discussion of insights and future research directions (Section 5).

II. Related Work

The work in [15] proposes an architectural benchmarking model to evaluate the floating-point performance of a massively parallel dataflow overlay on Versal platforms, targeting both digital-signal processor (DSP) resources in the PL and the AI Engine array. Similarly, the study in [16] evaluates the performance of Versal AI engine using finite-impulse response filters and fast-fourier transform as representative DSP kernels, with a focus on throughput, latency, power consumption, and hardware efficiency. However, both [15], [16] assume fine-grained programmability and direct control of individual AI Engine elements. In contrast, the scenario considered in this work relies on the Vitis AI framework, which abstracts the underlying AI Engine architecture and does not expose explicit, element-level programmability of AI Engine tiles. As a result, the assumptions underlying these benchmarking approaches do not directly apply to model-driven neural network deployments realized through Vitis AI. On a different line, the work in [18] proposes XVDPU, a high-performance convolutional neural network accelerator for Versal platforms that leverages the AI Engine to improve throughput and energy efficiency. The study primarily focuses on compute-centric optimizations and end-to-end inference performance. However, it does not explicitly characterize the maximum external memory bandwidth achievable by the AI Engine, which is a central aspect of the present work. Several studies have evaluated neural network deployments using the Vitis AI framework, highlighting its relevance in neural network deployment. Among these studies, the work in [6] evaluates Versal and Zynq UltraScale+ platforms across a broad set of convolutional and fully connected neural networks. The analysis investigates performance trends as a function of the computation-to-memory ratio and reports memory transfer behavior as an observed metric. While this provides valuable comparative insights, the study does not aim at identifying or saturating the maximum external memory bandwidth of the Versal AI Engine. Along a related direction, the work in [17] performs an in-depth analysis of memory interference in FPGA-based heterogeneous SoCs, including Versal devices, and experimentally shows that FPGA accelerator clusters can generate up to 24 GB/s of external DRAM traffic, corresponding to approximately 94% of the nominal bandwidth. However, their work focuses on interference characterization across multiple processing elements, and does not explicitly target the characterization of AI Engine corner cases under neural network workloads deployed through Vitis AI.

III. Proposed Approach

This section describes the overall approach adopted in this work. First, a reference deployment scenario is introduced to fix the hardware and software context considered throughout the study, relying on a commonly adopted Versal-based implementation. Then, the design of the proposed *NN memory-bomb* is presented, highlighting the architectural choices intentionally made to stress the external memory subsystem. Finally, the deployment of the *NN memory-bomb* within this reference scenario is discussed, with the goal of identifying the maximum average memory bandwidth achievable on the target platform under realistic deployment constraints.

A. Reference scenario

As reference deployment scenario, we consider a system based on a Versal AI Core SoC integrated on the VCK190 evaluation board [19]. This choice of the Versal AI Core platform (hereinafter referred to as the Versal platform) reflects its role as a well-established and widely adopted baseline for evaluating AI Engine–based acceleration, and its extensive use in both academic studies and industrial prototyping activities. The Versal platform features a heterogeneous architecture combining an AI Engine array with 400 AI engine elements (very-long instruction word processors), a PL fabric, a high-performance Network-on-Chip (NoC), and general-purpose processing cores, including Arm Cortex-A72 and Cortex-R5 processors. The NoC provides high-bandwidth connectivity among the heterogeneous components, the PL, and the external memory subsystem. The VCK190 board is equipped with external DDR memory providing a peak theoretical bandwidth of approximately 25.6 GB/s [17]. This memory subsystem constitutes a critical shared resource for AI Engine–based acceleration and therefore represents a key element for the memory-centric benchmarking conducted in this work. A high-level block diagram of the reference deployment scenario is reported in Fig. 1, based on the *Deep-learning Processor Unit Targeted Reference Design* (TRD) for VCK190 [20], a validated and well-documented baseline widely adopted for deploying and evaluating neural network workloads on Versal platforms [21]. The DPUCVDX8G accelerator integrates the AI Engine array and the PL as an overlay (see Fig. 1), enabling neural network deployment through the Vitis AI framework. The AI Engine array executes convolution-intensive kernels, while interface tiles orchestrate data movement through the NoC. AI Engine elements are grouped into batch handlers, each mapped to a private set of tiles to support concurrent batch processing, while control logic and model parameter storage are shared across batches to maximize hardware utilization. Within the TRD, several architectural parameters

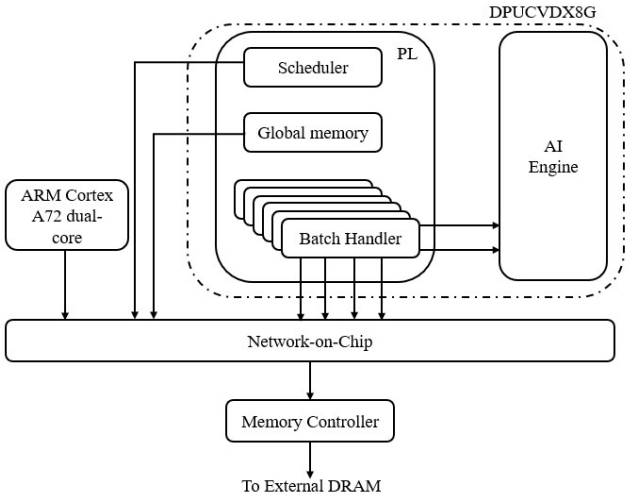

Fig. 1. Reference deployment scenario on the VCK190 platform: the DPUCVDX8G accelerator integrates the AI Engine array and the PL, connected through the NoC to the external DDR memory via AXI4-based interfaces. Connections between the Arm Cortex-A72 processors and the DPUCVDX8G are omitted for clarity.

Input: $(H, W, 3)$
$\big[$Conv2D + BatchNorm + MaxPool$\big] \times 5$
GlobalAveragePooling
Dense(1024) + Dropout
Dense(512) + Dropout
Dense(256)
Dense(128)
Dense(64)
Dense(10)

Fig. 2. High-level architecture of the proposed *NN memory bomb*. The network consists of five repeated Conv2D–BatchNorm–MaxPool blocks with progressively increasing feature map depth, followed by a GlobalAveragePooling layer and a fully connected classifier. This design is explicitly conceived to maximize sustained memory traffic between the AI Engine and the external DDR memory subsystem.

can be configured, such as the number of batch handlers implemented on the PL side. In this work, the DPUCVDX8G configuration corresponds to the default configuration provided by the TRD, identified as `C32B6CU1L2S2`. This choice is motivated by the fact that the Vitis AI reference design for VCK190 is based on this configuration and that, for designers planning an initial deployment of neural network workloads, it represents the most straightforward and readily available option. In the `C32B6CU1L2S2` notation, `C32` denotes the number of AI Engine elements allocated to each batch handler, `B6` indicates the presence of six batch handlers, `CU1` specifies a single compute unit, and `L2S2` denotes the presence of two high-performance interfaces per batch handler. These interfaces are used to transfer input images, activations, and feature maps between the DPU and the NoC, and ultimately to the external DDR memory. From a software perspective, the system runs a PetaLinux operating system on the dual-core Arm Cortex-A72 processors. Neural network compilation and deployment are performed using the Vitis AI framework, version 3.0 [22].

B. Neural Network Memory Bomb

The architecture of the proposed *NN memory bomb* is explicitly designed to maximize memory traffic between the AI Engine and the external DDR memory. Unlike conventional neural networks, which are typically optimized for accuracy or computational efficiency, this architecture is developed *by design* to stress the memory subsystem and to expose bandwidth-related performance limits of the target platform. A high-level representation of the proposed architecture is shown in Fig. 2. The network adopts a deep convolutional structure composed of repeated convolution–batch normalization–pooling blocks with progressively increasing numbers of feature maps. This design choice is motivated by the observation that convolutional layers operating on large feature maps dominate memory traffic in AI Engine–based accelerators. Each convolution generates intermediate activations that must be repeatedly loaded from and stored to external memory, while batch normalization further increases memory accesses due to additional read and write operations on feature maps. Pooling layers reduce the spatial resolution but preserve the depth of the feature maps, in turn maintaining a high volume of memory transfers across successive layers. The repetition of the convolution–batch normalization–pooling block five times is intentional. Increasing the number of blocks allows memory traffic to scale with network depth while progressively increasing the number of channels, which amplifies feature map sizes in the early and intermediate layers. This configuration enables the network to reach a memory-bound execution regime without relying on excessively large input sizes. The network input is defined as $(H, W, 3)$, where H and W denote the spatial resolution and the third dimension corresponds to the red-green-blue color channels. By varying H and W, the input size directly controls the size of the initial feature maps and, consequently, the overall memory traffic generated throughout the network. This parameterization enables systematic exploration of memory behavior as a function of input resolution. To limit computational complexity while preserving memory pressure, a global average pooling layer is introduced before the fully connected classifier. This operation collapses the spatial dimensions of the feature maps into a single vector per channel, reducing the number of parameters and arithmetic operations in the subsequent dense layers. As a result, execution remains dominated by memory transfers rather than by compute-intensive matrix multiplications. The fully connected classifier consists of a sequence of dense layers with decreasing numbers of units, from 1024 down to the final output layer with 10 units. This gradual reduction serves two purposes. First, it avoids introducing a sudden computational bottleneck that could shift execution toward a compute-bound regime. Second, it preserves a realistic classification pipeline similar to those

used in practical deep learning models. Dropout layers are included to reflect common deployment practices, without affecting memory traffic during inference.

C. Memory Bomb Deployment on the Reference Scenario

The proposed *NN memory bomb* is compiled and deployed on the reference Versal platform through the Vitis AI model-driven flow, which maps the network onto the DPUCVDX8G accelerator without manual hardware intervention. During compilation, architecture-aware optimizations such as layer fusion, operator rewriting, and scheduling transformations are applied, producing a runtime execution model tailored to the DPU architecture rather than directly reflecting the original network graph. Performance profiling is conducted using the Vitis Analyzer tool in non-debug mode [23], ensuring that reported metrics reflect the steady-state behavior of the deployed accelerator and its interaction with the external memory. The results are reported in Table I and analyze the impact of the input resolution on both computational and memory-related metrics. In the table, *Workload* denotes the computational demand of the neural network subgraph expressed in GOP, where each MAC operation is counted as two operations. *SWRT* and *HWRT* represent the software-managed and hardware execution times, respectively, both expressed in milliseconds. *Efficiency* reports the achieved computational throughput of the DPU in GOP/s. Memory-related metrics include *LdWB*, corresponding to the amount of weights and biases loaded from external memory, *LdFM* and *StFM*, representing the cumulative amount of feature map data loaded from and stored to external memory, respectively, and *AvgBW*, which captures the sustained average external memory bandwidth. In Table I, the labels 128, 256, 320, 512, and 1024 denote the spatial resolution of the network input, corresponding to square input tensors of size $(H, W, 3)$ with $H = W$. Increasing the input resolution directly increases the spatial dimensions of intermediate feature maps throughout the network, in turn amplifying memory traffic. As expected, both *LdFM* and *StFM* increase monotonically with the input resolution, reflecting the growth of intermediate feature map sizes. This trend is also reflected in the increase of *Workload*, *HWRT*, and *SWRT*, which scale with the amount of computation and data movement required by the network. In contrast, *Efficiency* remains relatively stable across configurations, indicating that the DPU sustains a comparable computational throughput while progressively increasing memory pressure. Notably, *LdWB* remains constant across all configurations, as the size of weights and biases is determined solely by the network architecture and is independent of the input resolution. This confirms that the observed increase in memory traffic is dominated by feature map transfers rather than by model parameters. The behavior of *AvgBW* highlights a different trend. The average external memory bandwidth initially increases with the input resolution, but eventually reaches a saturation regime imposed by the external DDR memory subsystem. Notably, the maximum *AvgBW* is observed for an input resolution of $(320, 320, 3)$. Beyond this point, further increases in input resolution lead to higher cumulative memory traffic, as evidenced by *LdFM* and *StFM*, but do not translate into higher sustained bandwidth. This behavior can be attributed to increased pressure on the memory controller and NoC interconnect when handling excessively large feature maps: at higher resolutions, the volume of concurrent memory transactions exceeds the optimal operating point of the DDR controller, introducing overhead that reduces the sustained average bandwidth despite the higher cumulative traffic. Based on these observations, the memory-bomb configuration with input resolution $(320, 320, 3)$ is selected as the representative workload for subsequent comparisons. This configuration effectively stresses the external memory subsystem while operating at the maximum bandwidth achievable on the reference platform, exposing the system-level corner case targeted in this work.

IV. Experimental Results

This section presents the experimental evaluation of the proposed memory-centric benchmarking approach. The overall objective of the experiments is twofold: first, to assess the ability of the proposed *NN memory bomb* to act as a worst-case workload for stressing the memory subsystem; second, to demonstrate how such a benchmark can be used to guide system designers in evaluating, *a priori*, whether a given neural network configuration can be sustained by the target Versal platform under realistic deployment constraints. All experiments are conducted on the reference deployment scenario described in Section 3.1, using the Vitis AI deployment flow. The evaluation focuses on memory-related profiling metrics introduced in Section 3.3. In the first set of experiments, the proposed *NN memory bomb* is compared against two representative architectures, namely LeNet [24] and miniGoogleNet [25]. LeNet is a classical convolutional neural network originally introduced for digit recognition tasks and is widely used as a lightweight baseline due to its shallow structure and limited number of parameters. miniGoogleNet, derived from the GoogLeNet family, introduces deeper convolutional pipelines and inception-style modules while remaining suitable for embedded deployment thanks to its reduced computational footprint. These two networks are selected as representative baselines because they reflect common design points in embedded deep learning. LeNet exemplifies compact workloads characterized by limited feature map sizes and low overall memory volume, whereas miniGoogleNet represents a more complex architecture with increased depth and feature map reuse, yet still constrained for edge platforms. Together, they provide a meaningful contrast to the proposed memory-bomb network. The results are reported in Table II. Among the evaluated networks, the proposed *NN memory bomb* achieves the highest average external memory bandwidth (AvgBW), together with larger cumulative feature map load (LdFM) and store (StFM) values, clearly identifying it as the most memory-intensive workload. Interestingly, despite its limited memory footprint, LeNet exhibits a relatively high AvgBW, due to its short execution time and bursty memory transactions over a reduced temporal

TABLE I

DETAILED PROFILING RESULTS FOR THE PROPOSED *NN memory bomb* DEPLOYED ON THE VERSAL AI ENGINE USING THE DPUCVDX8G ACCELERATOR.

Neural Network	Workload (GOP)	SWRT (ms)	HWRT (ms)	Efficiency (GOP/s)	LdWB (MB)	LdFM (MB)	StFM (MB)	AvgBW (MB/s)
Memory Bomb (128)	1.493	0.519	0.446	34.6	4.900	0.283	0.000	11636
Memory Bomb (256)	7.220	4.354	4.244	17.5	4.900	36.018	35.071	17903
Memory Bomb (320)	**11.718**	**6.559**	**6.447**	**18.8**	**4.900**	**74.339**	**72.805**	**23586**
Memory Bomb (512)	31.757	19.773	19.657	16.7	4.900	214.783	210.644	21892
Memory Bomb (1024)	133.165	83.586	83.466	16.5	4.900	927.974	898.271	21939

TABLE II

DETAILED PROFILING RESULTS FOR LENET AND MINIGOOGLENET, WITH THE PROPOSED *NN memory bomb* USED AS A WORST-CASE REFERENCE, DEPLOYED ON THE VERSAL AI ENGINE USING THE DPUCVDX8G ACCELERATOR.

Neural Network	Workload (GOP)	SWRT (ms)	HWRT (ms)	Efficiency (GOP/s)	LdWB (MB)	LdFM (MB)	StFM (MB)	AvgBW (MB/s)
LeNet	0.021	0.336	0.255	0.8	6.114	0.094	0.073	22540
miniGoogleNet	0.470	0.700	0.622	7.8	1.576	4.085	2.109	12491
Memory Bomb (320)	**11.718**	**6.559**	**6.447**	**18.8**	**4.900**	**74.339**	**72.805**	**23586**

window. In contrast, miniGoogleNet shows a lower AvgBW, as its deeper structure and higher computation-to-memory ratio distribute memory accesses over longer execution times.

A second experiment is performed using U-Net [26], a widely adopted convolutional neural network architecture for semantic segmentation. From an architectural perspective, U-Net is composed of a contracting path (encoder) and an expanding path (decoder), connected through multiple skip connections. While the encoder progressively reduces the spatial resolution to extract high-level features, the decoder reconstructs dense output maps by combining coarse semantic information with high-resolution features forwarded through skip connections. This design increases memory traffic, as large feature maps must be repeatedly loaded from and stored to external memory, particularly at higher input resolutions. The memory-related profiling results for U-Net at different input resolutions are reported in Table III. As the input size increases, both LdFM and StFM grow rapidly, reflecting the substantial feature map transfers induced by the encoder–decoder structure and the skip connections. In contrast, AvgBW increases more gradually and eventually approaches a saturation regime imposed by the external memory subsystem. The proposed memory-bomb network provides a useful reference point to interpret these results. Notably, the memory-bomb architecture reaches a higher AvgBW than U-Net while maintaining feature map traffic comparable to U-Net at intermediate input resolutions (e.g., 320×320). This observation highlights the role of the memory bomb as a worst-case workload that exposes the maximum memory bandwidth of the platform under realistic deployment constraints. From a system design perspective, this comparison enables a practical form of design-space exploration. For instance, a designer aiming to increase the input resolution of a U-Net-based application can leverage the memory-bomb benchmark to determine, *a priori*, whether the target Versal platform can sustain the resulting memory demand. If the memory-bomb workload saturates the memory subsystem at a given bandwidth level, any U-Net configuration whose memory requirements remain below this bound can be safely deployed without incurring memory-induced performance degradation. Overall, the experimental results confirm that the proposed *NN memory bomb* effectively acts as an observed worst-case workload for stressing the external memory subsystem of the Versal platform. Across all evaluated scenarios, including lightweight architectures such as LeNet, more structured designs such as miniGoogleNet, and memory-intensive networks such as U-Net, the memory-bomb consistently operates at a higher AvgBW. By comparing the memory requirements of a target neural network against the bandwidth level exposed by the memory-bomb workload, system designers can assess, *a priori*, whether a given deployment is likely to remain within the memory performance limits of the Versal platform, or whether memory-induced performance degradation should be expected.

V. CONCLUSION AND FUTURE WORK

This work presented a memory-centric benchmarking approach for neural network workloads deployed on AMD Versal platforms through the Vitis AI framework, highlighting the dominant role of memory bandwidth under realistic, model-driven deployment constraints. The proposed *NN memory bomb* drives the AI Engine to sustain an average external memory bandwidth of up to 23.585 GB/s, approaching the peak theoretical DDR bandwidth of 25.6 GB/s available on the VCK190 platform, confirming the effectiveness of the Versal hardware architecture and software stack in serving highly memory-intensive neural workloads. At the same time, the results reveal an important system-level implication. A single AI Engine workload can nearly saturate the external memory bandwidth, leaving limited headroom for concurrent activities. In practical deployments, additional processing elements, such as Arm Cortex-A72 or Cortex-R5 cores, programmable logic accelerators, or multiple DPU instances,

TABLE III

DETAILED PROFILING RESULTS FOR U-NET AT DIFFERENT INPUT RESOLUTIONS, WITH THE PROPOSED *NN memory bomb* USED AS A WORST-CASE REFERENCE, DEPLOYED ON THE VERSAL AI ENGINE USING THE DPUCVDX8G ACCELERATOR.

Neural Network	**Workload (GOP)**	**SWRT (ms)**	**HWRT (ms)**	**Efficiency (GOP/s)**	**LdWB (MB)**	**LdFM (MB)**	**StFM (MB)**	**AvgBW (MB/s)**
U-Net (128)	1.603	1.361	1.282	12.9	2.059	7.319	7.125	12868
U-Net (256)	6.413	5.759	5.646	11.7	2.059	47.263	46.500	16971
U-Net (320)	10.020	8.819	8.701	11.9	2.059	73.845	72.656	17073
U-Net (512)	25.650	23.713	23.599	11.2	2.059	219.026	216.000	18521
U-Net (1024)	102.600	92.856	92.739	11.4	2.059	967.740	954.000	20744
Memory Bomb (320)	**11.718**	**6.559**	**6.447**	**18.8**	**4.900**	**74.339**	**72.805**	**23586**

may simultaneously access the shared memory subsystem. In such scenarios, unmanaged contention may compromise performance isolation and introduce timing interference across heterogeneous components. Another relevant aspect concerns the generality of the proposed methodology. While the numerical results reported in this work are specific to the selected Versal device and reference deployment scenario, the proposed benchmarking approach is not tied to a single platform. The same memory-bomb neural network can be deployed on different devices and configurations to expose their respective memory bandwidth limits, making the approach portable across heterogeneous systems while preserving target-specific quantitative insights. It is worth noting that the saturation point exposed by the *NN memory bomb* is intrinsically linked to the specific DPUCVDX8G configuration adopted (`C32B6CU1L2S2`). Scaling the number of batch handlers or AXI interfaces on the PL side may shift this saturation point, and investigating such architectural variants represents a direction for future work. Future work will also investigate timing interference induced by AI Engine workloads on shared memory subsystems, building on recent research on memory interference and predictability [17], [27] in heterogeneous systems.

REFERENCES

[1] NVIDIA Corporation, "Jetson nano product development," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, 2024, accessed: 2026-04-06.

[2] AMD, Inc., "Versal adaptive socs," https://www.amd.com/en/products/adaptive-socs-and-fpgas/versal.html, 2024, accessed: 2026-04-06.

[3] ——, "Zynq ultrascale+ mpsoc," https://www.amd.com/en/products/adaptive-socs-and-fpgas/soc/zynq-ultrascale-plus-mpsoc.html, 2024, accessed: 2026-04-06.

[4] K. Manev *et al.*, "Unexpected diversity: Quantitative memory analysis for zynq ultrascale+ systems," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 179–187.

[5] J. Menzel *et al.*, "Efficient and distributed computation of electron repulsion integrals on amd ai engines," in *2025 IEEE 33rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2025, pp. 95–104.

[6] A. Leftheriotis *et al.*, "Evaluating versal acap and conventional fpga platforms for ai inference," in *2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2023, pp. 1–6.

[7] R. Singh and S. S. Gill, "Edge ai: A survey," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023.

[8] T. Di Mascio *et al.*, "Monica vision: An approach, a model and the interactive tools for cyber-physical systems designers," in *Proceedings of the 14th Biannual Conference of the Italian SIGCHI Chapter*, 2021, pp. 1–5.

[9] V. Muttillo, G. Valente, F. Federici, *et al.*, "A design methodology for soft-core platforms on fpga with smp linux, openmp support, and distributed hardware profiling system," *Journal of Embedded Systems*, vol. 2016, no. 15, 2017.

[10] The MathWorks, Inc., "Deep learning toolbox," https://www.mathworks.com/products/deep-learning.html, 2024, accessed: 2026-04-06.

[11] Intel Corporation, "Openvino toolkit," https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html, 2024, accessed: 2026-04-06.

[12] T. Pacini *et al.*, "Fpg-ai: A technology-independent framework for the automation of cnn deployment on fpgas," *IEEE Access*, vol. 11, pp. 32 759–32 775, 2023.

[13] AMD, Inc., "Vitis ai," https://www.amd.com/en/products/software/vitis-ai.html, 2024, accessed: 2026-04-06.

[14] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. Morgan Kaufmann, 2019.

[15] M. Bouaziz and S. A. Fahmy, "Benchmarking floating point performance of massively parallel dataflow overlays on amd versal compute primitives," in *2025 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2025, pp. 99–103.

[16] P. Flores *et al.*, "Evaluation of the versal intelligent engines for digital signal processing basic core units," in *Proceedings of the International Conference on Field-Programmable Technology (FPT)*. IEEE, 2021, pp. 1–8.

[17] G. Brilli *et al.*, "Understanding and mitigating memory interference in fpga-based hesocs," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1335–1340.

[18] X. Jia *et al.*, "Xvdpu: A high performance cnn accelerator on the versal platform powered by the ai engine," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 01–09.

[19] AMD, Inc., "Vck190 evaluation board," https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/vck190.html, 2024, accessed: 2026-01-31.

[20] ——, "Dpucvdx8g targeted reference design for vck190," https://www.xilinx.com/bin/public/openDownload?filename=DPUCVDX8G_VAI_v3.0.tar.gz, 2024, accessed: 2026-01-31.

[21] F. Restuccia *et al.*, "Time-predictable acceleration of deep neural networks on fpga soc platforms," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, 2021, pp. 441–454.

[22] AMD, Inc., "Vitis ai framework v3.0," https://xilinx.github.io/Vitis-AI/3.0/html/index.html, 2024, accessed: 2026-01-31.

[23] ——, "Vitis analyzer," https://docs.amd.com/r/en-US/Vitis-Tutorials-AI-Engine-Development/Vitis-Analyzer, 2024, accessed: 2026-01-31.

[24] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[25] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[26] O. Ronneberger *et al.*, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015, pp. 234–241.

[27] G. Valente *et al.*, "Fine-grained qos control via tightly-coupled bandwidth monitoring and regulation for fpga-based heterogeneous socs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 2, pp. 326–340, 2025.