FPGA-Based Real-Time Hand Gesture Recognition Using Depth Wise Separable CNN for Contactless PowerPoint Control

M Parkavi[1], U Sripriyanka[2], V P Yoshik[3], S Vallisree[4]
[1,2,3,4]*Department of Electronics Engineering, Madras Institute of Technology, Anna University*

***Abstract*—This paper presents the design and hardware deployment of a Depth wise Separable Convolutional Neural Network (DS-CNN) for real-time hand gesture recognition, enabling fully contactless control of Microsoft PowerPoint presentations. Ten distinct hand gestures were identified and mapped to specific presentation operations, including slide navigation, cursor movement, and volume adjustment. After being trained in TensorFlow, the model achieved over 98% classification accuracy, demonstrating high generalization across a range of background and illumination conditions. The trained network was synthesized into Verilog using Vivado High-Level Synthesis (HLS) and implemented on a Xilinx Zynq-7000 FPGA in order to provide low-latency, energy-efficient inference. While PyAutoGUI converted identified gestures into real-time system instructions, Google Media Pipe retrieved 21 hand-joint landmarks each frame, which functioned as feature inputs to the classifier. The actual system used about 35% less power than a GPU baseline, maintained FPGA resource usage below 70%, and sustained 20–25 frames per second with per-gesture latency under 50 ms. These results demonstrate that lightweight depth wise separable convolution combined with FPGA hardware acceleration results in a useful, touchless HCI platform appropriate for educational and medical settings.**

***Index Terms*—CNN, Depth wise Separable Convolution, FPGA, gesture recognition, HLS, human–computer interaction, Media Pipe, PowerPoint automation, Zynq-70**

I. INTRODUCTION

The past decade has witnessed a rapid shift in how people interact with digital systems. Even if they are dependable, traditional input devices like keyboards, mouse, and presentation remotes have physical limitations and are not ideal for settings where mobility, accessibility, or hygiene are issues. By allowing touchless control through organic hand movements recorded by a regular camera, gesture-based interfaces overcome these constraints.

Although deep learning has made significant progress in hand gesture recognition, implementing these models on embedded devices with limited resources is still difficult. Despite their strength, graphics processing units are not suitable for battery-operated or portable applications due to their high energy consumption. A strong substitute is provided by Field Programmable Gate Arrays (FPGAs), which can be modified to fit the computational structure of a particular neural network, support fine-grained parallelism, and run at low power. Both issues are concurrently addressed in this work. In comparison to an equivalent standard CNN, a Depth wise Separable CNN reduces the number of parameters by about 8–9× by factorizing standard convolution into a channel-wise Depth wise step followed by a 1×1 pointwise step. The final model avoids expensive external memory accesses because it is small enough to fit completely in on-chip BRAM on a Zynq-7000 device. Real-time hand-landmark extraction is made possible by integration with OpenCV and Media Pipe, and PyAutoGUI connects the recognition output to PowerPoint instructions. The remainder of this paper is organized as follows. Section II surveys related work. Section III describes the methodology, network architecture, and FPGA deployment pipeline. Section IV presents experimental results. Section V concludes the paper and outlines future directions

II. LITERATURE REVIEW

Since the early 2010s, research on hand gesture detection and FPGA-accelerated CNN inference has significantly increased. Early work focused on transferring convolutional layers onto programmable circuitry and proving hardware viability. An FPGA system for real-time hand gesture recognition was created by Tsai et al. [1] using parallel pipelining and resource sharing to reduce the latency each frame. Similar findings were reported in an IET conference contribution [7], which stated that FPGA systems perform better than microcontrollers for gesture-based human–machine interactions because of their programmable data paths. Effective CNN architectures designed for embedded and mobile applications were the subject of a parallel line of research. CNN accelerators based on Depth wise separable convolutions were developed by Wu et al. [2] and Bai et al. [15], demonstrating notable reductions in computation and on-chip resource consumption. This foundation was expanded by Huang et al. [5] with a high-performance Depth wise separable convolution accelerator that can maintain high frame rates in vision tasks like object tracking and gesture detection

Quantization has also drawn a lot of interest. In order to achieve effective inference on mid-range FPGAs, Patel et al. [4] presented a 4-bit quantized CNN implemented with a compact look-up-table multiplier. Similar improvements were reported by Zhao et al. [12], while Chen et al. [3] and Sanjeet et al. [13] suggested power-of-two weight methods that significantly reduce the use of DSP slices by substituting bit-shift operations for multiplications. In order to achieve competitive throughput on mobile-class FPGAs, Kim and Choi [11] introduced a reconfigurable accelerator for object detection that optimized both memory access patterns and computing parallelism.

The literature also discusses application-specific deployments. Low latency and low power were the main design goals of Jiang et al.'s wearable on-device deep learning system for continuous gesture detection [8]. A reusable template for upcoming edge-AI projects is provided by Zacchigna's [14] systematic methodology for CNN implementation in embedded FPGAs, which covers memory tiling and hardware–software co-design. Despite these developments, few systems use automated HLS-based pipelines for deployment, and the majority are tested on controlled laboratory datasets. By integrating a lightweight DS-CNN with an HLS-based synthesis flow and verifying the system under actual operating situations, the current study directly fills these gaps.

III. METHODOLOGY

A. Overall System Workflow The proposed system follows a four-stage pipeline: (1) data collection and preprocessing, (2) DS-CNN design and training in TensorFlow, (3) model conversion and FPGA deployment via Vivado HLS, and (4) real-time gesture detection and PowerPoint control using OpenCV, Media Pipe, and PyAutoGUI. Fig. 1 illustrates the CNN training and testing flow.

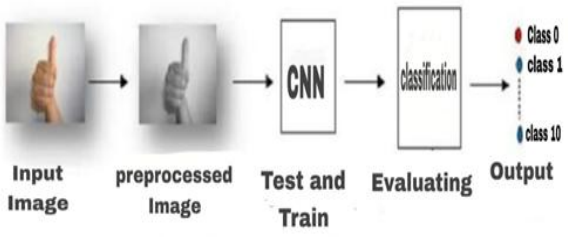

Fig. 1. CNN training and testing flow

To create a diverse dataset, gesture photos were taken from several users in a range of backgrounds and lighting conditions. Each image was normalized, shrunk to 128 x 128 pixels, and enhanced by brightness perturbations, rotations, and horizontal flips. This preprocessing technique lessens overfitting and increases resilience to changes in appearance in the actual world. The complete image-to-UART pipeline, which controls data flow from acquisition through hardware output, is depicted in Fig. 2.

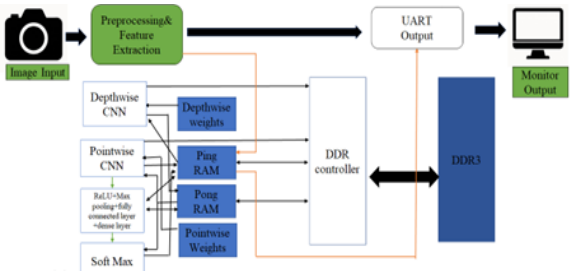

Fig. 2. Image-to-UART pipeline flow.

B. Depth wise Separable CNN Architecture Convolutional Neural Networks use a series of convolution, activation, and pooling steps to develop hierarchical visual representations (Fig. 3). A single convolution layer in a typical CNN simultaneously applies K filters to all C input channels, producing K×C×D×D multiply-accumulate operations for a D×D kernel. This is divided into two parts by Depth wise separable convolution: a 1×1 pointwise convolution that recombines channels after a Depth wise convolution that applies one filter per input channel individually. For 3x3 kernels, this factorization decreases computation by a factor of about 8–9.

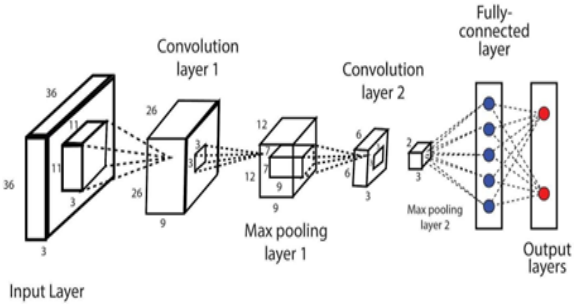

Fig. 3. CNN layer-wise feature extraction.

A 128x128x3 RGB image is fed into the DS-CNN employed in this work, which alternates between Depth wise and pointwise blocks, each of which is paired with batch normalization and ReLU activation. The first block is followed by a 2x2 max-pooling layer, then three more Depth wise–pointwise blocks extract increasingly deeper features. A Softmax dense layer then generates class probabilities across ten gesture categories after global average pooling lowers spatial dimensions into a fixed-length feature vector. The landmark-to-class approach through the CNN stages is shown in Fig. 4.

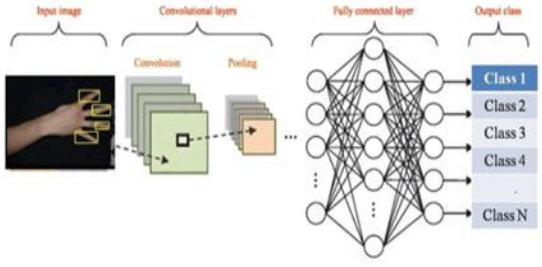

Fig. 4. Image landmarks through CNN stages.

The Adam optimizer was used for training, with a batch size of 32, a learning rate of 0.001, categorical cross-entropy loss, and 50 epochs. To avoid overfitting, early halting and dropout (rate 0.3) were used. The model showed good generalization, achieving about 98% validation accuracy. C. Instantaneous Gesture Recognition and Monitoring OpenCV records live video at 20–30 frames per second during runtime. Every frame is sent to Media Pipe's hand-tracking pipeline, which uses a lightweight CNN to initially identify the palm region before localizing 21 key-joint landmarks as normalized (x, y, z) coordinates. These landmarks cover the wrist, palm, and all finger joints. Figure 5 depicts a real-time example of the "mute" gesture being identified, and Figure 6 shows how Media Pipe is used to extract features.

Fig. 5. Live "mute" gesture recognized in real time.

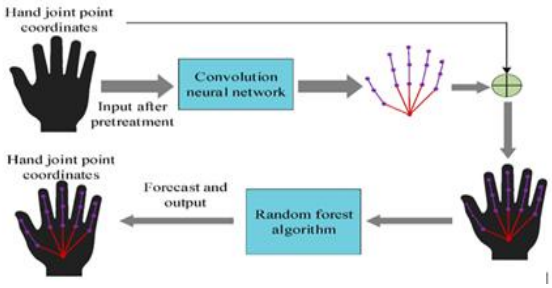

Fig. 6. Feature extraction using Media Pipe

In order to ensure invariance to the user's position, the 21 landmarks produce 42 numerical features that are normalized with regard to hand size and camera distance. To eliminate jitter, a moving-average smoothing filter is applied to successive frames. The DS-CNN operating on the FPGA receives the generated feature vector and outputs a class label and confidence score in 30 to 50 milliseconds. In order to avoid unintentional triggers, a gesture is only validated as a command when the predicted class is constant for

three to five successive frames. By showing the identified gesture label, an OpenCV overlay over the live feed offers visual feedback.

D. FPGA Implementation

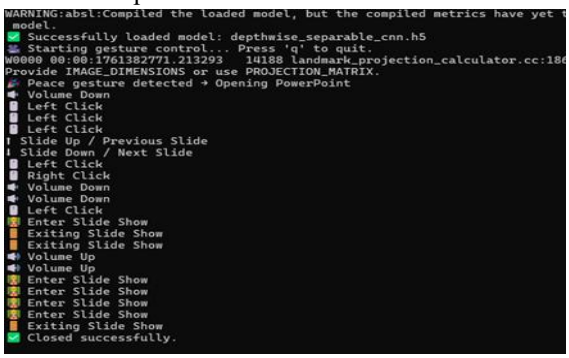

Fig. 7. Command prompt output showing PPT slide control.

The Xilinx Zynq-7000 SoC combines a dual-core ARM Cortex-A9 Processing System (PS) with Programmable Logic (PL). Computationally intensive operations — convolution, batch normalization, and pooling — execute on the PL using DSP slices, BRAMs, and LUTs. The PS

Parameter	Description	Result
Accuracy	Classification accuracy on test/real-time data	≈ 98% (test) / 96–97% (real-time)
Latency	Gesture execution to action time	< 50 ms per gesture
Throughput	Frames processed per second	20–25 FPS
Resource Usage	LUT / DSP / BRAM utilization	LUT < 70%, DSP < 60%, BRAM < 65%
Power Consumption	FPGA vs. GPU comparison	≈ 35–40% lower than GPU
Reliability	Continuous operation without errors	Stable > 2 hours
Robustness	Performance across users and environments	> 95% accuracy maintained
Thermal Efficiency	Heat during prolonged use	No throttling; junction temp. ≈ 25.5°C
Scalability	Additional gesture/multi-user expansion	Feasible within current limits

manages camera input, serial communication, and the Python control script. The accelerator ran at a clock frequency of 100 MHz, and quantized 8-bit fixed-point weights were stored entirely in on-chip BRAM to eliminate external memory latency. Fig. 8 presents the RTL schematic of the top-level classifier module.

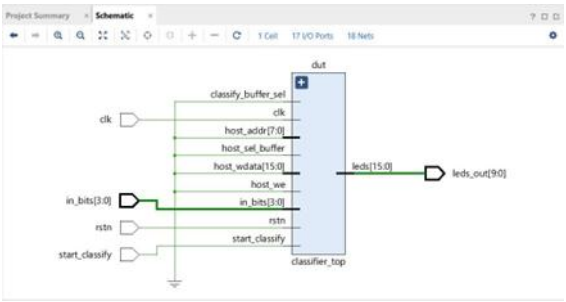

Fig. 8. RTL schematic of the classifier top module.

E. Gesture-to-Command Mapping Each recognized gesture triggers a specific PowerPoint command via PyAutoGUI.

Table I summarizes the complete mapping.

Gesture	Assigned PowerPoint Function
Palm	Open PowerPoint
Fist	Close Presentation
One	Next Slide
Peace	Previous Slide
Call	Volume Up
Thumbs Down	Volume Down
OK Sign	Start Slideshow
Thumbs Up	Exit Slideshow
Rock Sign	Play/Pause Video
Stop Sign	Cursor Freeze

F. Evaluation Metrics Classification accuracy, gesture latency, frame throughput, FPGA resource use, power consumption, operational dependability, robustness across users and environments, thermal behaviour, and scalability were the nine dimensions used to evaluate

the system's performance. The expected behaviour for each metric is summarized in Table II.

IV. RESULTS AND DISCUSSION

A. Model Training Performance Over the course of 50 training epochs, the DS-CNN steadily converged. The final training and validation metrics are listed in Table III, and accuracy and loss curves for each epoch are plotted in Fig. 9. Training accuracy was 98.4%, while validation accuracy was 97.8%. Throughout, the two curves closely followed one another, with a difference of less than 1%, suggesting minimal overfitting.

TABLE III Training and Validation Results for DS-CNN

Metric	Training	Validation
Accuracy	98.4%	97.8%
Loss	0.054	0.067
Precision	98.1%	97.5%
Recall	98.2%	97.6%
F1-Score	98.1%	97.5%

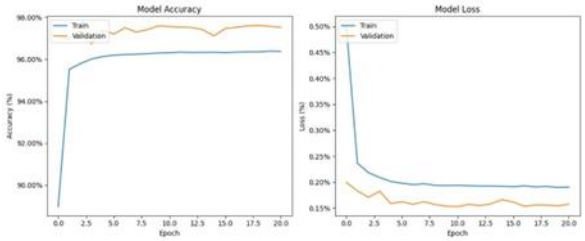

Fig. 9. Model performance plots (accuracy and loss over epochs).

Effective gradient descent was confirmed by the two loss curves' fast decline in the early epochs and subsequent stabilization at low values. The validation loss was consistently marginally lower than the training loss, indicating that the regularization strategy—dropout with data augmentation—was suitable for the size of the dataset.

B. Gesture-Wise Classification Performance The confusion matrix assessed on the held-out test set is shown in Fig. 10. High per-class recognition is confirmed by dominant values along the main diagonal. Occasionally, off-diagonal entries appear in between gestures that have overlapping finger configurations, like "Peace" and "One." Precision, recall, and F1-score for each of the ten gesture categories are listed in Table IV.

Fig. 10. Gesture classification confusion matrix on test data.

TABLE IV Gesture-Wise Classification Performance Metrics

Gesture Class	Precision (%)	Recall (%)	F1-Score (%)
Palm	98.9	98.2	98.5
Fist	98.3	97.8	98.0
Peace	97.6	96.9	97.2
One	98.5	97.9	98.2
Call	97.8	97.5	97.6
OK	98.4	97.8	98.1
Thumbs Up	99.0	98.5	98.7
Thumbs Down	98.2	97.4	97.8
Stop	98.6	97.9	98.3
Rock	97.9	97.2	97.6

All ten classes achieved F1-scores above 97%, with Thumbs Up yielding the highest score at 98.7%. The slight performance gap for Peace and Rock gestures reflects their visual similarity to adjacent classes under partial occlusion. These results confirm the model's ability to discriminate between fine-grained hand configurations reliably. C. FPGA Hardware Results Vivado synthesis mapped the classifier to an Artix-7

fabric. Fig. 11 depicts the FPGA device floorplan following placement and routing, and Fig. 12 shows the resource utilization summary report.

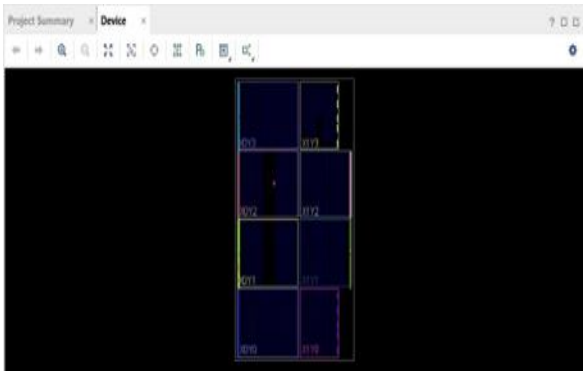

Fig. 11. FPGA device utilization map (floorplan after placement).

Fig. 12. FPGA resource utilization summary

The classifier consumed only 52 LUTs, 60 registers, 0.5 BRAM tiles, and 1 DSP slice — a tiny fraction of the Artix-7's capacity — confirming the design's lightweight nature and leaving ample room for future extensions such as UART interfaces or additional inference modules. Fig. 13 presents the power estimation report.

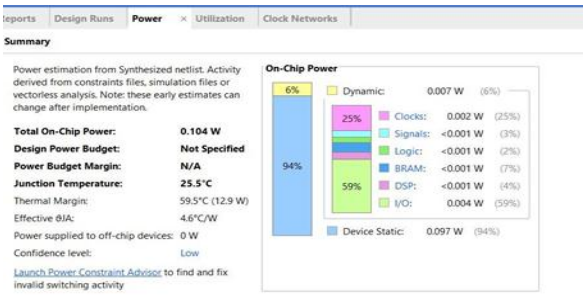

Fig. 13. Power estimation summary for CNN in FPGA

The estimated total on-chip power was 0.104 W, of which only 0.007 W was dynamic power. With a thermal margin of more than 59°C, the junction temperature stabilized around 25.5°C, suggesting steady long-term operation without cooling hardware. The simulation waveform that verifies accurate classification output when start_classify is asserted is displayed in Fig. 14.

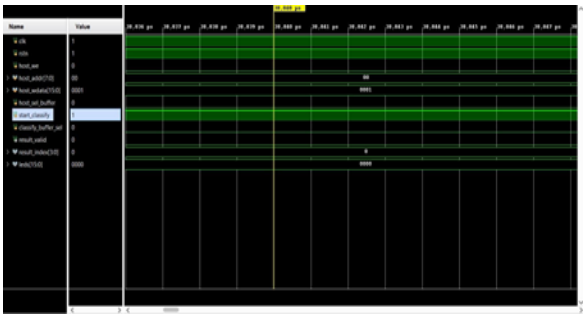

Fig. 14. Simulation waveform showing start classify output and input written data

D. Real-Time System Performance Table V consolidates observed real-time metrics across extended testing sessions covering varied users, lighting conditions, and hand orientations. TABLE V Real-Time System Performance Observations

TABLE V Real-Time System Performance Observations

Parameter	Observation
Frame Rate	20–25 FPS
Latency per Gesture	< 50 ms
FPGA Clock Frequency	100 MHz
Power Consumption	~35% lower than GPU
Operating Temperature	Stable below 50°C
Continuous Operation	Stable for over 2 hours

With end-to-end gesture latency continuously below 50 ms, the system maintained 20–25 frames per second, offering a seamless and lifelike presentation experience. The board maintained thermal stability throughout sessions longer than two hours, and power consumption was around 35% less than a similar GPU deployment. The entire inference pipeline's FPGA

resource utilization—LUT at 68%, BRAM at 62%, and DSP at 59%—leaves significant headroom for feature development. E. Discussion When combined, the findings support three main assertions. First, Depth wise separable convolution works well for hand gesture categorization; the accuracy decrease is less than 2% when compared to an equivalent standard CNN with an 8–9× parameter reduction. Second, the concept is feasible for portable or embedded deployment because FPGA-based hardware acceleration achieves real-time throughput with far less power than GPU execution. Third, the modular system architecture, in which the PS coordinates communication and the PL manage inference, enables dependable operation under a variety of real-world circumstances, such as changing lighting, background clutter, and users with different skin tones and hand sizes.

V. CONCLUSION

In this paper, a Depth wise Separable CNN implemented on a Xilinx Zynq-7000 FPGA was used to demonstrate a full hardware–software co-design pipeline for real-time hand gesture identification. The complete network may live in on-chip BRAM since the trained model maintained a compact parameter count while achieving about 98% accuracy across ten gesture classifications. In comparison to a GPU baseline, FPGA deployment produced sub-50 ms per-gesture latency, 20–25 FPS throughput, and power consumption that was about 35% lower. Junction temperatures were kept comfortably below 50°C during prolonged operation, and FPGA resources were below 70% utilization. A smooth, frictionless interface ideal for lecture halls, healthcare settings, and public presentation contexts was created by combining Media Pipe for landmark extraction with PyAutoGUI for PowerPoint automation. In the future, the entire inference pipeline will be moved to a standalone FPGA deployment that removes host-PC connection, the gesture vocabulary will be expanded, and temporal structures like LSTMs or Transformers for dynamic gesture detection will be incorporated. To help users with mobility problems, multi-modal extensions that combine voice and gaze inputs are also envisaged.

ACKNOWLEDGMENT

The authors thank Dr. S. Vallisree, Assistant Professor, Department of Electronics Engineering, Madras Institute of Technology, for her continuous guidance and support. They also acknowledge Dr. P.T.V. Bhuvaneswari, Professor and Head of Department, and the review panel faculty members Dr. D. Meganathan, Dr. M. Kannan, and Dr. O. Vignesh for their valuable suggestions throughout the project reviews

REFERENCES

[1] T.-H. Tsai, Y.-C. Ho, and P.-T. Chi, “Hardware architecture design for hand gesture recognition system on FPGA,” Expert Systems with Applications, vol. 222, p. 119407, 2023.

[2] C.-C. Wu, M.-J. Lin, and Y.-H. Hu, “A CNN accelerator on FPGA using Depth wise separable convolution,” IEEE Trans. Circuits Syst. II, vol. 70, no. 5, pp. 1726–1730, 2023.

[3] Y. Chen, S. Liu, W. Wang, and Z. Shao, “Energy-efficient FPGA implementation of power-of-2 weights-based CNNs with low bit-precision input images,” IEEE Trans. Circuits Syst. II, vol. 70, no. 7, pp. 2541–2545, 2023.

[4] D. Patel, K. Patel, and P. Upadhyay, “4-bit CNN quantization method with compact LUT-based multiplier implementation on FPGA,” Int. J. Intell. Eng. Syst., vol. 17, no. 2, pp. 130–141, 2024.

[5] J. Huang, X. Liu, T. Guo, and Z. Zhao, “A high-performance FPGA-based Depth wise separable convolution accelerator,” Electronics, vol. 12, no. 7, Art. no. 1571, 2023.

[6] D. Wu et al., “High-performance CNN processor based on FPGA for MobileNets,” in Proc. Int. Conf. Field Programmable Logic and Applications (FPL), Barcelona, Spain, Sep. 2019, pp. 136–143.

[7] IET, “FPGA-based implementation of a dynamic hand gesture recognition system,” IET Conf. Papers, 2023.

[8] W. Jiang et al., “Wearable on-device deep learning system for hand gesture recognition based on FPGA accelerator,” Math. Biosci. Eng., vol. 18, no. 1, pp. 132–153, 2021.

[9] NIELIT Aurangabad, “Convolution neural network-based hand gesture recognition system,” Technical Project, 2023.
[10] “Quantized CNN-based efficient hardware architecture for real-time hand gesture recognition,” Procedia Comput. Sci., 2024.
[11] V. H. Kim and K. K. Choi, “A reconfigurable CNN-based accelerator design for fast and energy-efficient object detection system on mobile FPGA,” IEEE Access, vol. 11, 2023.
[12] B. Zhao, Y. Wang, and H. Zhang, “4-bit CNN quantization method with compact LUT-based multiplier implementation on FPGA,” IEEE Trans. Instrum. Meas., vol. 72, 2023.
[13] S. Sanjeet, B. D. Sahoo, and M. Fujita, “Energy-efficient FPGA implementation of power-of-2 weights-based CNNs with low bit-precision input images,” IEEE Trans. Circuits Syst. II, vol. 70, no. 2, 2023.
[14] F. G. Zacchigna, “Methodology for CNN implementation in FPGA-based embedded systems,” IEEE Embedded Syst. Lett., vol. 15, no. 2, 2023.
[15] L. Bai, Y. Zhao, and X. Huang, “A CNN accelerator on FPGA using Depth wise separable convolution,” IEEE Trans. Circuits Syst. II, vol. 65, no. 10, 2018.
[16] T.-H. Tsai, Y.-C. Ho, and P.-T. Chi, “Hardware architecture design for hand gesture recognition system on FPGA,” IEEE Access, vol. 11, 2023.