IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Journal Européen des Systèmes Automatisés
Vol. 59, No. 1, January, 2026, pp. 157-166
Journal homepage: http://iieta.org/journals/jesa

Acceleration and Energy Optimization of Convolutional Neural Networks on Zynq-7000 SoC: Comparative Analysis with GPU Platforms

O. Slimani*, K. Boudjit

Laboratory of Instrumentation (LINS), Embedded Systems Team, University of Science and Technology Houari Boumediene, Algiers 16111, Algeria

Corresponding Author Email: oussama.slimani@usthb.edu.dz

https://doi.org/10.18280/jesa.590115

Received: 11 October 2025
Revised: 10 December 2025
Accepted: 19 December 2025
Available online: 31 January 2026

Keywords:
FPGA, Zynq-7000, Convolutional Neural Networks, PYNQ, hardware acceleration, GPU, energy efficiency

ABSTRACT

Artificial intelligence has experienced rapid growth in recent years, with computer vision emerging as one of its most influential application domains. In this context, Convolutional Neural Networks (CNNs) have revolutionized image analysis and understanding by enabling the automatic extraction of hierarchical and high-level features directly from raw visual data. Despite their impressive performance, CNNs impose substantial computational and memory requirements, which significantly hinder their deployment on embedded platforms where power consumption, hardware cost, and real-time constraints are critical factors. To overcome these limitations, this work investigates a hardware acceleration approach for CNN inference based on a Zynq-7000 all-programmable system-on-chip (APSoC). This platform integrates an ARM-based processing system (PS) with FPGA reconfigurable logic, enabling efficient hardware–software co-design. The proposed architecture offloads the most computationally intensive CNN operations to custom hardware accelerators implemented in the programmable logic (PL), while high-level control and data management are handled in software on the ARM processor. Performance, resource utilization, and energy efficiency are systematically evaluated and compared against a desktop-class NVIDIA GTX 1060 GPU. Experimental results demonstrate up to 47.7× speedup over ARM execution with sub-watt power, whereas the GPU reaches 79.2 W, confirming the suitability of Zynq-7000 platforms for energy-efficient edge inference in embedded real-time systems applications.

1. INTRODUCTION

Recent advances in artificial intelligence (AI) and deep learning have enabled significant progress in application domains such as autonomous driving, robotics, medical imaging, and intelligent surveillance. Convolutional neural networks (CNNs) form the backbone of these advances due to their ability to automatically learn hierarchical representations from raw data, achieving state-of-the-art accuracy in visual and signal-processing tasks [1, 2].

Despite their effectiveness, CNNs are computationally demanding. Convolutional layers involve a massive number of multiply–accumulate operations and require high memory bandwidth, which poses serious challenges for real-time deployment on resource-constrained embedded platforms [3]. While GPUs provide high throughput for CNN training and inference, their high power consumption and thermal requirements limit their applicability in embedded and edge-computing scenarios Conversely, energy-efficient processors such as ARM CPUs offer limited parallelism and often fail to meet real-time constraints for deep CNN workloads [4, 5].

To bridge this gap, heterogeneous architectures based on programmable systems-on-chip (SoCs) have emerged as a promising solution. Platforms such as the Xilinx Zynq-7000 family integrate ARM processing cores with FPGA programmable logic (PL), enabling hardware–software co-design and custom acceleration of computationally intensive CNN layers [6, 7]. FPGA-based accelerators can exploit fine-grained parallelism, data reuse, and precision optimization, leading to significant improvements in performance per watt compared to conventional processors [8].

The main contributions of the work are presented as follows:

(i) An energy-focused hardware-software co-design approach for CNN inference on Zynq-7000 SoC;

(ii) A quantitative and reproducible comparison between FPGA platforms and a desktop-class GPU with respect to performance, energy, and resource usage;

(iii) An experimental assessment focusing on memory (BRAM) saturation as the scalability bottleneck in deploying CNNs on FPGAs;

(iv) An efficient deployment strategy with HLS and PYNQ for low-power edge inference.

2. RELATED WORKS

Over the past decade, extensive research has been devoted to accelerating CNNs through dedicated hardware, particularly using FPGAs and heterogeneous SoCs. Early

works primarily focused on proving that FPGA-based acceleration could rival GPU performance in throughput while consuming far less power. The pioneering work FINN by Umuroglu et al. [9] introduced a framework for binarized and quantized neural network inference on FPGAs, achieving high throughput with minimal energy usage. Similarly, Angel-Eye presented a complete design flow for mapping CNNs onto embedded FPGAs, emphasizing efficient dataflow and memory reuse to optimize performance. During this period, research efforts also explored bit-serial architectures, quantization strategies, and on-chip buffering schemes to overcome the limited memory bandwidth of FPGAs [9, 10].

The focus shifted toward design automation and hybrid system-on-chip (SoC) implementations that leverage both the processing system (PS) and PL of modern devices. Surveys such as that by Mittal [11] highlighted a growing reliance on high-level synthesis (HLS) and hardware–software co-design to reduce development complexity while maintaining flexibility. The introduction of Xilinx's Vitis AI framework significantly accelerated this transition by offering standardized compilation flows, quantization support, and optimized IP cores for CNN deployment on Zynq and Versal platforms. Concurrently, studies demonstrated that hybrid designs offloading convolutional and pooling layers to the PL while managing control logic in the PS achieved up to 30× energy efficiency improvements over CPU-only approaches [12, 13].

Between 2023 and 2025, the field matured, with researchers focusing on the practical deployment of deep learning models for real-time embedded inference. Recent reviews emphasize three main advancements: (1) hardware-aware quantization and pruning techniques enabling complex models such as YOLOv8 or MobileNetV3 to fit within FPGA resource limits; (2) optimized DMA-based dataflows for faster communication between PS and PL; and (3) integration of automated compiler toolchains like FINN and Vitis AI into end-to-end workflows. Experimental studies in edge-AI platforms confirmed that modern FPGAs such as the Zynq-7000 and UltraScale+ achieve a favorable balance between inference latency, power efficiency, and flexibility, establishing them as viable alternatives to GPUs for embedded vision systems [14, 15].

More recent approaches like Vitis AI, FINN, hls4ml, and TVM have highly improved the automation and deployment speeds of CNN inferencing into FPGAs. The combination of these tools might even provide an improvement in performance and a reduction in development time. However, a comparative study of the above tools is not the focus of the present study and needs to be explored as a future reference.

Building upon these prior works, our study provides a quantitative comparative analysis between FPGA-based SoCs (PYNQ-Z1, ZedBoard) and a dedicated GPU (GTX 1060), focusing on measurable metrics such as inference speed, hardware utilization, and power consumption. The findings aim to reinforce the role of FPGA–SoC architectures as energy-efficient solutions for real-time CNN inference in embedded artificial intelligence applications.

3. MATERIALS AND METHODS

3.1 Study subjects: Convolutional Neural Network architectures

LeNet-5 was selected as a lightweight CNN representative for evaluating the proposed hardware acceleration framework under embedded constraints. Originally developed for handwritten digit recognition on the MNIST dataset, its compact architecture with two convolutional and two fully connected layers makes it well suited for validating functional correctness and baseline acceleration efficiency on resource-constrained hardware platforms [16].

An intermediate-complexity CNN for CIFAR-10 classification was used to evaluate scalability under increased computational and memory demands. With three convolutional layers and color image inputs, this model exhibits higher arithmetic intensity and data movement than LeNet-5, making it suitable for assessing performance and resource utilization on embedded platforms [17].

Lastly, Mask R-CNN was chosen as a modern, high-end deep learning model for instance segmentation and object detection. Because it was very hard to run and needed a lot of memory, this network was only tested on the GPU platform to find out what the best performance levels were on a powerful, unrestricted computing system [18].

Together, these models make it possible to systematically and meaningfully compare the trade-offs between performance, resource use, and energy efficiency for lightweight, intermediate, and large-scale CNN workloads.

Figure 1 illustrates the architecture of the LeNet convolutional neural network.

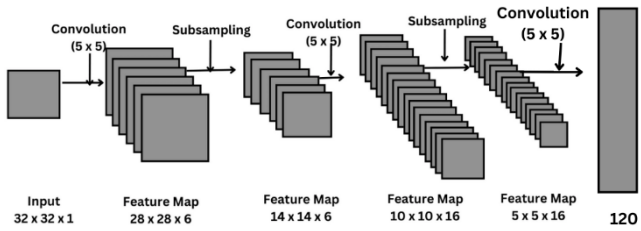

Figure 1. Architecture of the LeNet Convolutional Neural Network (CNN) [19]

Figure 2 presents the architecture of the CNN model used for CIFAR-10 classification.

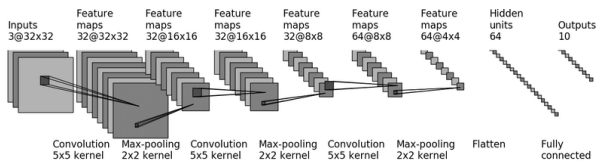

Figure 2. Architecture of the CIFAR-10 CNN [17]

LeNet and the CIFAR-10 CNN were chosen as examples of two different scales of computational complexity within the constraints of an embedded system. LeNet can be used as a lightweight baseline to validate fixed-point math, hardware function, and low-latency inference, while the CIFAR-10 CNN brings in a certain level of complexity in terms of computational requirements and memory, which can be used to study the scalability and efficiency of the same FPGA board. While a more realistic example would be the use of a lightweight CNN such as MobileNetV2 or EfficientNet-Lite, their integration in Zynq-7000-class FPGAs would necessitate extreme quantization and would be thus marked as future work.

3.2 Hardware platforms

The FPGA-based experimental platform is built on Zynq-7000 SoC technology, using both the PYNQ-Z1 and ZedBoard development boards. These boards are based on the XC7Z020 device, which integrates a dual-core ARM Cortex-A9 processing system (PS) with Artix-7 FPGA PL. The embedded system runs the PYNQ Linux operating system, which provides a high-level programming environment and enables seamless interaction between Python applications executed on the ARM processor and custom hardware accelerators deployed in the PL.

The hardware architecture was designed using Vivado HLS 2017.4, allowing the generation of optimized hardware accelerator IP cores from high-level C/C++ descriptions. These IP cores were then integrated into the Zynq SoC design to accelerate the computationally intensive components of the CNNs.

For performance comparison, a GPU-based reference platform was also employed. This platform consists of an ASUS GeForce GTX 1060 graphics card with 6 GB of VRAM. CNN inference for the Mask R-CNN model was implemented using the TensorFlow and Keras frameworks, with CUDA 10.0 and cuDNN 7.6 providing GPU acceleration.

Figure 3 shows the overall hardware architecture of the proposed PYNQ-based acceleration framework.

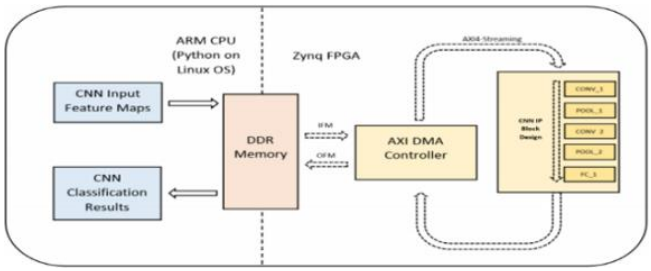

Figure 3. Overall hardware architecture of the PYNQ framework

3.3 Hardware–Software mapping strategy

The proposed system uses a heterogeneous execution architecture with co-design at the hardware-software interface. The most computation-intensive layers in a CNN, such as convolutional, pooling, and fully connected layers, will be designed using hardware accelerators in the FPGA PL. The motivation here is to use spatial parallelism, pipelining, and fixed-point arithmetic.

The ARM processing system (PS) handles overall control, memory, and data management, which involves pre-processing, accelerator invocation, and result retrieval in an image. The approach adopted ensures that control overhead is minimal and acceleration efficiency is maximized, since convolution layers are major contributors to both time and energy in CNN inference.

3.4 Experimental methodology

On the FPGA platform, the acceleration strategy consisted of mapping the convolutional, pooling, and fully connected layers of the CNNs onto the PL as dedicated hardware IP cores. Data transfers between external DDR memory and these hardware accelerators were managed through AXI4-Stream Direct Memory Access (DMA) channels, ensuring high-throughput communication while minimizing processor overhead. The ARM processing system (PS) coordinated the overall execution flow by dispatching input images to the hardware accelerators and collecting the final inference results.

On the GPU platform, the Mask R-CNN model was deployed using the Keras and TensorFlow frameworks, with NVIDIA CUDA and cuDNN libraries providing optimized hardware acceleration. The evaluation focused on inference tasks performed on images from the MS COCO dataset, with particular attention given to measuring execution latency and power consumption under realistic workload conditions.

Figure 4 depicts the data flow between the ARM processing system and the FPGA PL.

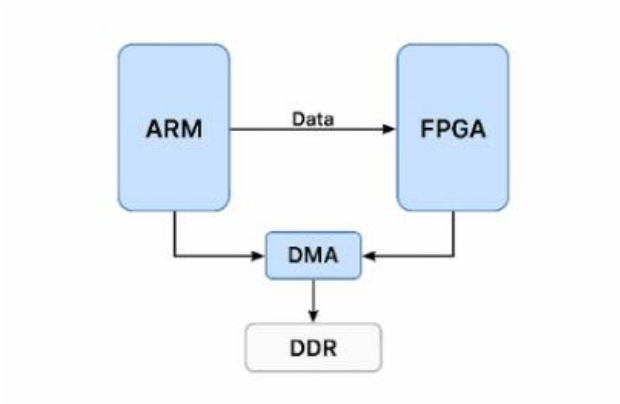

Figure 4. Data flow diagram (ARM ↔ FPGA)

3.5 Experimental setup and reproducibility details

For CNN inference on the FPGA platform, fixed-point operations are used in order to conserve hardware resources with minimal loss in classification accuracy. The fixed-point operation uses 16 bits for both weight and activation representations. However, fixed-point operations are not used in the ARM CPU implementation, which makes use of single-precision floating-point operations. The hardware accelerators are synthesized using Vivado HLS and directed by directives. Pipelining is used in the major loops of convolution in order to provide a low initiation interval and, in addition, partial loop unrolling is used within available DSP limits to enhance parallelism. Array partitioning is also used in local memory for feature maps and weights to support concurrent memory accesses.

The transfer of data between the ARM processing system (PS) and the PL has been accomplished using the AXI4-Stream DMA, thus facilitating the streaming of the input images and the intermediate feature maps between the external DDR memories and the hardware accelerators. The results obtained in the experiments suggest that the computation carried out on the FPGA has a dominant effect on the overall latency involved in the inference, whereas the communication overhead from the ARM to the FPGA using DMA is substantial. For the GPU platform, the inference tasks have been carried out using TensorFlow/Keras with the support of CUDA 10.0 and cuDNN 7.6, where the size of the mini-batch has been set to one to model the actual real-time scenario.

3.6 Methods of measurement and analysis

The evaluation of the implemented CNN models was

conducted using a set of quantitative metrics designed to ensure a fair and consistent comparison between the Zynq-7000 FPGA platform and the GPU-based solution. Execution time was measured over a batch of 600 test images and includes all data transfers between the ARM processing system (PS), external DDR memory, and the PL via the AXI4-Stream DMA interface. To improve measurement reliability, each experiment was repeated three times, and the average latency was reported.

Inference throughput, expressed in frames per second (FPS), was derived from the recorded execution time to assess the suitability of each platform for real-time processing. FPGA resource utilization including lookup tables (LUTs), digital signal processing blocks (DSPs), block RAMs (BRAMs), and flip-flops were obtained from Vivado post-implementation reports, enabling an analysis of the impact of increasing network complexity on hardware resource saturation.

Power consumption was estimated using the Vivado Power Analyzer for the Zynq-7000 platform and GPU Tweak II for the NVIDIA GPU. In addition, classification accuracy was evaluated on the MNIST and CIFAR-10 test datasets to verify that hardware-level optimizations and quantization did not introduce significant degradation in inference performance.

Overall, this evaluation framework provides a comprehensive and reliable basis for comparing performance, hardware resource usage, and energy efficiency across heterogeneous computing platforms.

Table 1 summarizes the performance metrics and measurement criteria used in this study.

This evaluation methodology ensures a rigorous and platform-agnostic comparison, explicitly considering the architectural differences between heterogeneous ARM–FPGA systems and GPU-based accelerators.

All experiments were conducted under identical conditions, and reported values correspond to the average of three runs to reduce measurement variability.

3.7 Toolchain selection and scope of the study

For experimentation, a tool flow has been chosen that consists of Vivado HLS 2017.4, combined with PYNQ. The experimental tool flow has been chosen for its compatibility with older hardware, as it is stable for use on Zynq-7000 XC7Z020, which is still supported by current literature. More recent software frameworks are Vitis AI, FINN, hls4ml, and TVM, which all support newer architectures.

In this case, the research aims to investigate system-level behavior without focusing on the compiler level. The need to utilize a full-fledged toolchain is imperative in this context in order to systematically exclude framework-dependent factors and focus on architectural-related effects.

This section provides a coherent and reproducible description of the experimental setup, execution flow, and evaluation metrics used throughout this study.

Table 1. Summary of performance and measurement criteria

Criterion	Description	Tool / Method
Latency (s)	Execution time for 600 test images	Python high-resolution timer
Throughput (FPS)	Number of images processed per second	Derived from total execution time
FPGA Resource Usage	LUTs, DSPs, BRAMs, Flip-Flops utilization	Vivado implementation report
Power Consumption (W)	Average during inference	Vivado Power Analyzer / GPU Tweak II
Classification Accuracy (%)	Correct predictions on test datasets	MNIST / CIFAR-10 test sets

4. RESULTS AND DISCUSSIONS

4.1 Execution times and accuracy

In this section of the paper, the results obtained are presented in Table 2 in order to highlight the information that will enable us to discuss them later.

Table 2. Table showing the results obtained from the two CNNs, LeNet and CIFAR

	FPGA: Deployment Phase CIFAR10	FPGA: Rapid Test Function Deployment	ARM CPU: Deployment Phase
Execution time	CIFAR: 4.0067727389999845 s LeNet: 2.250272926999969 s	CIFAR: 3.8113785109999867 s LeNet: 1.1258434439999974 s	CIFAR: 191.132702477 s LeNet: \
CPU execution time	CIFAR: User 9.58 s, sys: 350 ms, total: 9.93 s LeNet: User 3.51 s, sys: 2.62 s, total: 6.13 s	CIFAR: User 4.15 s, sys:180 ms, total: 4.33 LeNet: User 1.18 s, sys:70 ms, total:1.25 s	CIFAR: \ LeNet: User 22 s, sys:13.6 s, total:17.8 s
Real time taken by the CPU (wall time)	CIFAR : 10.1 s LeNet: 3.08 s	CIFAR: 4.53 s LeNet: 1.25 s	CIFAR : \ LeNet : 17.8 s
(Accuracy)	CIFAR: 0.74 (74%) LeNet: 0.983333333333(98.3%)	CIFAR: 0.74 (74%) LeNet: 0.98333 (98.33%)	CIFAR: 0.75166667 (75.2%) LeNet : 0.985 (98.5%)

In Table 2, “CPU execution time” refers to the sum of user and system processing times reported by the operating system, which reflect active CPU computation. In contrast, “real time taken by the CPU” corresponds to wall-clock time and includes additional delays due to memory access latency, data movement, and operating system scheduling. The observed

gap between these metrics highlights the impact of memory and I/O overhead on CPU-only CNN inference.

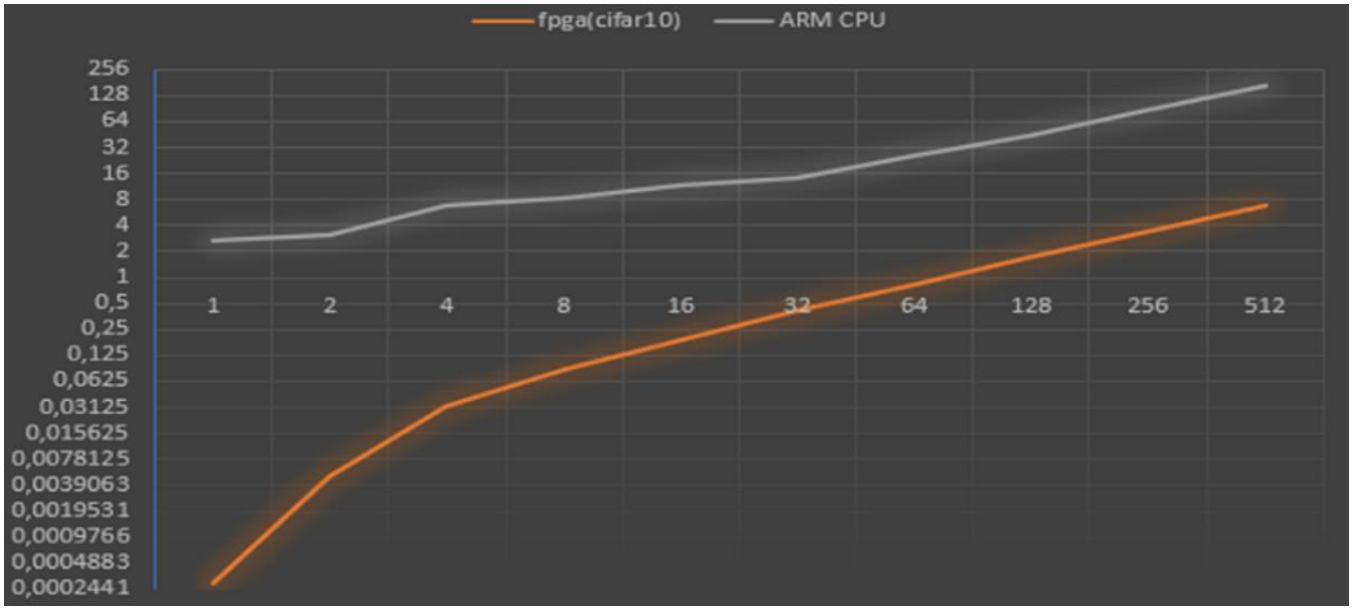

Figure 5. Graph showing deployment latency on FPGA and ARM CPU based on input image size

The analysis of the results highlights two key observations. First, the significantly lower execution time of LeNet compared to the CIFAR-10 network can be directly attributed to its simpler architectural design. LeNet comprises only two convolutional and pooling layers, whereas the CIFAR-10 model employs three convolutional stages, leading to a higher computational workload. This outcome confirms the strong correlation between network depth, architectural complexity, and overall execution time.

Second, the hardware deployment results clearly demonstrate the substantial performance advantage of the FPGA over the ARM CPU. In particular, the CIFAR-10 network achieves an acceleration factor of approximately 47× on the FPGA, with execution time reduced from 191 seconds on the CPU to only 4 seconds on the FPGA. This performance gap highlights the fundamental benefit of hardware-level parallelism: the FPGA is capable of concurrently processing multiple data paths and operations, whereas the CPU relies on largely sequential execution, resulting in significantly higher latency.

Figure 5 compares the deployment latency on the FPGA and ARM CPU as a function of the input image size.

4.2 Hardware acceleration on FPGA

One of the most significant outcomes of this study is the substantial performance gain achieved through hardware acceleration on the FPGA platform. On the PYNQ-Z1 board, execution of the CIFAR-10 CNN on the ARM processing system (PS) alone required 191.1 seconds to process 600 images. By offloading the computation to the FPGA PL, the execution time was reduced to only 4.0 seconds, corresponding to a speedup of approximately 47.7×.

The fact that there are performance improvements validates that accelerating convolutional and fully connected layers gives the greatest improvement because convolutional and fully connected layers are responsible for most computations and energy expenditure performed while doing inference in a CNN.

A similar trend is observed with the LeNet-5 network, where hardware acceleration reduced inference latency by several orders of magnitude, enabling near-real-time performance with a throughput of 266 FPS. These gains are primarily attributed to the fine-grained hardware parallelism enabled by the FPGA architecture, in which convolution and multiply–accumulate operations are executed concurrently across multiple DSP blocks within the PL.

4.3 Resource consumption analysis

The experimental results demonstrate that CNN model complexity has a direct and pronounced impact on FPGA resource utilization. The lightweight LeNet network occupies approximately 66% of the available LUTs, 22% of the digital signal processing (DSP) blocks, and fully saturates the block RAM (BRAM) resources, while maintaining a low average power consumption of 0.567 W.

In contrast, the deeper CIFAR-10 CNN exhibits substantially higher hardware demands, consuming 67% of LUTs, 87% of DSP resources, and likewise exhausting 100% of the available BRAMs, with a corresponding increase in power consumption to 0.862 W. Although logic utilization remains within acceptable limits, the complete saturation of BRAMs in both configurations highlights a key limitation of FPGA-based acceleration for deep neural networks.

These results indicate that on embedded FPGA platforms, on-chip memory capacity rather than computational logic often becomes the primary bottleneck when deploying deeper CNN architectures. Consequently, memory optimization strategies such as data reuse, buffering, compression, or network quantization are essential to further scale CNN implementations on resource-constrained FPGA devices.

As most FPGA acceleration research, ranging from classic computer architecture to modern machine learning, has predominantly explored the throughput aspect, the importance of system-level trade-offs in this chapter cannot be overstated.

Figures 6 and 7 present the FPGA resource utilization for the LeNet and CIFAR-10 CNNs, respectively.

Instance	Module	BRAM_18K	DSP48E	FF	LUT
AXI_DMA_MASTER_U0	AXI_DMA_MASTER	0	0	586	6917
AXI_DMA_SLAVE_U0	AXI_DMA_SLAVE	0	0	511	6696
FC_1u_500u_10u_U0	FC_1u_500u_10u_s	10	21	2152	5963
FC_1u_800u_500u_U0	FC_1u_800u_500u_s	200	25	2161	7191
SCIG_U0	SCIG	8	0	1047	7745
SCIG_1_U0	SCIG_1	8	0	804	5518
SMM_1u_25u_20u_U0	SMM_1u_25u_20u_s	25	10	1468	6014
SMM_1u_500u_50u_U0	SMM_1u_500u_50u_s	25	27	2668	7245
pool_2u_20u_24u_U0	pool_2u_20u_24u_s	0	6	1469	5304
pool_2u_50u_8u_U0	pool_2u_50u_8u_s	0	6	1485	5309
Total		276	95	14351	63902

Figure 6. Resource consumption – LeNet

Instance	Module	BRAM_18K	DSP48E	FF	LUT
AXI_DMA_MASTER_U0	AXI_DMA_MASTER	0	0	586	6917
AXI_DMA_SLAVE_U0	AXI_DMA_SLAVE	0	0	511	6696
FC_1u_1024u_64u_U0	FC_1u_1024u_64u_s	64	41	3963	7760
FC_1u_64u_10u_U0	FC_1u_64u_10u_s	16	25	2564	5971
SCIG_U0	SCIG	16	0	1099	6057
SCIG_1_U0	SCIG_1	16	0	910	8985
SCIG_2_U0	SCIG_2	16	0	859	9005
SMM_1u_75u_32u_U0	SMM_1u_75u_32u_s	25	34	3664	6893
SMM_1u_800u_32u_U0	SMM_1u_800u_32u_s	25	34	3282	6983
SMM_1u_800u_64u_U0	SMM_1u_800u_64u_s	50	34	3286	6986
pool_2u_32u_16u_U0	pool_2u_32u_16u_s	0	6	1435	5178
pool_2u_32u_32u_U0	pool_2u_32u_32u_s	0	6	1557	5684
pool_2u_64u_8u_U0	pool_2u_64u_8u_s	0	6	1469	5390
Total		228	186	25185	88505

Figure 7. Resource consumption – CIFAR-10

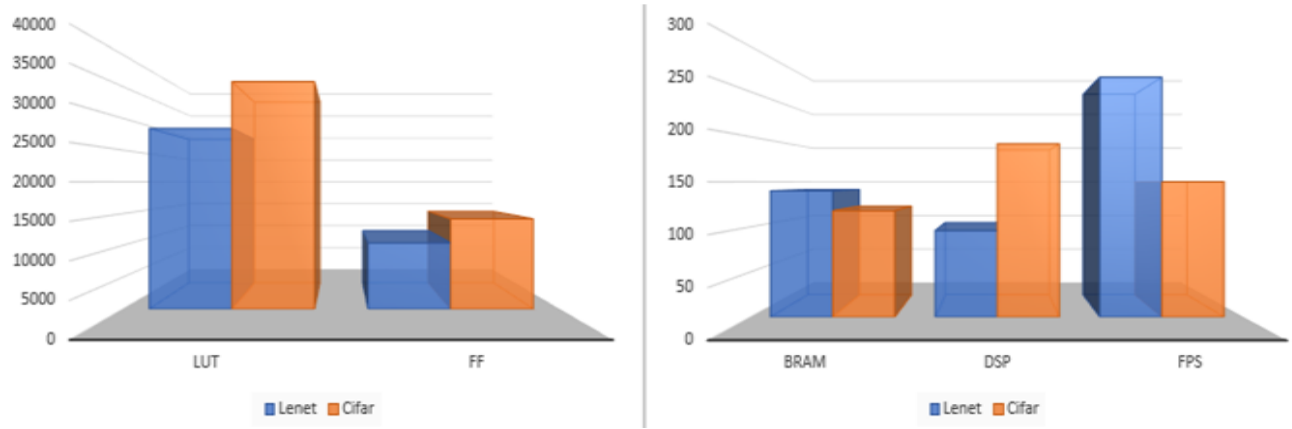

Figure 8. Histogram representation of the performance of the two CNNs LeNet and CIFAR10

Figure 8 highlights the impact of model complexity on FPGA resource consumption (LUT, FF, BRAM, DSP) and achievable frame rate for the LeNet and CIFAR-10 CNNs.

The analysis of the hardware implementation for the LeNet and CIFAR CNNs reveals an initial challenge related to resource overutilization. Early estimates produced by HLS indicated that both models significantly exceeded the available FPGA capacity, with projected look-up table (LUT) utilization reaching 120% for LeNet and 166% for the CIFAR network. These preliminary results underscored the difficulty of directly mapping CNN architectures onto resource-constrained embedded FPGAs without optimization.

Through iterative HLS optimization, this oversizing issue was effectively mitigated in the final implementations. For the LeNet model, LUT utilization was reduced to a feasible 66%, albeit at the expense of fully saturating the on-chip BRAM resources. Similarly, the optimized CIFAR network achieved a reduced LUT usage of 99%, while heavily exploiting DSP blocks, which reached 87% utilization. In both implementations, the consumption of other resources, such as flip-flops (FFs) and LUTRAMs, remained relatively low.

These results clearly demonstrate the effectiveness of HLS optimization techniques in reshaping computationally intensive CNN architectures to meet the hardware constraints of the target FPGA. By appropriately trading off logic, memory, and DSP usage, HLS enables the practical deployment of deep learning models on embedded FPGA platforms.

BRAM Saturation and Scalability Limitations

BRAM Saturation and BRAM Scaling:

The results of the post-implementations prove that the LeNet and CIFAR-10 CNNs achieve complete saturation of

the BRAM resources available on the Zynq-7000 device. This signifies that the major bottleneck for the deeper CNN architecture is the availability of the on-chip memories rather than the logic or DSP resources.

The result of the saturation of the BRAMs means that the storage of the intermediate feature maps and the weights becomes difficult, causing a latency overhead. The proposed solutions to the latency overhead caused by the saturation of the BRAMs include tile, streaming, compression, reduced precision, and layer fusion. The above techniques have been widely identified to be integral to the ability to support complex CNN configurations on FPGAs.

4.4 Accuracy assessment

The classification accuracy results obtained on the FPGA and ARM CPU platforms exhibit only marginal differences. For the CIFAR-10 network, the CPU-based implementation achieves an accuracy of 75.2%, while the FPGA-accelerated version attains 74.0%. Although fixed-point quantization slightly reduces numerical precision, the observed accuracy degradation remains below 2%, confirming the suitability of the chosen representation for embedded inference.

Importantly, this minor accuracy loss does not compromise the overall predictive capability or relevance of the model. These results confirm that CNN hardware acceleration on FPGA platforms can be effectively deployed while preserving classification reliability, making it well suited for performance- and energy-constrained embedded applications.

4.5 Comparison with GPU

On the NVIDIA GTX 1060 GPU platform, inference with the Mask R-CNN model required approximately 1,338 ms per image, corresponding to a throughput of 0.75 FPS. During execution, the GPU operated at near-maximum utilization (98–100%), with an average power consumption of 79.2 W and a memory footprint of approximately 5.5 GB.

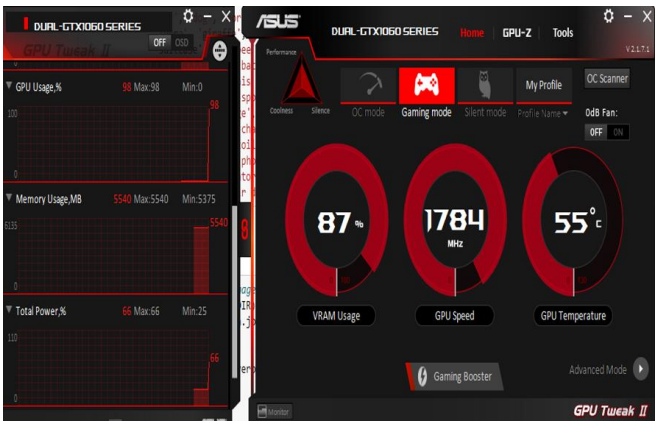

Figure 9. Resources used to process a single image with the GPU

These results clearly demonstrate the substantial computational capability of GPUs when executing large and complex neural networks. However, they also highlight the inherent trade-off between performance and energy efficiency, confirming that such GPU-based solutions are poorly suited for embedded or edge computing environments, where power consumption and memory capacity are critical constraints.

Figure 9 reports the hardware resource usage of the GPU during the processing of a single image.

The figure illustrates the results of the proposed implementation, including the hardware resource usage reported by GPU Tweak II. These measurements are subsequently summarized in a table to facilitate interpretation and analysis.

Figure 10. Result of processing a single image with the GPU

Figure 10 illustrates the result of GPU-based inference on a single input image, highlighting the high computational load induced by dense multi-object detection in complex urban scenes.

Table 3 compares the ideal and experimental GPU resource usage during inference.

The analysis of the results highlights a fundamental trade-off between detection performance and hardware cost. Although the use of optimized pre-trained weights enables high detection accuracy and relatively acceptable inference latency strongly dependent on GPU capabilities this level of performance comes at a substantial hardware expense. Even with cuDNN optimizations, memory usage remains extremely high, reaching approximately 5.54 GB of VRAM, which is prohibitive for most embedded systems.

Moreover, GPU utilization approaches its maximum when processing a single image, leading to sustained high operational stress and inefficient usage of available resources. Power consumption is likewise significant, averaging 79.2 W (approximately 66% utilization), which is incompatible with the energy-efficiency requirements of embedded and edge computing platforms. Finally, despite the considerable hardware demand, the achieved throughput of approximately one frame per second, while sufficient for basic detection, remains modest relative to the level of resource and energy consumption involved.

These observations reinforce the conclusion that, although GPUs are highly effective for complex deep learning models, their deployment for such workloads is ill-suited to energy-constrained embedded environments.

Table 3. Comparison of resources used in the ideal and experimental cases

	Execution Time	Memory Usage	Evaluation of GPU Usage	Total Power Used	Images Processed per Second	Detection Accuracy
Ideal case	200 ms	50%	Not mentioned	Not mentioned	5 fps	50% to 100%
Experimental case	≈ 1338 ms	87%	98% to 100%	≈ 66%	≈ 1 fps	71.3% to 100%

Table 4. Comparative analysis of hardware resources and characteristics across different platforms

Criterion	PYNQ-Z1 (Zynq-7000)	ZedBoard (Zynq-7000)	GPU Nvidia GTX 1060
Execution time	Slow to moderate depending on CNN complexity	Moderate (similar to PYNQ)	Very fast for deep networks
Resource utilization	Moderate (≈ 67% LUT, 87% DSP, 100% BRAM)	Slightly higher (≈ 70% LUT, 84% DSP)	Very high (VRAM ≈ 5–6 GB, GPU load ≈ 98%)
Power consumption	Low (< 1 W)	Moderate (~1.05 W)	Very high (~79 W)
Processing type	Hybrid parallel/sequential (PS + PL)	Hybrid parallel/sequential (PS + PL)	Fully parallel
Image processing rate (FPS)	150–266 FPS	142–250 FPS	0.75 FPS (Mask R-CNN)
Classification accuracy	Medium to high (≈ 74–75%)	Medium (≈ 73–74%)	Very high (≈ 90–95%)
Reconfigurability	Yes – FPGA dynamically reconfigurable	Yes – FPGA dynamically reconfigurable	No – fixed architecture
Training capability	No (inference only)	No (inference only)	Yes (training and inference)
Programming language	Python (via PYNQ) / C / HLS	C / HDL / Vivado HLS	Python / C / CUDA
Development tools	Vivado / Vivado HLS / PYNQ Framework	Vivado / SDK / Petalinux	TensorFlow / CUDA / cuDNN
Embedded integration	Very easy (PYNQ Linux + Jupyter)	Easy (hardware-oriented prototyping)	Difficult (size, power, cooling)

4.6 Extended performance and efficiency metrics

Energy Efficiency:

A metric that will be used to determine the energy efficiency of the tested platforms is the FPS/W metric. The FPGA-based LeNet model runs at approximately 470 FPS/W and the CIFAR-10 model at around 174 FPS/W. For the Mask R-CNN model, running on the GPU architecture, the FPS/W metric is only around 0.01 because of its high energy consumption. It is easy to see that the FPGA-based SoC performs much better.

Area Efficiency:

The area efficiency metric is compared based on the achieved throughput and the utilization of the FPGA resources. The designs achieve efficient performance density based on the usage of the existing LUT/DSP resources, thus validating the use of spatial parallelism for enhanced PL performance.

End-to-End Latency:

The total latency of inference can be broken down into the computation time of the data in the PL and the data transfer time from the ARM processing system to the FPGA. Experiential results suggest that the computation time is larger than the data transfer overhead.

Memory Bandwidth and Batch Size Concerns:

Because of the limited BRAM on the Zynq-7000 SoC, the definition of memory organization and buffering has a big effect on the scalability solution. Experiments for the real-time embedded inference problem were performed with a batch size of one. Though higher batch sizes can speed up the GPU execution speed, they are less relevant for theedge system application and thus not taken into consideration.

4.7 Comparison of the platforms used

A comparative analysis of the evaluated platforms reveals two distinct but complementary trade-offs. First, the comparison between the NVIDIA GTX 1060 GPU and the PYNQ-Z1 FPGA platform highlights fundamentally different design philosophies. The GPU delivers the raw computational throughput (TFLOPS) required for large-scale deep neural networks, such as ResNet and Mask R-CNN. However, its high power consumption, reaching 79.2 W, effectively rules it out for embedded and edge computing applications.

In contrast, the PYNQ-Z1 excels in energy efficiency, consuming less than 1 W while delivering an exceptional performance-per-watt ratio (FPS/W) for lightweight CNN models, with inference throughput ranging from 150 to 266 FPS. This makes it particularly well suited for embedded AI (Edge AI) deployments. Furthermore, the PYNQ framework simplifies development through hybrid hardware–software programming using Python, reducing implementation complexity and development time.

The second comparison focuses on the ZedBoard and the PYNQ-Z1, both of which are based on the same Zynq XC7Z020 SoC. While the ZedBoard offers a more robust and extensible hardware platform better suited for industrial prototyping and system integration, this comes at the cost of marginally higher power consumption. The PYNQ-Z1, by contrast, is optimized for rapid prototyping and educational use, offering superior energy efficiency while maintaining nearly identical computational performance, with less than a 5% difference in inference latency.

Table 4 provides a comparative analysis of the evaluated hardware platforms.

This comparison underscores the fundamentally different design objectives of the three evaluated platforms. The PYNQ-Z1 achieves an effective balance among energy efficiency, programmability, and real-time performance, making it particularly well suited for embedded vision and edge AI applications. The ZedBoard, while based on the same SoC architecture, offers enhanced connectivity and greater scalability, rendering it more appropriate for research-oriented or industrial prototyping, albeit at the expense of modestly higher power consumption. In contrast, the NVIDIA GTX 1060 GPU remains unmatched in terms of raw computational throughput and training capability; however, its substantial

energy consumption and hardware cost significantly constrain its applicability in embedded and energy-constrained environments.

5. PRACTICAL DEPLOYMENT CONSIDERATIONS FOR EDGE AI SYSTEMS

Although this work focuses on the evaluation at the development-board level, deploying CNN accelerations in real embedded systems introduces additional challenges that must be tackled.

Sensor Interfaces:

In the case of practical edge AI systems, image and signal data usually have to be directly acquired from sensors such as cameras, LiDAR, or industrial sensors. In a Zynq-based platform, the interface for sensors can be implemented by standard protocols such as MIPI CSI, SPI, and I2C and be directly connected with the PL.

RTOS & System Integration:

For real-time applications, the ARM processing system can run an RTOS or a real-time configuration of Linux. Hardware accelerators in the PL can be exposed as either memory-mapped peripherals or DMA-driven accelerators, allowing deterministic task scheduling and bounded latency.

Power and Thermal Management:

Amongst the key advantages of FPGA-based SoCs over GPU-based ones is their low power consumption, which simplifies thermal design. In practice, techniques like DVFS, clock gating, and workload-aware scheduling can further reduce the power consumption. Model Update and Maintainability: Updating CNN models is necessary in deployed systems. Model update and accelerator adaptation are achievable in FPGA-based SoCs by means of partial or full reconfiguration without replacement of the entire system. Software-level updates are supported on the ARM processor. These considerations show that the proposed FPGA-based acceleration approach is particularly suitable for consideration in complete embedded and edge AI systems beyond development-board prototypes.

6. CONCLUSIONS

The primary objective of this work was to study, implement, and compare CNNs across heterogeneous hardware platforms, namely Zynq-7000 FPGA-based systems (PYNQ-Z1 and ZedBoard) and an NVIDIA GTX 1060 GPU. The study focused on evaluating the trade-offs between computational performance, energy consumption, and hardware resource utilization in order to identify the most suitable platform for deploying computer vision applications in embedded environments.

A comprehensive literature review provided the theoretical foundation in artificial intelligence, machine learning, and deep learning. This background was complemented by a practical implementation phase using a wide range of hardware and software tools, including FPGA development boards, a GPU platform, Vivado HLS, Jupyter Notebook, Anaconda, and deep learning frameworks such as Caffe, Lasagne, and CUDA with Python. These tools enabled the implementation of three CNN architectures on FPGA platforms and a complex detection network on GPU, forming the basis for a rigorous and objective comparative evaluation.

Experimental results clearly demonstrate the advantages of FPGA-based hardware acceleration for embedded vision workloads. The LeNet and CIFAR-10 models deployed on the PYNQ-Z1 achieved acceleration factors of up to 47× compared to execution on the embedded ARM processor, while maintaining power consumption below 1 W. The ZedBoard, although offering greater robustness and extensibility, exhibited slightly reduced performance due to its more generic hardware configuration. In contrast, the GTX 1060 GPU demonstrated excellent performance and accuracy for complex models such as Mask R-CNN, but at the cost of very high-power consumption (approximately 79 W), making it unsuitable for energy-constrained embedded systems.

These findings highlight a key conclusion: FPGA-based hardware parallelism provides an optimal compromise between performance, energy efficiency, and architectural flexibility. As a result, Zynq-7000 SoCs emerge as a highly effective solution for deploying CNNs in embedded and edge AI applications such as smart cameras, robotics, drones, and autonomous systems.

Several directions for future work could extend and strengthen this research. These include integrating real-time image acquisition to evaluate CNN inference under practical conditions, deploying more advanced models (e.g., YOLO or Mask R-CNN) on FPGA through deeper optimization of quantization and dataflow, and developing automated tools for CNN accelerator generation to simplify FPGA integration. Additionally, exploring heterogeneous multi-accelerator systems combining CPUs, GPUs, FPGAs, and NPUs could further optimize the balance between accuracy, latency, and power consumption.

In conclusion, this work provides a significant experimental contribution to the study of CNN hardware deployment on embedded platforms. It demonstrates the strong potential of Zynq-7000 SoCs in enabling efficient, low-power, and flexible embedded AI solutions, and establishes a solid foundation for future research in distributed and energy-efficient intelligent computing.

The analysis reveals that memory capacity, particularly BRAM availability, represents the dominant bottleneck for scaling CNN inference on embedded FPGA platforms, emphasizing the need for memory-aware accelerator design.

Future work will target further research leveraging contemporary FPGA deep learning compilers and development frameworks, such as Vitis AI and FINN, to assess their effectiveness on energy efficiency and scalability for existing and future FPGA architectures.

REFERENCES

[1] Tan, M., Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning, pp. 6105-6114.

[2] Canziani, A., Paszke, A., Culurciello, E. (2016). An analysis of deep neural network models for practical applications. arXiv preprint arXiv:1605.07678. https://doi.org/10.48550/arXiv.1605.07678

[3] Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S. (2020). Efficient Processing of Deep Neural Networks. San Rafael: Morgan & Claypool Publishers. https://doi.org/10.1007/978-3-031-01766-7

[4] Carballo-Hernández, W., Pelcat, M., Berry, F. (2021).

Why is FPGA-GPU heterogeneity the best option for embedded deep neural networks? arXiv preprint arXiv:2102.01343. https://doi.org/10.48550/arXiv.2102.01343

[5] Mittal, S. (2019). A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. Journal of Systems Architecture, 97: 428-442. https://doi.org/10.1016/j.sysarc.2019.01.011

[6] AMD Technical Information Portal. https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM.

[7] Bjerge, K., Schougaard, J.H., Larsen, D.E. (2021). A scalable and efficient convolutional neural network accelerator using HLS for a system-on-chip design. Microprocessors and Microsystems, 87: 4363. https://doi.org/10.1016/j.micpro.2021.104363

[8] Zhu, C., Huang, K., Yang, S., Zhu, Z., Zhang, H., Shen, H. (2020). An efficient hardware accelerator for structured sparse convolutional neural networks on FPGAs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(9): 1953-1965. https://doi.org/10.1109/TVLSI.2020.3002779

[9] Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P., Jahre, M., Vissers, K. (2017). Finn: A framework for fast, scalable binarized neural network inference. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 65-74. https://doi.org/10.1145/3020078.3021744

[10] Guo, K., Sui, L., Qiu, J., Yu, J., Wang, J., Yao, S., Han, S., Wang, Y., Yang, H. (2017). Angel-eye: A complete design flow for mapping CNN onto embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(1): 35-47. https://doi.org/10.1109/TCAD.2017.2705069

[11] Mittal, S. (2020). A survey of FPGA-based accelerators for convolutional neural networks. Neural Computing and Applications, 32(4): 1109-1139. https://doi.org/10.1007/s00521-018-3761-1

[12] Vitis AI User Guides / DPU Product Guides — VitisTM AI 3.0 documentation. https://xilinx.github.io/Vitis-AI/3.0/html/docs/reference/release_documentation.html.

[13] Liu, B., Zhou, Y., Feng, L., Fu, H., Fu, P. (2022). Hybrid CNN-SVM inference accelerator on FPGA using HLS. Electronics, 11(14): 2208. https://doi.org/10.3390/electronics11142208

[14] Maciel, L.A., Souza, M.A., Freitas, H.C. (2023). Energy-efficient CPU+ FPGA-based CNN architecture for intrusion detection systems. IEEE Consumer Electronics Magazine, 13(4): 65-72. https://doi.org/10.1109/MCE.2023.3283730

[15] Dhilleswararao, P., Boppu, S., Manikandan, M.S., Cenkeramaddi, L.R. (2022). Efficient hardware architectures for accelerating deep neural networks: Survey. IEEE Access, 10: 131788-131828. https://doi.org/10.1109/ACCESS.2022.3229767

[16] Guedria, S. (2020). A scalable and component-based deep learning parallelism platform: An application to convolutional neural networks for medical imaging segmentation. Doctoral dissertation, Université Grenoble Alpes.

[17] Zhou, Y., Song, S., Cheung, N.M. (2017). On classification of distorted images with deep convolutional neural networks. arXiv preprint arXiv:1701.01924. https://doi.org/10.48550/arXiv.1701.01924

[18] He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017). Mask R-CNN. In 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, pp. 2980-2988. https://doi.org/10.1109/ICCV.2017.322

[19] Code 360 by Coding Ninjas. 2024 Naukri.com. https://www.naukri.com/code360/library/lenet-5.