Hardware Acceleration for Object Detection using YOLOv5 Deep Learning Algorithm on Xilinx Zynq FPGA Platform

Taoufik Saidani
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia | Laboratory of Electronics and Microelectronics (EμE), Faculty of Sciences, Monastir University, Tunisia
taoufik.saidan@nbu.edu.sa (corresponding author)

Refka Ghodhbani
Department of Computer Science, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia | Laboratory of Electronics and Microelectronics (EμE), Monastir University, Faculty of Sciences, Tunisia
refka.ghodhbani@nbu.edu.sa

Ahmed Alhomoud
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
aalhomoud@nbu.edu.sa

Ahmad Alshammari
Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
ahmad.almkhaidsh@nbu.edu.sa

Hafedh Zayani
Department of Electrical Engineering, Faculty of Engineering, Northern Border University, Saudi Arabia
hafedh.zayani@nbu.edu.sa

Mohammed Ben Ammar
Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Saudi Arabia
mohammed.ammar@nbu.edu.sa

Received: 16 December 2023 | Revised: 1 January 2024 | Accepted: 9 January 2024

 DOI: https://doi.org/10.48084/etasr.6761

ABSTRACT

Object recognition presents considerable difficulties within the domain of computer vision. Field-Programmable Gate Arrays (FPGAs) offer a flexible hardware platform, having exceptional computing capabilities due to their adaptable topologies, enabling highly parallel, high-performance, and diverse operations that allow for customized reconfiguration of integrated circuits to enhance the effectiveness of object detection accelerators. However, there is a scarcity of assessments that offer a comprehensive analysis of FPGA-based object detection accelerators, and there is currently no comprehensive framework to enable object detection specifically tailored to the unique characteristics of FPGA technology. The You Only Look Once (YOLO) algorithm is an innovative method that combines speed and accuracy in object detection. This study implemented the YOLOv5 algorithm on a Xilinx® Zynq-7000 System on a Chip (SoC) to perform real-time object detection. Using the MS-COCO dataset, the proposed study showed an

improvement in resource utilization with approximately 42 thousand (78%) look-up tables, 56 thousand (52%) flip-flops, 65 (46%) BRAMs, and 19 (9%) DSPs at a frequency of 250 MHz, improving the effectiveness compared to previous simulated results.

Keywords-object detection; YOLOv5; high level synthesis; FPGA; HDL coder

I. INTRODUCTION

Using state-of-the-art hardware platforms, such as GPUs and FPGAs, has substantially improved deep learning performance in object detection. Several industries find these platforms useful, including video surveillance, industrial design, target tracking, and Advanced Driver Assistance Systems (ADAS). Deep learning object identification has been considerably improved by the progress of many techniques, such as the Single-Shot Detector (SSD) [1], the Spatial Pyramid clustering network (SPP-net) [2], and the Region-based Convolutional Neural System [3]. You Only Look Once (YOLO) [4] and Faster R-CNN [5] are two referenced methodological approaches. Among the many effective deep learning techniques, the YOLO approach stands out for its remarkable speed and reliability in object detection [6]. The YOLO method uses a single convolutional neural network to predict the item's confidence level, the likelihood of classes linked to the bounding boxes, and the coordinates of the boxes. Standard Application-Specific Integrated Circuits (ASICs) are not very effective in some tasks. As FPGAs use reconfiguration to maximize the adaptability-performance trade-off [7], they are preferred for the execution of deep learning algorithms compared to ASICs and GPUs [7]. This study used YOLO networks to identify objects in the MS-COCO benchmark dataset, which includes a wider variety of categories and occurrences compared to the PASCAL VOC and ImageNet datasets [8]. The MS-COCO dataset consists of photographs showcasing 91 unique item types, with almost 2.5 million accurately annotated instances. This dataset comprises around 10% of a certain category per image, in contrast to the PASCAL VOC and ImageNet benchmark datasets, which encompass 60% of both categories.

Many studies have discussed the design and implementation of FPGAs for deep learning algorithms due to their suitability for this purpose [9-10]. Several FPGA topologies have been proposed for the execution of YOLO algorithms. In [10], the REQ-YOLO framework was proposed, which was specifically designed for object identification and fine-tuned for FPGA implementation and incorporated an innovative Processing Element (PE) arrangement. However, this framework demonstrated an exceptionally high amount of resource utilization. In [11], the rapid Finite Impulse Response (FIR) technique, often known as FFA, was proposed to improve resource use. This technique was specifically designed to efficiently compute CNN models and was successfully implemented on a Xilinx Zynq ZC706. MiniYOLOv3 is an embedded software that has been built and tested using the MS-COCO dataset for real-time applications. In [12], a Multi-Scale Feature Pyramid Network (MSFPN) was specifically built to extract features in a very efficient manner. The design used both group and depth-wise convolution, however, it did not achieve better accuracy and speed than the proposed technique.

In [13], a literature survey was conducted for hardware implementations of visual trackers, highlighting the lack of hardware implementations of state-of-the-art tracking algorithms. In [14], VGG16 was implemented on Xilinx Zynq ZC706 and Virtex VC707 boards, respectively. In [15], the REQ-YOLO was proposed as a resource-aware, systematic weight quantization framework for object detection. In [16], the binary network was combined with a Support Vector Machine (SVM) regression to specifically target lightweight YOLO-v2 using a Xilinx Zynq® Ultrascale+ MPSoC. The use of a shared streaming binarized CNN to sequentially process each layer can effectively decrease the computing complexity of the procedure but does not adequately address the problem of decreasing external memory access. To achieve this objective, a hardware architecture using the VC707 FGPA was introduced in [16] that integrated fast processing capabilities with low power usage, specifically designed for object detection. The resource use for this architecture remained consistently high, regardless of the total efficiency and power. Another limitation was the need to optimize latency, as stated in [12], and the use of a compact YOLOv3 architecture specifically optimized for low-end FPGAs was proposed. Although many FPGA-based hardware solutions have shown the ability to improve CNNs, only a limited number of these systems have extensively studied the complete design process to achieve fast deployment and outstanding power efficiency.

This study proposes the utilization of FPGAs to accelerate the YOLOv5 algorithm, aiming to reduce prediction time while maintaining a high level of accuracy. The YOLOv5 object identification algorithm was accelerated on the Zynq-7000 SoC using efficient resource use, and its performance was compared to other existing approaches. The YOLOv5 algorithm was implemented with hardware acceleration using Vivado 2017.4.

II. YOLO FOR OBJECT DETECTION

The YOLO approach employs the Darknet-53 framework [17] for object detection, which is a cutting-edge algorithm known for its exceptional efficiency and accuracy. The Darknet architecture consists of 53 convolutional layers, which is evident from its name. YOLO v5 has superior speed compared to its predecessors. YOLOv5 employs the Darknet framework as its foundational structure, incorporating YOLOv3 on top. The intermediate layer comprises the Path Aggregation Network (PAN) [18], Spatial Pyramid Pooling (SPP) [19], and the Feature Pyramid Network (FPN) [20]. Figure 1 illustrates the conventional structure of YOLOv5. Extracting features from images involves the deployment of convolutional layers and the use of anchor boxes with a kernel size of $k \times k$ to predict the bounding boxes and perform regression, finally producing a feature map. Additionally, it incorporates pooling layers that are used to sample each input map. A Fully Connected (FC) layer can operate as a classifier. Neural networks use activation functions to enhance their ability to accurately represent data, which is achieved by incorporating layers of nonlinearity. The

activation functions often employed in neural networks include the Rectified Linear Unit (ReLU), Leaky ReLU, Sigmoid, and Hyperbolic Tangent functions.

YOLO denotes the phrase "You Only Look Once". This study used version 5, a cutting-edge object detection algorithm that currently stands as the most advanced algorithm of its kind. This is a revolutionary CNN that accurately detects objects in real-time. This method uses a singular neural network to analyze the entirety of the image, subsequently segmenting it into distinct components and making predictions on the bounding boxes and probabilities associated with each individual element. The bounding boxes are assigned weights based on the anticipated probability. The "just looks once" method examines the image by making predictions after a single forward propagation pass through the neural network. After performing non-maximum suppression, which guarantees that each object is only identified once, the algorithm presented in Figure 2 proceeds to provide the discovered items.

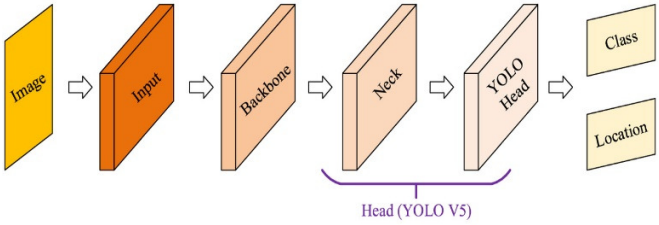

Fig. 1. YOLOv5 system for object detection.

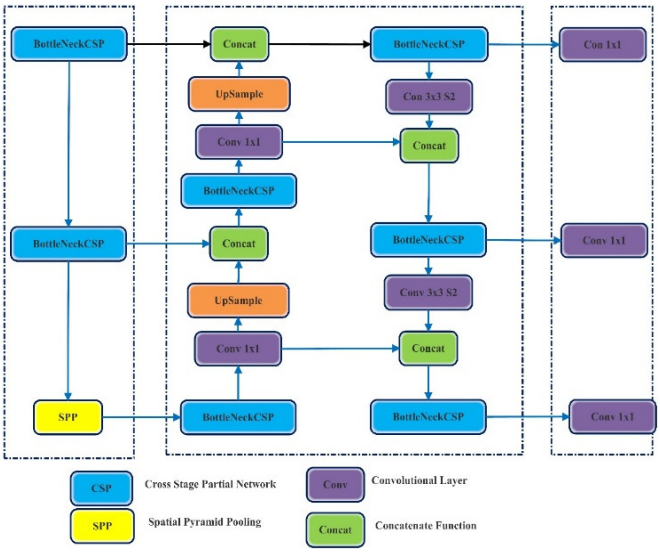

Fig. 2. The algorithm of the YOLOv5 model.

III. PROPOSED ARCHITECTURE

Figure 3 shows the proposed hardware implementation for YOLOv5, illustrating how the Zynq platform can execute the algorithm. Xilinx Programmable Logic (PL) from the 7-series [13], AMBA Interconnects, DDR3 component memory, an ARM Cortex-A9 Processing System (PS), and some peripherals are part of the ZynqTM 7000 XC7Z020-CLG484-1 SoC that was used in the implementation.

Vivado 20.17.4 has a plethora of libraries for the implementation of deep learning algorithms [16]. Deep learning algorithms are implemented using Vivado High-Level Synthesis (HLS) in conjunction with certain Targeted Reference Designs (TRDs) [1]. Each layer is processed sequentially in hardware acceleration. The designer has the option to change the 3×3 kernel size of the CONV layer. When designing the streaming system, it is essential to optimize the timing for each layer. To increase the system throughput in high-level synthesis, loop pipelining is used because of the large amount of incoming data. The Leaky ReLU activation function is used in the network model to solve the gradient vanishing problem [1].

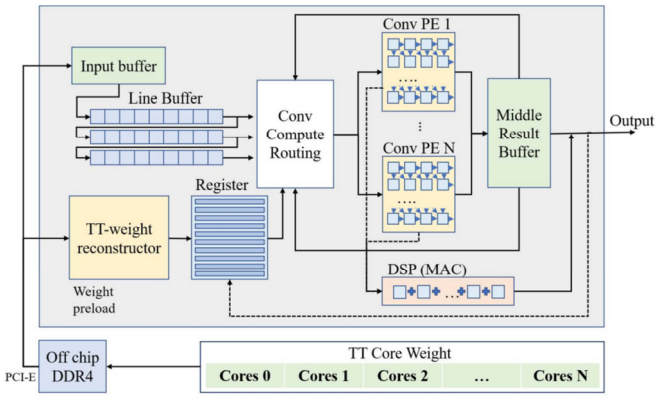

Fig. 3. Proposed architecture of YOLOv5.

A. *Design Overview*

The YOLOv5 network processes each layer in a specific order, except for the routing layer. It is possible to predetermine the routing layer by providing an address. A memory-data connection is needed for the hardware accelerator to read, process, and write data [2]. By recycling data and tiling the convolution loops, the loop tiling technique optimizes memory access time, which is especially useful for huge amounts of data. When the burst duration exceeds a specific threshold, the effective bandwidth of the FPGA begins to plateau after initially increasing with the burst length. In terms of data layout, the data tiling approach frequently causes DRAM to have non-continuous access. Combining them into a single block, the proposed technique improves the placement of kernel weights inside a tile. This improves effective memory bandwidth by making the most of available external memory bandwidth while reducing the number of memory accesses. Enhancing computational performance through the use of input and output parallelism is crucial for convolutional layer acceleration. Achieving parallelism requires the construction of many parallel addition trees and multiplication units.

Data received from DRAM are distributed to the on-chip buffers by the Data Scatter module, which is also responsible for producing the correct write address. Transferring data from the output buffer back to the DRAM and producing the DRAM write-back address are the two main functions of the Data Gather module. Computations for the Pool, Conv, and Reorg layers, two convolutional layers and one with a Leaky ReLU,

are handled by the pixel buffers. For data translation, the upgraded hardware has one AXI4-Lite slave interface in addition to two AXI4 master interfaces. A link is formed between the weighted buffers and the FC layer using the proposed technique.

B. Hardware Implementation

Several components are part of the proposed FPGA implementation, including an Interrupt Controller, Direct Memory Access (DMA), and General Purpose Input/Output (GPIO). A parameter and processing array, data decoding and rearranging modules, and the PL itself make up the program. The DDR memory in both components is used to connect to the SoC through the Memory Generator Interface, which also carries out the storage of network parameters and mapping of features.

The Deep Learning HDL Toolbox2 in MATLAB R2018b (MathWorks Inc.) is compatible with Xilinx and Intel FPGAs and SoCs for inference purposes. This toolbox allows users to create personalized deep-learning models and assess the performance of hardware implementation. It is feasible to construct and train DL models in MATLAB, as it is also possible to import pre-trained models from different frameworks. The HDL toolkit facilitates the use of 8-bit integer quantization for weight optimization, resulting in enhanced throughput, memory efficiency, power consumption, and computing demands. The DL model can be transformed into portable and synthesizable Verilog or VHDL code using HDL Coder and Simulink. This code can then be deployed on hardware such as FPGA and SoCs. An advantage of this toolbox is its ability to automatically produce FPGA instructions through the compile command, eliminating the need for human reprogramming. The FPGA can be connected and instructions can be deployed using either the Ethernet or JTAG interface, utilizing the DL processor IP core. When making predictions, it is possible to assess the real-time device performance parameters, such as latency at the layer level, to identify any limitations. The synthesizable Register Transfer Level (RTL) generated by HDL Coder is capable of accommodating various workflows and devices. A customizable IP core that features standard AXI interfaces can be developed for seamless integration into SoCs.

Figure 4 shows the IP core of the MATLAB HDL processor. The toolbox comprises a DL processor with four AXI4 master interfaces, DDR external memory, convolution, FC layer processor, activation normalization, and a controller with scheduling logic. The input data and parameters can be saved in the external DDR memory and then transferred to the block RAM via one of the AXI4 master interfaces. The block RAM supplies activations to the general convolution and fully connected processor. The convolution and FC processor execute the convolution operation and compute the FC layers. The activation normalization module is responsible for including the ReLU nonlinearity, a max-pooling layer, or executing Local Response Normalization (LRN). The IP core consists of two controllers: one for the convolution and another for the FC layer. The generic convolutional/fully connected processor with activation normalization can process only one layer sequentially. The subsequent layer undergoes processing through the controller, specifically in terms of scheduling.

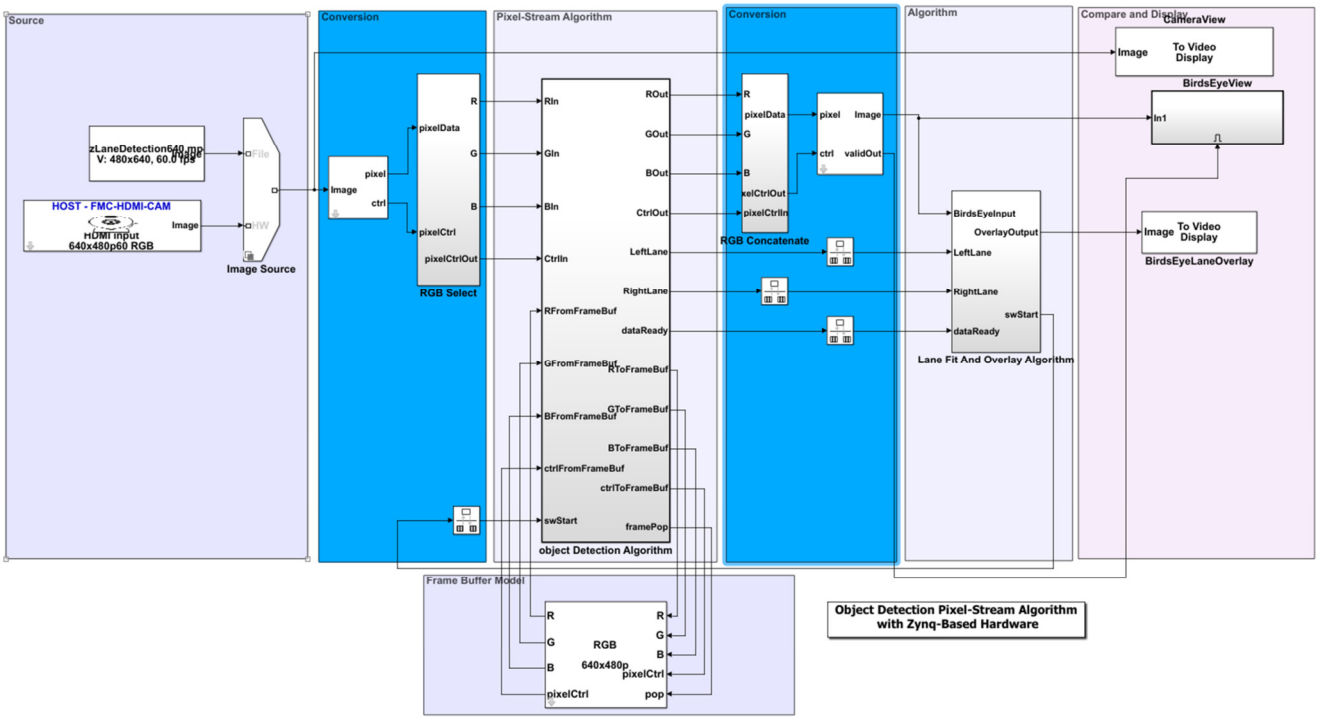

Fig. 4. HDL coder proposal for object detection on Zynq7000.

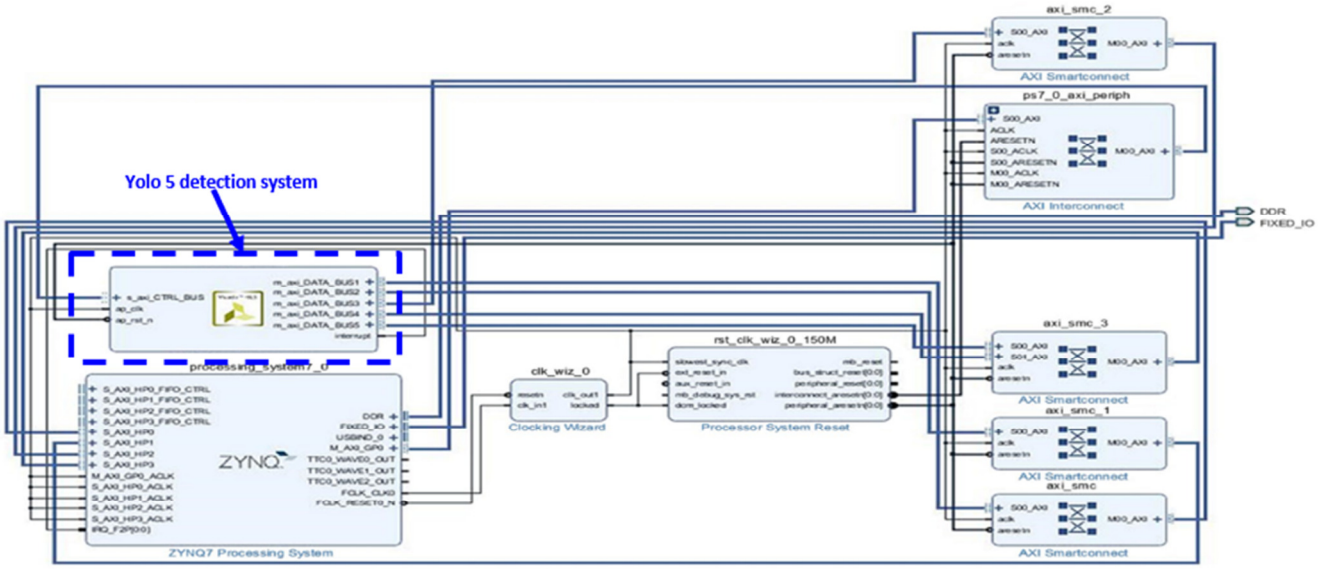

Fig. 5. RTL design for YOLOv5 for object detection Vivado project.

The processing phase involves sending PL configuration instructions via GPIO after ARM executes them for each layer. Control signals are transmitted to the relevant units based on decoded instructions. The original picture is retrieved from the Programmable SoC Direct Read (PSDDR) module and transmitted to the PL module by the Direct Memory Access (DMA) module. Before transmission to the processing array, the pixels are rearranged by the input data reordering module. The PL-DDR provides the DDR controller in PL with the model parameters, which are then stored in the parameters buffer.

The Processing Array (PA) receives its parameters from the parameter buffer. Processing Elements (PEs) are used to construct PAs to facilitate concurrent processing. These components of parallel processing perform calculations for several output channels using the same input feature maps. These parallel processing components use the same input feature mappings to compute for a large number of output channels. At the same time, the PEs complete the computations for each layer. The next step is for the PS-DDR to help get the feature maps from the last layer back to the host PC. To obtain the object detection results, the host PC uses the final feature maps to run Non-Maximum Suppression (NMS).

Resource availability and data interdependence in the design are the limiting factors in the achieved data transmission rate. Figure 5 shows the proposed implementation of Zynq-SoC-based real-time object detection. The proposed approach uses more BRAMs, therefore, it can be implemented with a suitable data processing rate and great efficiency for object detection in real-time. Table I shows a qualitative example of hardware acceleration using the MS-COCO dataset in the proposed system for object detection. The proposed YOLOv5-SOC can achieve an overall performance of 120 fps while effectively predicting different types of items.

TABLE I. RESOURCE UTILIZATION AND MAXIMUM FREQUENCY FOR THE PROPOSED SYSTEM

Xilinx platform	Zynq 7020
Maximum frequency	250 Mhz
LUTs	42025 (78.99%)
Flip flops	56141 (52.76)
BRAMs	65.50 (46.79%)
DSPs	19 (8.64%)

IV. CONCLUSION

This study aimed to enhance the performance of object detection by developing a reconfigurable architecture. The YOLOv5 algorithm for object detection was implemented with hardware acceleration on the Zynq SoC. The MS-COCO benchmark dataset was used to evaluate the YOLOv5 algorithm for object detection when executed with hardware acceleration on the Zynq SoC. The Vivado HLS tool was used to synthesize and implement an object detection algorithm on an SoC. This implementation achieves superior resource efficiency, with 78% of Look-Up Tables (LUTs), 52% of Flip-Flops, 42% of Block RAMs (BRAMs), and 8.4% of Digital Signal Processors (DSPs) being utilized. These results were compared with previous implementations. The proposed acceleration achieved a performance of 120 fps at a 250 MHz frequency while minimizing power consumption and effectively utilizing resources. The results demonstrate that the implementation of the YOLOv5 algorithm on FPGA hardware yields a performance that is twice as efficient, with significantly reduced prediction time. FPGAs are well-suited for implementing intricate algorithms, while their primary negative is the extensive time required for development. Despite the development of new tools, reconfiguration of FPGAs remains a tough task.

ACKNOWLEDGMENT

The authors gratefully acknowledge the approval and the support of this research study by the Grant No. CSCR-2023-

12-2065 from the Deanship of Scientific Research at Northern Border University, Arar, K.S.A.

REFERENCES

[1] T. Saidani, "Deep Learning Approach: YOLOv5-based Custom Object Detection," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12158–12163, Dec. 2023, https://doi.org/10.48084/etasr.6397.

[2] R. Ghodhbani, T. Saidani, A. Alhomoud, A. Alshammari, and R. Ahmed, "Real Time FPGA Implementation of an Efficient High Speed Harris Corner Detection Algorithm Based on High-Level Synthesis," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12169–12174, Dec. 2023, https://doi.org/10.48084/etasr.6406.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 580–587, https://doi.org/10.1109/CVPR.2014.81.

[4] A. B. Amjoud and M. Amrouch, "Object Detection Using Deep Learning, CNNs and Vision Transformers: A Review," *IEEE Access*, vol. 11, pp. 35479–35516, 2023, https://doi.org/10.1109/ACCESS.2023.3266093.

[5] X. Yang, C. Zhuang, W. Feng, Z. Yang, and Q. Wang, "FPGA Implementation of a Deep Learning Acceleration Core Architecture for Image Target Detection," *Applied Sciences*, vol. 13, no. 7, Jan. 2023, Art. no. 4144, https://doi.org/10.3390/app13074144.

[6] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv, Apr. 22, 2020, https://doi.org/10.48550/arXiv.2004.10934.

[7] A. Boutros, S. Yazdanshenas, and V. Betz, "You Cannot Improve What You Do not Measure: FPGA vs. ASIC Efficiency Gaps for Convolutional Neural Network Inference," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, Sep. 2018, https://doi.org/10.1145/3242898.

[8] R. Rajamohanan and B. C. Latha, "An Optimized YOLO v5 Model for Tomato Leaf Disease Classification with Field Dataset," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12033–12038, Dec. 2023, https://doi.org/10.48084/etasr.6377.

[9] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-Based Accelerators of Deep Learning Networks for Learning and Classification: A Review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019, https://doi.org/10.1109/ACCESS.2018.2890150.

[10] E. Wang *et al.*, "Deep Neural Network Approximation for Custom Hardware: Where We've Been, Where We're Going," *ACM Computing Surveys*, vol. 52, no. 2, Feb. 2019, https://doi.org/10.1145/3309551.

[11] M. A. Dias and D. A. P. Ferreira, "Deep Learning in Reconfigurable Hardware: A Survey," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Rio de Janeiro, Brazil, Feb. 2019, pp. 95–98, https://doi.org/10.1109/IPDPSW.2019.00026.

[12] Q. C. Mao, H. M. Sun, Y. B. Liu, and R.-S. Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications," *IEEE Access*, vol. 7, pp. 133529–133538, 2019, https://doi.org/10.1109/ACCESS.2019.2941547.

[13] A. H. A. El-Shafie and S. E. D. Habib, "Survey on hardware implementations of visual object trackers," *IET Image Processing*, vol. 13, no. 6, pp. 863–876, 2019, https://doi.org/10.1049/iet-ipr.2018.5952.

[14] J. Wang, J. Lin, and Z. Wang, "Efficient Hardware Architectures for Deep Convolutional Neural Network," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1941–1953, Jun. 2018, https://doi.org/10.1109/TCSI.2017.2767204.

[15] C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, and Y. Liang, "REQ-YOLO: A Resource-Aware, Efficient Quantization Framework for Object Detection on FPGAs," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Seaside, CA, USA, Oct. 2019, pp. 33–42, https://doi.org/10.1145/3289602.3293904.

[16] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato, "A Lightweight YOLOv2: A Binarized CNN with A Parallel Support Vector Regression for an FPGA," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, USA, Oct. 2018, pp. 31–40, https://doi.org/10.1145/3174243.3174266.

[17] A. G. Blaiech, K. Ben Khalifa, C. Valderrama, M. A. C. Fernandes, and M. H. Bedoui, "A Survey and Taxonomy of FPGA-based Deep Learning Accelerators," *Journal of Systems Architecture*, vol. 98, pp. 331–345, Sep. 2019, https://doi.org/10.1016/j.sysarc.2019.01.007.

[18] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. F. Nielsen, "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms," *Signal Processing: Image Communication*, vol. 68, pp. 101–119, Oct. 2018, https://doi.org/10.1016/j.image.2018.07.007.

[19] P. Babu and E. Parthasarathy, "Reconfigurable FPGA Architectures: A Survey and Applications," *Journal of The Institution of Engineers (India): Series B*, vol. 102, no. 1, pp. 143–156, Feb. 2021, https://doi.org/10.1007/s40031-020-00508-y.

[20] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, vol. 97, May 2020, Art. no. 103910, https://doi.org/10.1016/j.imavis.2020.103910.