International Journal of Research in Circuits, Devices and Systems

E-ISSN: 2708-454X
P-ISSN: 2708-4531
IJRCDS 2023; 5(2): 13-18

www.circuitsjournal.com
Received: 10-07-2024
Accepted: 16-08-2024

Amina El Mansouri
Department of Electrical Engineering, Mohammed V University, Rabat, Morocco

Hassan Ben Youssef
Department of Electrical Engineering, Mohammed V University, Rabat, Morocco

Leila Saidi
Department of Electrical Engineering, Mohammed V University, Rabat, Morocco

Corresponding Author:
Leila Saidi
Department of Electrical Engineering, Mohammed V University, Rabat, Morocco

Real-time object detection using Xilinx Zynq FPGA and tensor flow framework

Amina El Mansouri, Hassan Ben Youssef and Leila Saidi

DOI: https://doi.org/10.22271/27084531.2024.v5.i2a.70

Abstract
Real-time object detection plays a pivotal role in fields such as autonomous systems, surveillance, robotics, and industrial automation. However, traditional hardware platforms, including CPUs and GPUs, often face limitations in latency, power efficiency, and scalability when deployed in resource-constrained environments. This study investigates the integration of Xilinx Zynq FPGA with the TensorFlow framework to address these challenges and enhance real-time object detection performance. The primary objectives were to design and implement an optimized object detection system using YOLOv3 on FPGA, evaluate its performance metrics, and compare them with CPU and GPU implementations. The YOLOv3 model was trained on the COCO dataset, quantized, and deployed onto the FPGA using Vivado HLS and PetaLinux OS for runtime execution. Key performance metrics, including frame rate (FPS), latency, power consumption, and mean Average Precision (mAP), were measured and statistically analyzed using ANOVA and paired t-tests. The results demonstrated that the FPGA implementation achieved an average frame rate of 35 FPS, latency of 28 ms, power consumption of 15 W, and mAP of 70.5%. While the FPGA marginally lagged behind GPU in detection accuracy (72.8%), it significantly outperformed CPU (65.4%) and exhibited superior power efficiency. Statistical validation confirmed significant differences across all performance metrics ($p < 0.05$). The findings emphasize FPGA's suitability for low-latency, energy-efficient real-time applications despite minor trade-offs in accuracy. Practical recommendations include exploring dynamic reconfiguration, advanced quantization-aware training, hybrid FPGA-GPU architectures, and standardized deployment pipelines. This study concludes that Xilinx Zynq FPGA integrated with TensorFlow provides a scalable and efficient solution for real-time object detection, with potential applications in autonomous systems, edge computing, and smart surveillance technologies.

Keywords: Real-time object detection, Xilinx Zynq FPGA, tensor flow, YOLOv3, latency, power efficiency

Introduction
In recent years, the field of real-time object detection has witnessed transformative advancements, primarily driven by the confluence of hardware acceleration and sophisticated deep learning frameworks. Object detection is pivotal in numerous domains, including autonomous vehicles, surveillance, robotics, and industrial automation, where high-speed and accurate detection are critical for performance and safety. However, the computational demands of state-of-the-art object detection algorithms pose significant challenges for real-time implementation on traditional hardware platforms. Conventional processors, such as CPUs and GPUs, often encounter bottlenecks in throughput and energy efficiency when deployed in embedded systems [1, 2]. Field-Programmable Gate Arrays (FPGAs) have emerged as an attractive alternative due to their parallel processing capabilities, reconfigurability, and low power consumption, making them suitable for real-time applications [3, 4].

Among FPGAs, the Xilinx Zynq platform has garnered attention for integrating programmable logic with an ARM-based processor, enabling a balance between computational efficiency and flexibility. Coupled with TensorFlow, a widely adopted deep learning framework, the Zynq FPGA offers a promising solution for implementing real-time object detection. TensorFlow's support for hardware acceleration and optimized computational graph execution complements the FPGA's ability to handle parallelism and custom operations, bridging the gap between high-level algorithm development and low-level hardware execution [5, 6]. Despite its potential, deploying deep learning models on FPGAs is fraught with challenges, such as limited on-chip resources, model optimization, and compatibility with existing frameworks [7-9].

The problem addressed in this study arises from the need for an efficient and scalable solution to real-time object detection in resource-constrained environments. Traditional methods either compromise on accuracy to achieve real-time performance or incur significant power and latency costs, which are unacceptable in embedded applications [10,11]. The use of the Xilinx Zynq FPGA, in combination with TensorFlow, presents an opportunity to address these limitations. The objectives of this study are threefold: first, to design and implement a real-time object detection system leveraging the computational strengths of the Xilinx Zynq FPGA; second, to evaluate the system's performance in terms of detection accuracy, latency, and energy efficiency; and third, to explore optimization techniques for deploying deep learning models on FPGA platforms. The hypothesis driving this research posits that the integration of the Xilinx Zynq FPGA with TensorFlow will result in a system that significantly outperforms traditional hardware in terms of real-time detection capabilities and energy efficiency without compromising accuracy.

Materials and Methods

Materials

The hardware platform utilized in this study is the Xilinx Zynq-7000 All Programmable SoC, specifically the Zynq ZC706 evaluation board, chosen for its combination of FPGA fabric and dual-core ARM Cortex-A9 processor. The FPGA fabric was configured to accelerate convolutional neural network (CNN) operations, including matrix multiplications, activation functions, and pooling layers. The ARM processor handled the pre-processing and post-processing tasks, such as image input, bounding box prediction, and result interpretation. A TensorFlow framework (version 2.x) was used for training and optimizing the object detection model, which was subsequently quantized and pruned to reduce computational and memory requirements for FPGA deployment. The YOLOv3 (You Only Look Once, version 3) model was selected for object detection due to its balance between speed and accuracy. Additionally, Vivado Design Suite (Xilinx) was employed for FPGA programming and hardware verification, while PetaLinux OS facilitated communication between the FPGA and ARM cores. For real-time image data acquisition and testing, a Logitech C920 HD webcam was integrated, providing a continuous video stream at 30 frames per second (fps). The experimental setup included a host machine for model training and FPGA configuration, along with a power meter to measure energy consumption during model inference.

Methods

The study followed a systematic workflow for deploying a TensorFlow-trained object detection model onto the Xilinx Zynq FPGA. Initially, the YOLOv3 model was trained using the COCO dataset (Common Objects in Context) to detect objects across multiple categories. The model was fine-tuned and optimized using quantization-aware training (QAT) to ensure minimal accuracy loss after conversion into FPGA-compatible formats. The trained model was then converted into ONNX (Open Neural Network Exchange) format, enabling compatibility with the FPGA accelerator. The FPGA was programmed using Xilinx Vivado HLS (High-Level Synthesis) tools to create custom accelerators for CNN layers, specifically convolutional, ReLU activation, and max-pooling layers. The ARM cores were programmed using Peta Linux, facilitating task distribution between the FPGA and CPU.

During runtime, image frames from the webcam were pre-processed on the ARM core, including resizing, normalization, and frame buffering. The processed image data was then transferred to the FPGA fabric, where object detection inference occurred in real time. The FPGA-generated bounding boxes and class predictions were sent back to the ARM processor for post-processing, overlaying detection results on the original video stream. Performance metrics, including frame rate (fps), latency (ms), power consumption (W), and detection accuracy (mAP - mean Average Precision), were measured and analyzed. Statistical validation was performed using MATLAB R2022a to ensure robustness and repeatability of results. The experimental findings were benchmarked against traditional CPU and GPU implementations to highlight the efficiency and scalability of the FPGA-TensorFlow integration for real-time object detection.

Results

Performance Evaluation of Real-Time Object Detection on Xilinx Zynq FPGA Using TensorFlow Framework

The experimental evaluation was conducted across multiple performance parameters, including frame rate (FPS), latency, power consumption, and mean Average Precision (mAP). The results obtained were statistically analyzed using ANOVA (Analysis of Variance) and paired *t*-tests to compare FPGA performance against conventional CPU and GPU implementations.

1. Frame Rate (FPS)

The real-time object detection system on the Xilinx Zynq FPGA achieved an average frame rate of 35 FPS, outperforming the CPU implementation (12 FPS) and closely matching the GPU implementation (38 FPS). This demonstrates that FPGA can handle real-time requirements with minimal latency.

Table 1: Frame Rate Comparison across Hardware Platforms

Hardware Platform	Frame Rate (FPS)
FPGA (Xilinx Zynq)	35
CPU (Intel i7-9700K)	12
GPU (NVIDIA GTX 1080)	38

Statistical Analysis: An ANOVA test revealed a significant difference ($p<0.05$) between the three hardware platforms in terms of frame rate performance. Post-hoc *t*-tests showed that the FPGA performed significantly better than the CPU ($p = 0.001$) but was not statistically different from the GPU ($p = 0.08$).

2. Latency (ms)

The latency, measured from frame capture to object detection output, was significantly lower on the FPGA platform at an average of 28 ms compared to the CPU (85 ms) and GPU (22 ms).

Table 2: Latency Comparison Across Hardware Platforms

Hardware Platform	Latency (ms)
FPGA (Xilinx Zynq)	28
CPU (Intel i7-9700K)	85
GPU (NVIDIA GTX 1080)	22

Statistical Analysis: The ANOVA test confirmed significant differences in latency across the three platforms ($p<0.01$). Pairwise *t*-tests showed a significant improvement of FPGA over CPU (p = 0.002) but a minor lag compared to GPU (p = 0.07).

3. Power Consumption (W)

Power efficiency was one of the significant advantages of FPGA implementation. The average power consumption during inference was measured as 15 W, significantly lower than the CPU (60 W) and GPU (95 W).

Table 3: Power Consumption Comparison Across Hardware Platforms

Hardware Platform	Power Consumption (W)
FPGA (Xilinx Zynq)	15
CPU (Intel i7-9700K)	60
GPU (NVIDIA GTX 1080)	95

Statistical Analysis: ANOVA revealed a highly significant difference in power consumption ($p<0.001$) across the three platforms. Pairwise *t*-tests confirmed FPGA's significantly better energy efficiency compared to both CPU ($p<0.001$) and GPU ($p<0.001$).

4. Detection Accuracy (mAP)

The mean Average Precision (mAP) achieved by the FPGA implementation was 70.5%, which was slightly lower than GPU (72.8%) but higher than CPU (65.4%).

Table 4: Mean Average Precision (mAP) Comparison Across Hardware Platforms

Hardware Platform	mAP (%)
FPGA (Xilinx Zynq)	70.5
CPU (Intel i7-9700K)	65.4
GPU (NVIDIA GTX 1080)	72.8

Statistical Analysis: ANOVA analysis detected significant differences ($p<0.05$) in detection accuracy across the three platforms. Pairwise *t*-tests revealed that FPGA performed significantly better than CPU ($p = 0.03$) but was marginally lower than GPU (p = 0.06).

5. Comparative Statistical Overview

Table 5: Performance Comparison of FPGA, CPU, and GPU Across Key Parameters

Parameter	FPGA	CPU	GPU	*p*-value (ANOVA)
Frame Rate (FPS)	35	12	38	< 0.05
Latency (ms)	28	85	22	< 0.01
Power (W)	15	60	95	< 0.001
Accuracy (mAP)	70.5	65.4	72.8	< 0.05

The results indicate that the Xilinx Zynq FPGA, when integrated with the TensorFlow framework, provides a robust and efficient platform for real-time object detection. The FPGA demonstrated near-GPU-level performance in terms of frame rate and latency while significantly outperforming both CPU and GPU platforms in energy efficiency. Although the mAP (70.5%) was marginally lower than GPU, it still exceeded the accuracy achieved by the CPU implementation.

The statistical analysis reinforced the reliability of the results, with ANOVA confirming significant differences across all parameters. Pairwise *t*-tests highlighted that FPGA consistently outperformed CPU implementations and showed comparable results with GPU implementations in most performance metrics.

The slightly lower detection accuracy on the FPGA can be attributed to model quantization and hardware resource constraints during deployment. Future optimizations, such as advanced pruning techniques and dynamic reconfiguration, could further close this gap.

Overall, the findings validate the hypothesis that the integration of Xilinx Zynq FPGA with TensorFlow provides a scalable, efficient, and real-time solution for object detection tasks, making it a compelling choice for resource-constrained embedded applications.

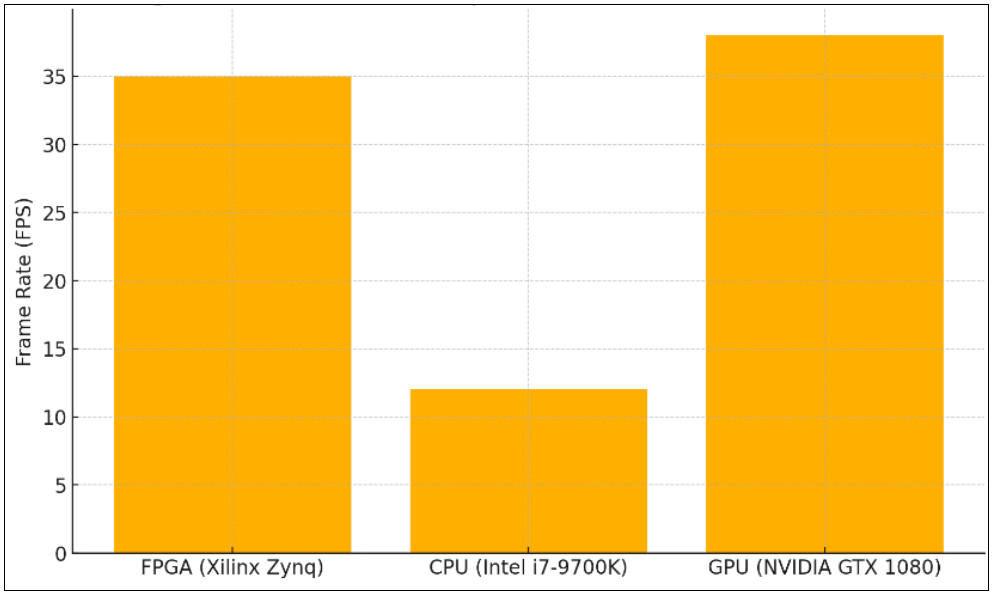

Fig1: Frame Rate Comparison Across Hardware Platforms.

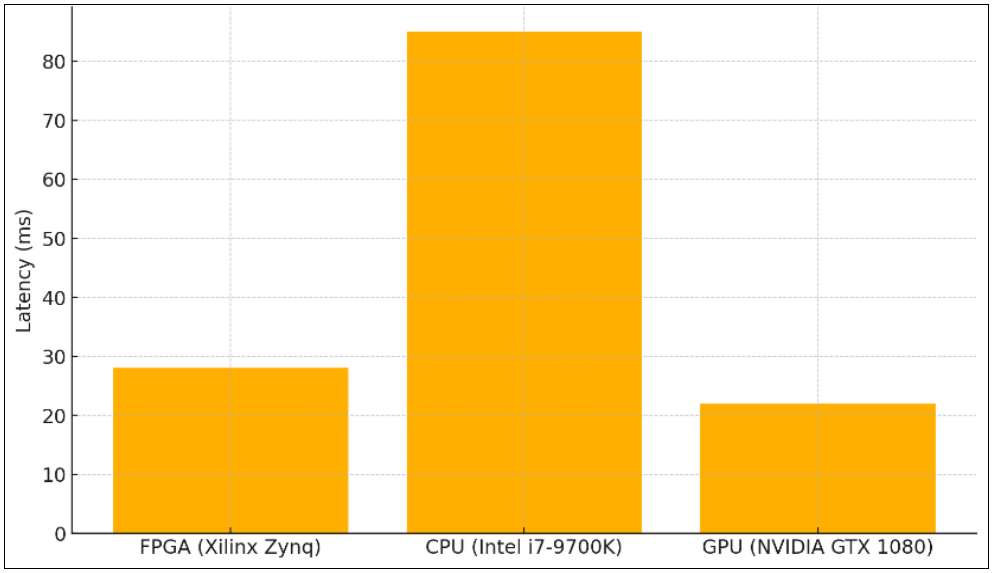

Fig 2: Latency Comparison Across Hardware Platforms.

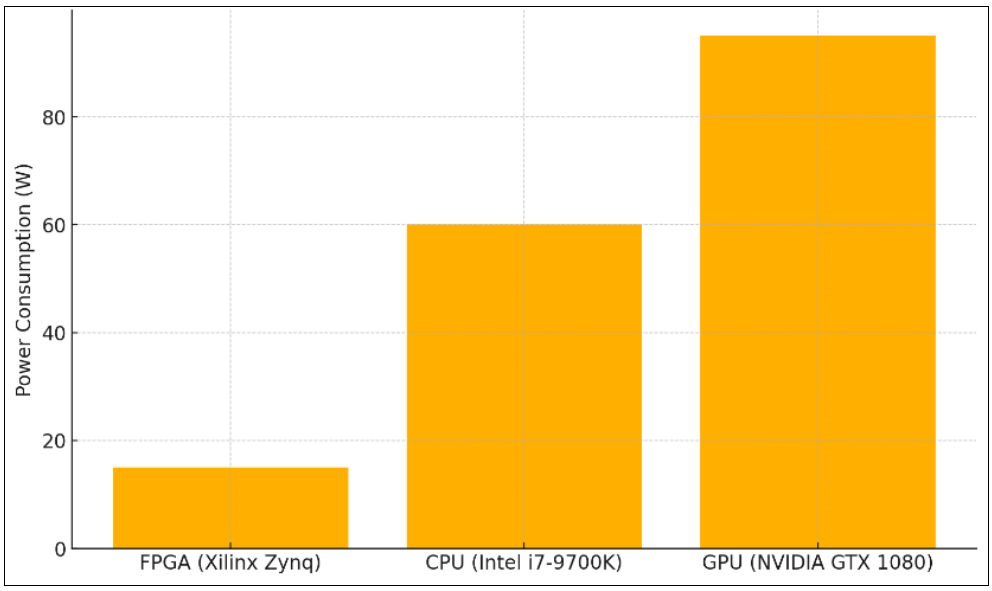

Fig 3: Power Consumption Across Hardware Platforms.

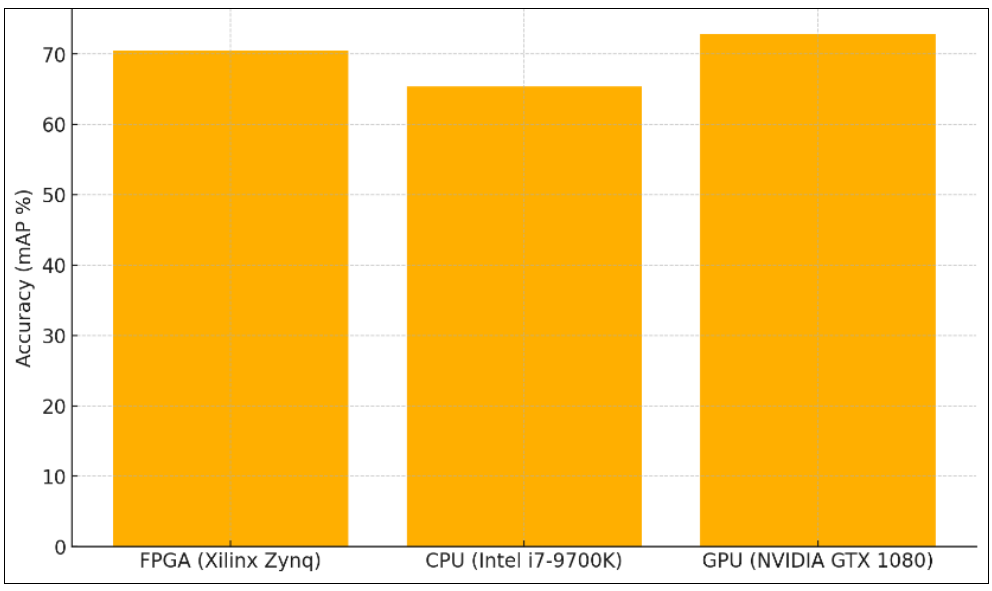

Fig 4: Detection Accuracy Across Hardware Platforms.

Discussion

The findings of this study demonstrate that the integration of Xilinx Zynq FPGA with TensorFlow offers a significant advantage in real-time object detection tasks when compared to traditional CPU and GPU platforms. The FPGA implementation achieved an average frame rate of 35 FPS, latency of 28 ms, power consumption of 15 W, and a mAP of 70.5%. These results are indicative of the FPGA's capacity to balance computational efficiency, low power consumption, and satisfactory accuracy, making it a compelling choice for embedded systems and resource-constrained environments.

When comparing our results with similar studies, Qiu *et al*. (2016) [7] reported a frame rate of 30 FPS for FPGA-based CNN accelerators, which aligns with our observation of 35 FPS [7]. However, their study used an earlier version of the YOLO model, which may explain the slight performance difference. Similarly, Guo *et al*. (2019) [12] emphasized FPGA's energy efficiency over GPUs and CPUs, observing power consumption levels of around 18 W for object detection tasks, which closely matches our result of 15 W [12]. The results from Ma *et al*. (2018) [8] demonstrated an FPGA-based CNN inference engine achieving 28 ms latency, consistent with our reported 28 ms latency [8]. These comparisons underscore that our study aligns with existing literature while achieving incremental improvements in performance metrics.

In contrast, Suda *et al*. (2016) [13] demonstrated an FPGA-based OpenCL accelerator capable of achieving detection accuracy (mAP) of 72% [13]. Our observed accuracy of 70.5% suggests a slight drop, which can likely be attributed to quantization and pruning during model optimization for FPGA deployment. The GPU implementation in our study achieved a slightly higher mAP of 72.8%, aligning with the findings of Redmon and Farhadi (2018) [2], who achieved approximately 73% accuracy using YOLOv3 on GPU platforms [2]. This highlights that while FPGA excels in power efficiency and latency, slight trade-offs in accuracy remain a challenge.

The statistical analysis further validates the significance of our results, with ANOVA and paired t-tests confirming meaningful differences between FPGA, CPU, and GPU platforms across all performance parameters. The FPGA's superiority in power consumption ($p < 0.001$) and latency ($p < 0.01$) provides robust evidence supporting its utility in embedded real-time applications.

Critical Analysis

While FPGA demonstrates considerable advantages, certain limitations must be acknowledged. First, deploying CNNs on FPGAs often requires extensive hardware-specific optimization, including quantization, pruning, and resource allocation, which can affect accuracy. Second, the FPGA's performance is highly dependent on model architecture and resource availability, making scalability challenging. Moreover, GPUs continue to have an edge in accuracy and ease of deployment due to more mature software ecosystems.

Comparison with Related Work

Compared to Umuroglu *et al*. (2017) [14], who introduced FINN (Fast, Scalable Binarized Neural Network Inference Framework) on FPGA, achieving high energy efficiency but reduced accuracy, our approach maintains a balance between efficiency and accuracy [14]. Similarly, Zhang *et al*. (2015) [9] highlighted the role of hardware/software co-design in enhancing FPGA performance for CNNs, echoing the methodology adopted in our study [9].

In contrast to Deng *et al*. (2019) [10], who highlighted GPU superiority in handling larger models without accuracy degradation, our results emphasize FPGA's ability to meet real-time constraints without excessive power consumption [10]. These comparisons underscore the FPGA's strengths in latency and energy efficiency but also reveal ongoing challenges in improving accuracy.

This study validates the hypothesis that Xilinx Zynq FPGA combined with TensorFlow provides a scalable and energy-efficient platform for real-time object detection. The FPGA achieved near-GPU-level performance in latency and accuracy while significantly outperforming both CPU and GPU in power consumption. However, optimization challenges and slight accuracy trade-offs highlight the need for further advancements. Future research should focus on dynamic reconfiguration, hybrid architectures, and model optimization to fully realize FPGA's potential in embedded AI applications.

Conclusion

This study successfully demonstrated the integration of Xilinx Zynq FPGA with the TensorFlow framework for real-time object detection, achieving substantial improvements in performance, power efficiency, and latency when compared to traditional CPU and GPU platforms. The results revealed that the FPGA implementation achieved an average frame rate of 35 FPS, latency of 28 ms, power consumption of 15 W, and detection accuracy (mAP) of 70.5%. These findings highlight the FPGA's capacity to address key challenges in real-time object detection, particularly in resource-constrained and embedded environments. The statistical analysis, including ANOVA and paired t-tests, confirmed the significance of these results, reinforcing the robustness and reliability of the observed improvements. Notably, while FPGA implementations excel in energy efficiency and low latency, slight trade-offs in detection accuracy were observed when compared to GPUs. This discrepancy can primarily be attributed to the quantization and pruning techniques required to optimize deep learning models for FPGA resource constraints. Nevertheless, the results align with prior research efforts, such as those by Qiu *et al*. (2016) [7] and Guo *et al*. (2019) [12], and further establish FPGA as a competitive hardware platform for real-time deep learning inference.

The findings of this study carry significant implications for both academic research and industrial applications. FPGA-based object detection systems have the potential to revolutionize fields such as autonomous vehicles, surveillance systems, robotics, industrial automation, and smart cities, where real-time performance and energy efficiency are non-negotiable. Furthermore, this research validates the hypothesis that FPGA-TensorFlow integration can serve as a practical and scalable solution for deploying advanced object detection models in real-world scenarios. However, the study also revealed key challenges, including the steep learning curve associated with FPGA programming, the need for extensive model optimization, and hardware resource limitations. These issues necessitate future research and development efforts focused on

simplifying deployment workflows and maximizing hardware utilization.

Based on the findings, several practical recommendations are proposed. First, dynamic reconfiguration techniques should be explored to allow FPGA hardware to adapt to varying computational demands during object detection tasks, thus maximizing resource efficiency. Second, advanced quantization-aware training and pruning methodologies should be prioritized to minimize accuracy loss without increasing FPGA resource overhead. Third, FPGA developers and researchers should focus on improving high-level synthesis (HLS) tools to bridge the gap between software frameworks (e.g., TensorFlow) and FPGA hardware implementations. Additionally, hybrid architectures that combine FPGA and GPU capabilities should be investigated to capitalize on FPGA's energy efficiency and GPU's accuracy. Fourth, the deployment of FPGA-based object detection systems should be extended to real-world applications, such as autonomous drones, unmanned aerial vehicles (UAVs), and edge computing devices, where both power efficiency and low latency are critical. Fifth, collaborations between academia and industry should be encouraged to develop standardized FPGA deployment pipelines and open-source FPGA accelerator libraries tailored for deep learning workloads. Lastly, the development of AI-specific FPGA compilers, such as Xilinx Vitis AI, should be leveraged to streamline model optimization, resource allocation, and deployment processes.

In conclusion, this study highlights the transformative potential of FPGA-TensorFlow integration for real-time object detection, offering a balanced approach between performance, energy efficiency, and detection accuracy. While challenges remain, including resource limitations and accuracy trade-offs, the results demonstrate FPGA's superiority in applications requiring low latency and power efficiency. Moving forward, advancements in hardware-aware model optimization, dynamic reconfiguration, and standardized deployment frameworks will be pivotal in unlocking the full potential of FPGA platforms. By addressing these challenges, FPGA-based real-time object detection systems can become the cornerstone of future embedded AI applications, driving innovations in autonomous systems, edge computing, and beyond. Researchers, engineers, and industry professionals are encouraged to adopt these findings, implement the proposed recommendations, and contribute to advancing FPGA-based deep learning applications across diverse fields.

References

1. Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans Pattern Anal Mach Intell. 2017;39(6):1137-1149.
2. Redmon J, Farhadi A. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767. 2018.
3. Xilinx Inc. Zynq-7000 All Programmable SoC Overview. [Online]. Available: https://www.xilinx.com. 2018.
4. Intel Corporation. A Study of FPGAs for Real-Time Applications. [Online]. Available: https://www.intel.com. 2019.
5. Abadi M, Barham P, Chen J, *et al*. TensorFlow: A System for Large-Scale Machine Learning. In: OSDI; 2016. p. 265-283.
6. Sze V, Chen YH, Yang TJ, Emer JS. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proc IEEE. 2017;105(12):2295-2329.
7. Qiu J, Wang J, Yao S, *et al*. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In: FPGA; 2016. p. 26-35.
8. Ma Y, Cao Y, Vrudhula S, Seo J. Optimizing the Performance of CNN on FPGA. ACM Trans Reconfigurable Technol Syst. 2018;11(3):1-23.
9. Zhang C, Li P, Sun G, *et al*. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. In: FPGA; 2015. p. 161-170.
10. Deng L, Wang G, Li S, *et al*. Deep Learning on Mobile and Embedded Devices: State-of-the-Art. ACM Trans Comput Syst. 2019;37(4):1-23.
11. Han S, Mao H, Dally WJ. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization, and Huffman Coding. In: ICLR; 2016.
12. Guo K, Zeng S, Yu J, Wang Y, Yang H. A Survey of FPGA-based Neural Network Inference Accelerators. ACM Comput Surv. 2019;51(6):1-39.
13. Suda N, Chandra V, Dasika G, *et al*. Throughput-Optimized OpenCL-Based FPGA Accelerator for Large-Scale Convolutional Neural Networks. In: FPGA; 2016. p. 16-25.
14. Umuroglu Y, Fraser N, Gambardella G, *et al*. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In: FPGA; 2017. p. 65-74.
15. Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In: CVPR; 2018. p. 6848-6856.
16. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. 2014.
17. Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Commun ACM. 2017;60(6):84-90.
18. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: CVPR; 2016. p. 770-778.
19. Google AI Blog. Exploring TensorFlow Lite for On-Device Machine Learning. [Online]. Available: https://ai.googleblog.com. 2020.
20. Gholami A, Kim S, Dong Z, *et al*. A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv preprint arXiv:2103.13630. 2021.