CNN Acceleration on PYNQ-Z2 Using HLS and AXI DMA for Edge AI Applications

A Hardware-Software Co-Design Approach for Low-Latency Image Processing on Zynq 7000 SoC

1st Kushaan Krishna Bhat
Department of Electronics & Telecommunication Engineering
K J Somaiya Institute of Technology
Mumbai, India
kushaan.bhat@somaiya.edu

2nd Vidhan Kishor Bhasme
Department of Electronics & Telecommunication Engineering
K J Somaiya Institute of Technology
Mumbai, India
vidhan.b@somaiya.edu

3rd Madhav Suresh Asawa
Department of Electronics & Telecommunication Engineering
K J Somaiya Institute of Technology
Mumbai, India
madhav.asawa@somaiya.edu

4th Aman Radheykrishna Bharadwaj
Department of Electronics & Telecommunication Engineering
K J Somaiya Institute of Technology
Mumbai, India
aman.bharadwaj@somaiya.edu

5th Prof. Sagar Vijay Mhatre
Department of Electronics & Telecommunication Engineering
K J Somaiya Institute of Technology
Mumbai, India
sagar.mhatre@somaiya.edu

***Abstract*—This project explores how real-time object detection can be made faster and more efficient by using a custom hardware accelerator based on a Convolutional Neural Network (CNN). Built on the PYNQ-Z2 SoC platform, the system focuses on recognizing traffic signs quickly and accurately. We trained a compact CNN on the GTSRB dataset and optimized it for hardware by converting it to a 16-bit fixed-point format. Using Vitis HLS, we turned the model into a hardware IP core and integrated it with AXI interfaces for smooth communication between the processor and FPGA logic. By offloading heavy computation to the FPGA, the system runs faster and uses less power than traditional CPU-based approaches. This design shows how deep learning can be efficiently deployed on reconfigurable hardware for smart, real-time applications like autonomous vehicles and smart surveillance.**

Keywords—CNN Accelerator, FPGA, PYNQ-Z2, Vivado HLS, AXI DMA, Hardware-Software Co-design, Convolutional Neural Network, Zynq-7000 SoC, Real-time Image Processing, High-Level Synthesis.

I. INTRODUCTION

As smart systems become more integrated into daily life from autonomous vehicles to intelligent surveillance, there's growing demand for real-time, efficient image processing. Convolutional Neural Networks (CNNs) power many of these systems but often struggle with speed and energy use on traditional processors, especially in Edge AI Applications. [4]

To address this, a custom CNN accelerator was developed on the PYNQ-Z2 platform, leveraging its Zynq-7000 SoC, which combines ARM processing with FPGA fabric. A lightweight CNN trained on the GTSRB (German Traffic Signal Recognition Benchmark) dataset was optimized to a 16-bit fixed-point format, then implemented in hardware using Vitis HLS.

The proposed design uses AXI DMA and AXI-Lite buses to offload CNN inference to the programmable logic, enabling faster, energy-efficient classification. This hardware-software co-design approach offers a practical and scalable solution for deploying deep learning at the edge device balancing performance, power efficiency, and real-time capability in smart embedded systems.

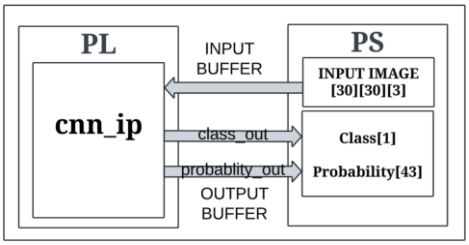

Fig. 1. Proposed Hardware Architecture Block Diagram

II. BACKGROUND

Real-time object detection is key to technologies like autonomous driving and smart surveillance, where quick and accurate decisions matter. CNNs are widely used for these tasks but often require more power and processing than edge devices can handle.[2] FPGAs offer a flexible, energy-efficient solution by handling computations in parallel. The PYNQ-Z2, combining an ARM processor with FPGA fabric, provides an ideal platform to accelerate CNNs, making real-time, on-device AI both practical and efficient. [5]

III. RELATED WORKS

Recent developments in FPGA-based acceleration of deep learning models have showcased various strategies to balance inference performance, flexibility, and power efficiency — particularly for edge AI applications.

LSTM accelerators such as the one proposed in Enhancing Edge Performance [1] demonstrate how high-level synthesis on PYNQ-Z2 can yield a 3.7× speedup over ARM-only execution, proving the viability of sequence models for edge devices. In contrast, our work focuses on CNN-based spatial inference but shares the same design philosophy of efficient hardware-software co-design.

In The Implementation of a Power Efficient BCNN-based Object Detection Acceleration [2], a binarized YOLOv2 model is deployed on an FPGA-SoC and achieves 15 FPS at just 1.45 W. While this highlights extreme power savings using binarized operations, our approach preserves higher accuracy by using fixed-point precision and optimized data movement.

Work in FPGA-based Deep Learning Models for Object Detection and Recognition [3] explores lightweight CNNs on the ZedBoard using the PYNQ framework, showing feasibility with a 100 ms inference time on CIFAR-10. Unlike their software-focused deployment, we develop a fully custom CNN accelerator using Vitis HLS and integrate it tightly via AXI-DMA for real-time performance.

Comparative studies such as SSD, YOLO, and Faster R-CNN on PYNQ-Z2 and Movidius NCS [8] confirm that FPGAs often outperform CPUs in both speed and accuracy for edge inference. However, most rely on existing IPs or toolchains. Our design stands apart by offering a fully custom pipeline tailored for a specific task—traffic sign recognition—optimized at both algorithmic and hardware levels.

Efficient Implementation of FPGA-based Object Detection Using Multi-Scale Attention [9] introduces spatial attention and quantization to improve SSD latency and accuracy. While their design targets more complex models, we focus on lightweight CNNs and demonstrate that even modest architectures can be competitive when properly optimized through HLS and dataflow techniques.

On the same note, Implementation of CNNs on FPGA to detect Objects [10] and Design and Implementation of CNN Accelerator using FPGA [6] exploit pipelining and buffering to increase performance. This is the thinking behind our accelerator which utilizes loop optimizations, AXI streaming, to maximize the throughput of the Zynq 7000 SoC.

A study entitled FPGA High-Performance Implementation Method of CNN [7] explains the utilization of trained LeNet-5 models in PyTorch that are converted to fixed-point quantization using HLS. We go beyond this work to consider more realistic, real-world data (GTSRB) and demonstrate accuracy verification post-hardware deployment and provide more than 11x speedup of CPU inference.

The OCR and the BNN-based classifiers are quantized to accessorize the FPGA-Based Hardware Acceleration Using PYNQ-Z2 [11]. Our project has the same approach to quantization, but it is done to CNN, which was trained on GTSRB, to ensure greater precision with a low latency inference on actual traffic sign data.

Finally, the last study Benchmarks pre-trained models on Zynq-based FPGA, Real-time Object Detection [10] can implement CNN on the Zynq-based FPGA platform using the PYNQ method, including SSD-Inception v2, and exhibits the accuracy-latency trade-off. We lay complementary ground by demonstrating how a convolution network fully programmable in logic may satisfy real-time requirements with a limited power and resource consumption, when designed specifically to offload convolution operations.

In combination, the works highlight the increasing presence of the FPGA-based acceleration in edge AI. These pillars are constructed in our design through the integration of model-based quantization, HLS and efficient data processing through AXI-DMA and provide a scalable power-efficient and compact CNN accelerator for real-time systems.

IV. Methodology

This project followed a structured hardware-software co-design approach to build a CNN-based accelerator. The aim was to develop an efficient system on the PYNQ-Z2 board capable of classifying traffic signs in real time by combining machine learning techniques with FPGA-based hardware implementation.

1. Dataset and CNN Model Training

A lightweight CNN was trained to recognize traffic signs using the GTSRB dataset, which contains over 50,000 images across 43 categories. The model was developed with Keras and TensorFlow, utilizing standard layers such as convolution, pooling, ReLU activation, and dense layers.

To enable efficient deployment on embedded hardware, the trained floating-point weights and activations were converted to a 16-bit fixed-point format. This quantization reduced memory usage and computational complexity while maintaining strong performance.

The model achieved a training accuracy of 98.65% and a validation accuracy of 95.63%, demonstrating effective recognition capabilities.

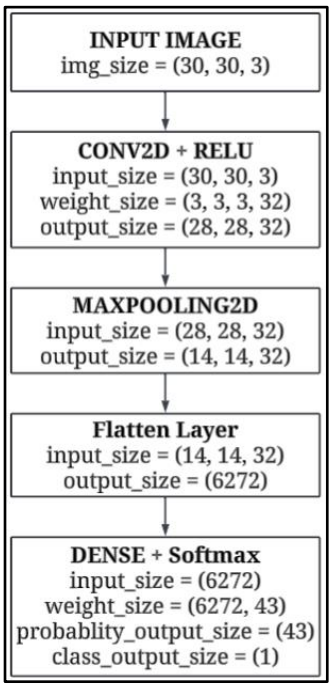

Fig. 4.1.1 CNN Model Architecture

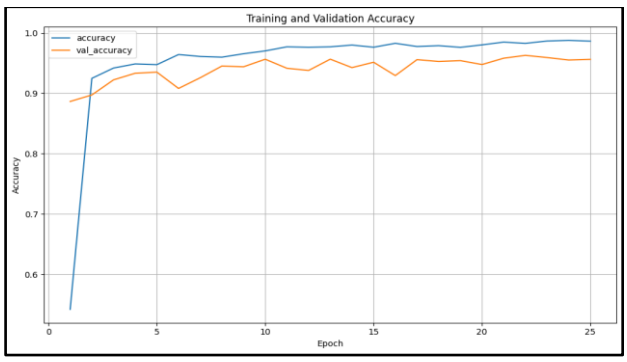

Fig. 4.1.2 Epoch vs Training & Validation Accuracy

2. *Hardware Accelerator Design Using HLS*

The next phase involved implementing the CNN on an FPGA using Vitis HLS. The CNN functions were described in HLS and then converted into a custom hardware IP core. To boost performance, techniques like loop pipelining, unrolling, and dataflow were applied, increasing parallelism and cutting down latency.

This specialized CNN core was designed to handle one image at a time, delivering classification results quickly and efficiently while using minimal hardware resources. The accelerator runs entirely on the Programmable Logic section of the Zynq 7000 SoC, maximizing throughput.

Utilization Estimates

Summary

Name	BRAM_18K	DSP	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	4	-
FIFO	-	-	-	-	-
Instance	172	77	17864	28977	-
Memory	73	-	64	22	0
Multiplexer	-	-	-	1463	-
Register	-	-	81	-	-
Total	245	77	18009	30466	0
Available	280	220	106400	53200	0
Utilization (%)	87	35	16	57	0

Fig. 4.2.1 PYNQ Post HLS-Synthesis Utilization Estimation

3. *Integration with PYNQ-Z2 via AXI Interfaces*

The CNN accelerator was connected to the system using AXI-Lite for control commands and AXI-Stream with DMA for fast data transfer between the ARM processor and FPGA. This setup ensured smooth control and quick movement of images and results.

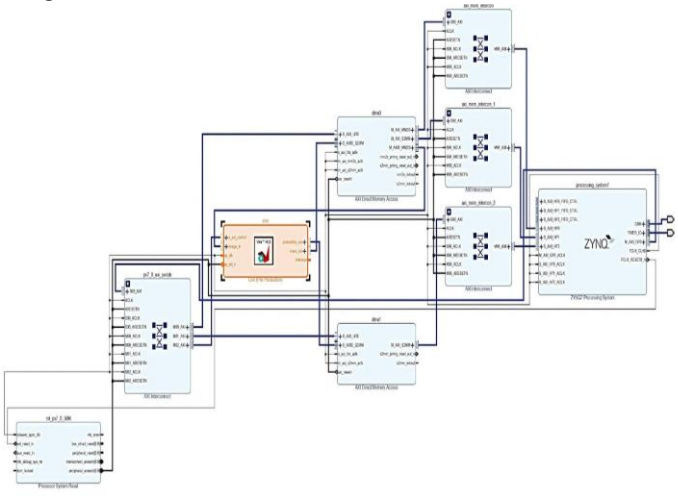

Fig. 4.3.1 System Design of CNN Accelerator Integration on PYNQ-Z2 Using AXI DMA

This communication setup creates a seamless and fast connection, enabling real-time inference without slowing down the processor.

4. *Software Control and Testing*

The PYNQ-Z2 board runs a Jupyter Notebook server for remote Python control of both processor and FPGA. A notebook was created to preprocess images, send them to the CNN accelerator, and display real-time results.

V. RESULTS & DISCUSSION

Performance on the PYNQ-Z2 was evaluated by measuring power efficiency, speed, resource utilizations, and accuracy to ensure the CNN accelerator meets real-world edge AI demands.

1. *Power Analysis*

Power consumption was measured to assess efficiency. The ARM Cortex-A9 processor PS used around 1.53 W, while the CNN accelerator running on the FPGA PL, including all logic and resources, consumed just 0.35 W, highlighting its low power usage. These values were obtained using the on-chip power analysis feature in Vivado, which reports power based on the implemented design and estimated switching activity.

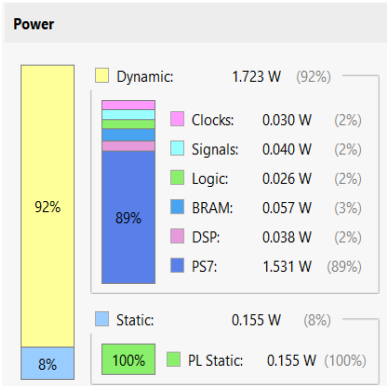

Fig. 5.1.1 On-Chip Power Consumption Report of CNN Accelerator

2. *Performance Analysis*

Inference time was benchmarked against a standard CPU (Intel Core i5-7200U @ 2.5 GHz) to gauge real-world speedup:

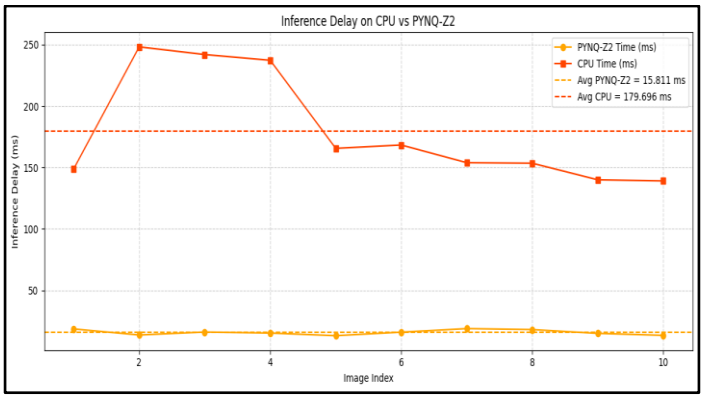

Fig. 5.2.1 Inference Delay CPU vs PYNQ-Z2

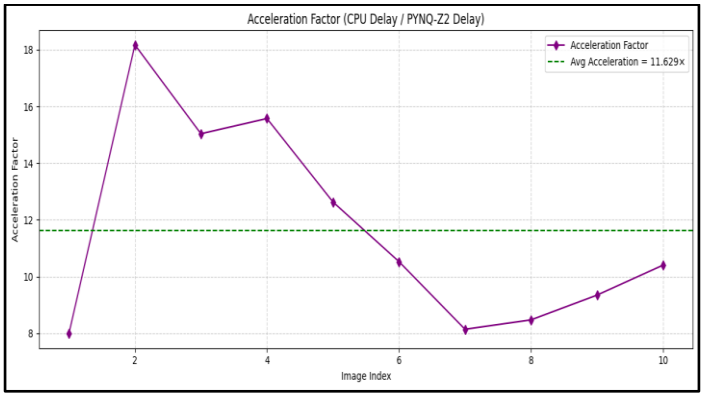

Fig. 5.2.2 Acceleration Factor (CPU Delay / PYNQ-Z2 Delay)

Platform	*Avg. Inference Delay*
Intel Core i5-7200U	179.6963 ms
PYNQ-Z2	15.8112 ms
Acceleration Factor	**~11.3651×**

Table 5.2.1 Avg Inference Delay & Avg Acceleration Factor

Even though the Intel CPU runs at a much higher clock speed, the FPGA-based accelerator finishes the same task 11× faster. This highlights the advantage of running a dedicated, parallelized hardware design for fixed-function tasks like CNN inference.

3. *Area Analysis*

Post-synthesis reports revealed the accelerator used FPGA resources efficiently, with logic, DSPs, and memory well balanced. The design meets timing requirements without congestion, allowing room for future expansion or added features without exceeding the FPGAs capacity.

Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	14182	53200	26.66
LUTRAM	965	17400	5.55
FF	14279	106400	13.42
BRAM	126.50	140	90.36
DSP	77	220	35.00
BUFG	1	32	3.13

Fig. 5.3.1 Post Implementation Utilization Report

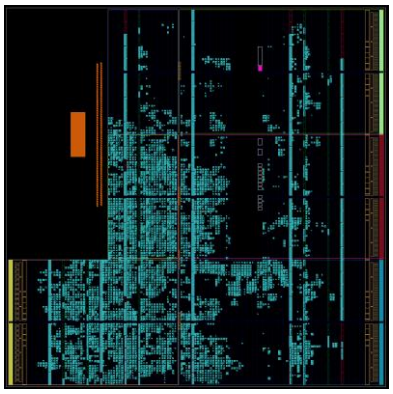

Fig. 5.3.2 FPGA Placement & Routing of CNN Accelerator

4. *Accuracy Analysis*

Image Index	*Python Model Inference Class*	*PYNQ-Z2 Inferenced Class*
image0	16	16
image1	1	1
image2	38	38
image3	17	17
image4	11	11
image5	38	38
image6	22	22
image7	12	12
image8	25	25
image9	35	35

Table 5.4.1 Python Model vs. PYNQ-Z2 Inference Class

The hardware-accelerated CNN implemented on the PYNQ-Z2 board demonstrates exceptional performance, preserving the inference accuracy of the original Python-based model entirely. As evident from Table 5.4.1, the class predictions generated by the FPGA match exactly with those of the software model, confirming the correctness and reliability of the hardware design.

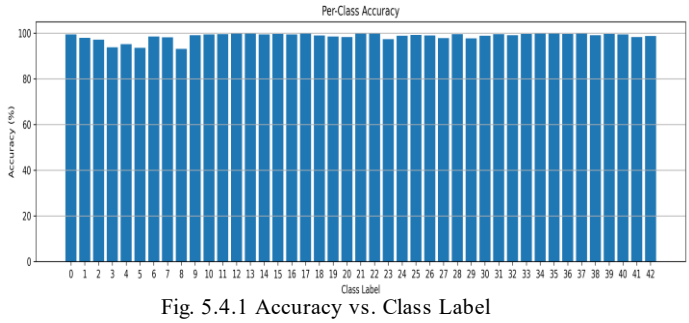

Fig. 5.4.1 Accuracy vs. Class Label

Figure 5.4.1 illustrates the per-class accuracy distribution across all 43 classes. Most of the classes achieve near-perfect accuracy, with several attainments of 100%. A few classes—such as Class 3, 4, and 5—show slightly lower performance (~93-95%), likely due to minor class imbalance or visual similarity between certain traffic signs. Nevertheless, accuracy remains consistently high, with no class falling below 93%.

This uniform accuracy distribution underscores the robustness of the model across diverse classes and validates the pipeline's ability to generalize well—even when deployed on hardware.

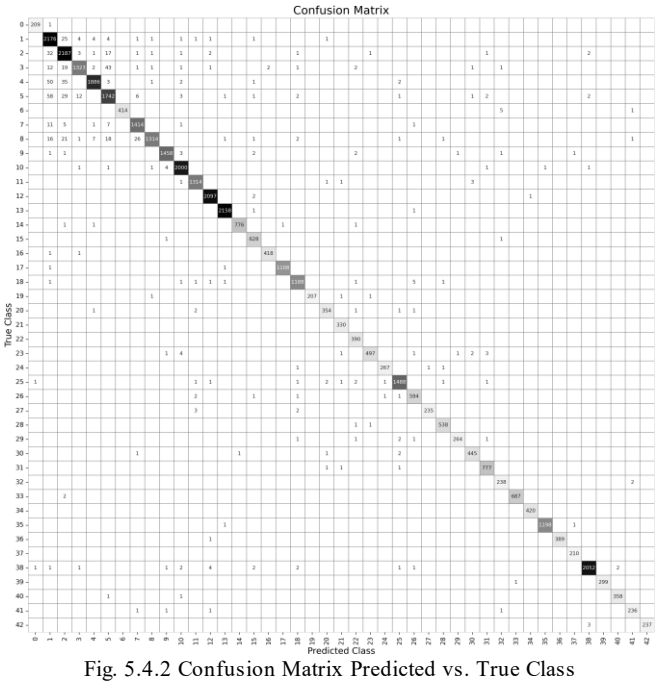

Fig. 5.4.2 Confusion Matrix Predicted vs. True Class

A confusion matrix was generated to visualize misclassifications. The matrix remains highly diagonal, further confirming that the classifier rarely confuses one class for another. Misclassifications, where present, were mostly limited to visually similar signs, such as speed limit signs

with marginal numeric differences. The near-zero off-diagonal entries confirm strong inter-class separability.

Metric	*Value*
Accuracy	0.9865
Precision (Macro)	0.9873
Recall (Macro)	0.9865
F1-Score (Macro)	0.9868
Precision (Weighted)	0.9828
Recall (Weighted)	0.9824
F1-Score (Weighted)	0.9824

Table 5.4.2 Model Performance Metrics

The macro-averaged metrics, which treat each class equally, reflect the model's strong performance across all categories, independent of class distribution. Weighted metrics, which factor in class support, also show high values, confirming that even frequent classes were handled accurately.

VI. CONCLUSION

This project showcases how combining high-level synthesis with hardware-software co-design can deliver real time, power-efficient AI on the PYNQ-Z2. The CNN accelerator runs fast and accurately, achieving over 11 times speedup compared to a CPU, while staying compact. With fixed-point quantization and efficient data handling, this FPGA-based approach offers a powerful, scalable solution for embedded AI applications like autonomous vehicles, smart surveillance, and industrial monitoring.

Codebase on GitHub is Publicly available on: https://github.com/kushaanbhat/CNN_HW_Accelerator

ACKNOWLEDGMENT

We thank Dr. Vivek Sunnapwar and Dr. Jayashree Khanapuri for their support. We are especially grateful to Prof. Sagar Vijay Mhatre for his valuable guidance and encouragement throughout this project.

REFERENCES

[1] S. Mhatre, V. J. Dongre and S. Mande, "Enhancing Edge Performance: A Comparative Analysis of LSTM Inference Using Hardware Acceleration," 2025 International Conference on Machine Learning and Autonomous Systems (ICMLAS), Prawet, Thailand, 2025

[2] H. Kim and K. Choi, "The Implementation of a Power Efficient BCNN-based Object Detection Acceleration on a Xilinx FPGA-SoC," in Proc. IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, USA, 2019, pp. 457-4571.

[3] K. S. P. Kaarmukilan, S. Poddar, and A. T. K., "FPGA based Deep Learning Models for Object Detection and Recognition: Comparison of Object Detection Models using FPGA," in Proc. 3rd International Symposium on Devices, Circuits, and Systems (ISDCS), 2020.

[4] H. Kim and K. Choi, "Low Power FPGA-SoC Design Techniques for CNN-based Object Detection Accelerator," in Proc. IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Atlanta, USA, 2019.

[5] M. Dhouibi, A. K. Ben Salem, and S. Ben Saoud, "CNN for object recognition implementation on FPGA using PYNQ framework," in Proc. International Conference, Tunisia Polytechnic School, University of Carthage, Tunisia, 2021.

[6] X. Guan, Z. Wang, and H. Fang, "Design and Implementation of CNN Accelerator Based on FPGA," School of Automation, Beijing Institute of Technology, Beijing, China.

[7] X. Zhen and B. He, "Research on FPGA High-Performance Implementation Method of CNN," in Proc. 2021 IEEE 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 2021, pp. 1-5.

[8] Kumar, N. S., & Madhumati, G. L. (2023). Implementation of convolutional neural networks on FPGA for object detection. 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), 1–5.

[9] Furuta, M., Ban, K., Kobayashi, D., & Shibata, T. (2021). An Efficient Implementation of FPGA-based Object Detection Using Multi-scale Attention. IEEE, 321–325.

[10] Sharma, A., Singh, V., & Rani, A. (2019). Implementation of CNN on Zynq based FPGA for Real-time Object Detection [Journal-article]. I.C.E Division, Netaji Subhas Institute of Technology, University of Delhi.

[11] Johnson, V. C., Bali, J., Tanvashi, S., & Kolanur, C. B. (2023). FPGA-Based Hardware Acceleration Using PYNQ-Z2. IEEE, 1–4.