CPU-Only Object Detection Baseline on Xilinx PYNQ-Z2

Project: Bharat AI-SoC Student Challenge

GitHub Repository: https://github.com/sudharson-pv/Bharat-AI-SoC-Student-Challenge

Team Members: Sudharson P V, Suriya Prasanth M, Sam Benny P

Program: B.E. Electronics Engineering (VLSI Design & Technology)

Institution: K. S. Rangasamy College of Technology

1. Abstrat

This project implements a CPU-only pedestrian detection system on the PYNQ-Z2 development board, built around the Zynq-7020 System-on-Chip. The work was carried out under Problem Statement 5 of the Bharat AI-SoC Student Challenge.

The detection pipeline follows a classical computer vision approach using Histogram of Oriented Gradients (HOG) for feature extraction, Support Vector Machine (SVM) for classification, and Non-Maximum Suppression (NMS) for refining bounding boxes.

All computations are executed exclusively on the dual-core ARM Cortex-A9 Processing System (PS). The FPGA Programmable Logic (PL) is intentionally not used in this phase. The goal is to establish a measurable CPU baseline for future FPGA-based acceleration.

2. Introduction

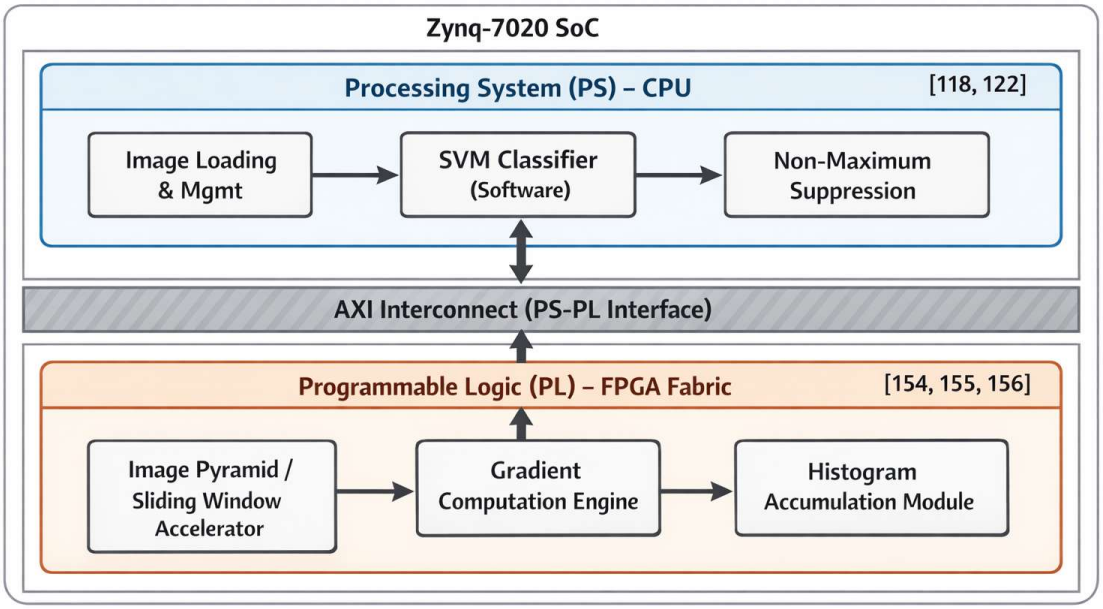
Embedded object detection is used in smart surveillance, automotive safety systems, edge AI deployments, and smart city infrastructure. Running real-time detection on resource-constrained SoCs introduces challenges such as limited computational throughput, restricted memory bandwidth, and tight power constraints.

This project evaluates the limitations of a CPU-only implementation before migrating computationally intensive components to programmable logic.

3. System Architecture

The system is implemented on the PYNQ-Z2 board integrating the Zynq-7020 SoC. The Processing System consists of a dual-core ARM Cortex-A9 processor with I-Cache, D-Cache, MMU (ITLB and DTLB), ALU, barrel shifter, multiplier, divider, FPU, and a 32×32-bit register file.

Only the ARM Processing System is used in this baseline. The FPGA fabric remains unused to preserve software-only benchmarking.



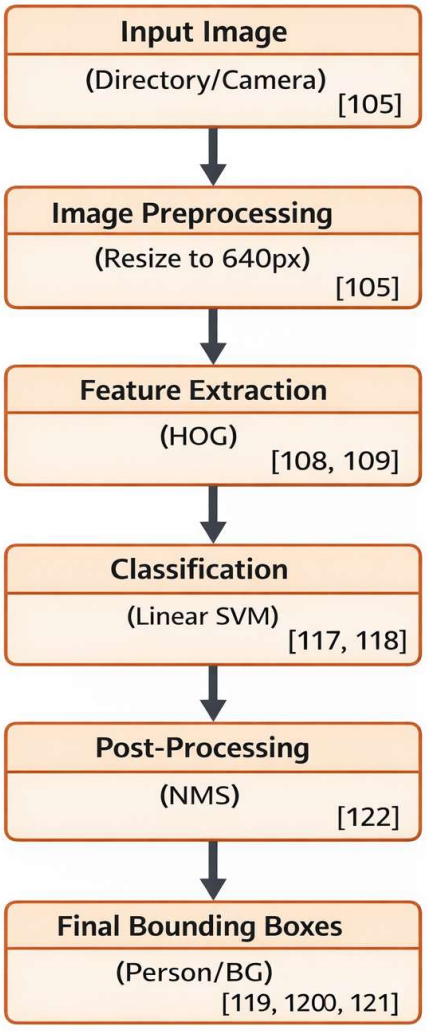
4. Methodology

All input images are resized to a fixed width of 640 pixels while preserving aspect ratio. HOG is used for feature extraction due to its deterministic behavior and suitability for embedded benchmarking.

Classification is performed using OpenCV's pre-trained SVM-based pedestrian detector. Non-Maximum Suppression (NMS) eliminates overlapping detections.

5. Experimental Setup

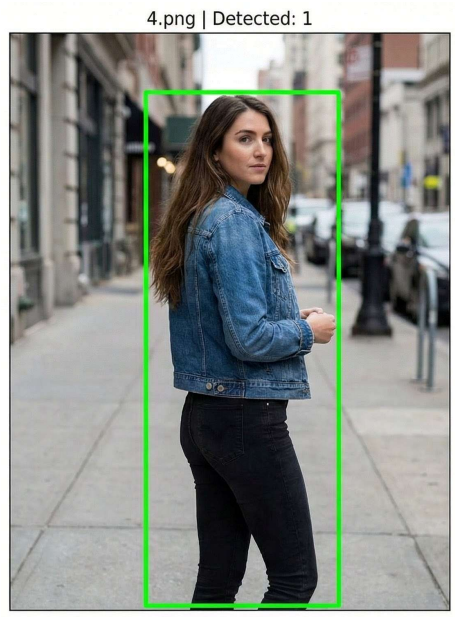
Image Directory: /home/xilinx/jupyter_notebooks/images
Software Stack: Python 3 with OpenCV (4.x)
Timing Method: time.time()
Only algorithm execution time is measured; visualization is excluded.

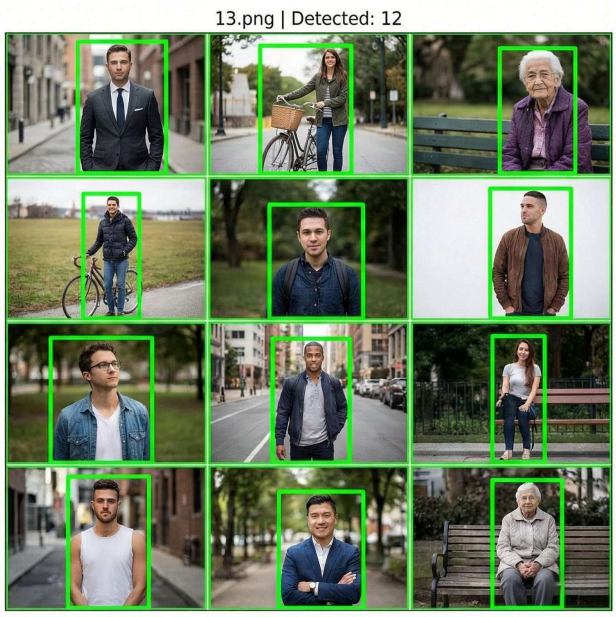


6. Performance Evaluation

Metrics include Average Latency, Minimum Latency, Maximum Latency, FPS, and Total Execution Time.

Performance bottlenecks include sliding window scanning, multi-scale image pyramid generation, CPU-only execution, absence of NEON SIMD optimization, and lack of hardware acceleration.





7. Significance of the Baseline

This CPU-only implementation provides measurable reference latency and FPS values. It serves as a quantitative benchmark for future FPGA acceleration and validates the need for hardware/software partitioning.

8. Future Work – FPGA Acceleration Strategy

Planned hardware modules include gradient computation engine, sliding window accelerator, histogram accumulation module, and AXI-based PS-PL communication.

```
===== CPU-ONLY PERFORMANCE (PYNQ-Z2) =====

Total images processed : 12
Average latency        : 7650.45 ms
Min latency            : 3100.10 ms
Max latency            : 9850.30 ms
Throughput (FPS)       : 0.13
Total execution time   : 91.80 s
Total execution time
```

Target Goals:

- ≥5× latency improvement
- >2 FPS threshold
- Optimized DDR transfers
- Reduced CPU workload

This will transform the design into a heterogeneous computing architecture.

9. Conclusion

The CPU-only pedestrian detection pipeline on PYNQ-Z2 establishes clear embedded performance limitations. Results confirm that CPU-only HOG detection is insufficient for real-time applications, reinforcing the need for FPGA-based acceleration.