Real-Time Edge Face Detection Using a Lightweight Multi-Task Cascaded Convolutional Neural Network with Hardware-Software Co-Design

Original Scientific Paper

Van-Khoa Pham*
Faculty of Advanced Education, Ho Chi Minh City University of Technology and Engineering,
Ho Chi Minh City 700000, Vietnam
khoapv@hcmute.edu.vn

Minh-Duc Phan
Faculty of Advanced Education, Ho Chi Minh City University of Technology and Engineering,
Ho Chi Minh City 700000, Vietnam
21119303@student.hcmute.edu.vn

*Corresponding author

Abstract *– This study presents a hardware-software co-design framework for real-time face detection on resource-constrained edge platforms using a lightweight Multi-Task Cascaded Convolutional Neural Network (MTCNN). The proposed detector replaces selected standard convolutions with depthwise-separable convolutions to reduce computation and parameter count, incorporates Squeeze-and-Excitation (SE) blocks to preserve channel selectivity after compression, and follows an 8-bit integer (INT8)-oriented deployment workflow in which post-training quantization (PTQ) is used for deployment preparation. The optimized detection-oriented cascade is deployed on the AMD Kria KV260, where P-Net, R-Net, and O-Net are implemented as Vitis HLS kernels and integrated through Advanced eXtensible Interface (AXI)-based communication between the processing system (PS) and programmable logic (PL). In this lightweight configuration, O-Net performs final face classification and bounding-box regression only, without the landmark branch of the original full MTCNN. Under the reported evaluation settings, the lightweight model reduces the parameter count from 495,850 to 197,202 (60.2%) and achieves 96.3% AP under the WIDER FACE validation protocol, compared with 97.8% for the original MTCNN. In the current KV260 implementation, the complete deployed system sustains 30 frames per second (FPS), while standalone FPGA kernels at 100 MHz achieve latencies of 0.035 ms, 0.810 ms, and 3.525 ms for P-Net, R-Net, and O-Net, respectively. These results support the feasibility of the proposed framework for real-time edge face detection under resource constraints.*

Keywords: *Edge Computing, Face Detection, MTCNN, Hardware Accelerator, Depthwise-Separable Convolution, Squeeze-and-Excitation*

Received: March 6, 2026; Received in revised form: April 26, 2026; Accepted: May 26, 2026

1. INTRODUCTION

Face detection is a fundamental front-end task in many vision applications, including surveillance, access control, driver monitoring, and human-computer interaction [1]. As these applications move toward embedded and edge platforms, deployment must satisfy not only detection accuracy but also strict constraints on latency, memory footprint, and power consumption. Consequently, lightweight and hardware-aware model design has become essential for real-time operation on resource-constrained systems. Among existing approaches, the Multi-Task Cascaded Convolutional Neural Network (MTCNN) remains a strong baseline because of its three-stage coarse-to-fine architecture, namely P-Net, R-Net, and O-Net [2]. However, the original MTCNN is not well suited to compact edge deployment because repeated multi-scale processing and cascaded inference impose substantial demands on computation, memory bandwidth, and on-chip storage. Although prior studies have accelerated MTCNN and related convolutional neural network (CNN) models on field-programmable gate array (FPGA) platforms [3-6], most have focused primarily on runtime acceleration rather than jointly optimizing network architecture, quantization, and hardware mapping. In this context, high-level synthesis (HLS) provides a practical implementation path for heterogeneous systems [7-9].

Recent studies on lightweight CNNs provide several effective directions for embedded deployment. Depthwise-separable convolution reduces arithmetic complexity and parameter count in compact vision networks [10], [11], whereas Squeeze-and-Excitation (SE) blocks help preserve channel discrimination under aggressive compression [12]. Model quantization further supports low-power inference by reducing memory usage and enabling efficient integer arithmetic [13-16]. From the hardware perspective, prior FPGA studies have shown that the benefits of lightweight convolutions depend strongly on dataflow design, workload partitioning, and memory reuse [17-21]. Therefore, efficient edge deployment requires a co-design strategy in which model compression and hardware implementation are developed jointly rather than separately.

Despite these advances, a gap remains between algorithm-level optimization and deployment-level realization for face detection on resource-constrained edge platforms. Prior studies often address only one aspect of the problem, such as model acceleration, network compression, or FPGA implementation, without jointly considering the interaction among architectural simplification, quantization readiness, and hardware mapping in the deployed system [3-6], [13-21]. In contrast to recent studies on related face-analysis tasks such as deepfake detection [22], the present work focuses specifically on deployment-oriented lightweight face detection, where real-time performance depends jointly on detector compactness, cascade efficiency, and hardware mapping on an edge FPGA platform. As a result, many reported solutions either retain accuracy at the cost of higher hardware complexity or improve speed without sufficiently addressing memory traffic, power efficiency, and workload imbalance across the MTCNN cascade. This limitation is especially critical in edge face detection, where real-time performance depends not only on arithmetic reduction but also on efficient coordination among computation, data movement, and control overhead [17-21].

To address these challenges, this study proposes a unified hardware-software co-design framework for lightweight MTCNN-based face detection on resource-constrained edge platforms. At the network level, selected standard convolutions are replaced with depthwise-separable convolutions to reduce computation and parameter count, while SE blocks are introduced to preserve channel discrimination under strong compression. At the deployment level, the framework follows an INT8-oriented workflow in which post-training quantization (PTQ) serves as the default path and quantization-aware training (QAT) is considered only when needed. The optimized P-Net, R-Net, and O-Net are implemented as Vitis HLS kernels on the AMD Kria KV260 and integrated through AXI-based communication between the Processing System (PS) and Programmable Logic (PL). By jointly considering model structure, quantization readiness, and hardware mapping, the framework improves model compactness and runtime efficiency while maintaining competitive detection performance under the reported evaluation settings.

The rest of the study is organized as follows. Section 2 reviews the MTCNN pipeline, edge-deployment challenges, and the motivation for the redesign. Section 3 presents the lightweight MTCNN architecture, the INT8-oriented deployment workflow, and the hardware-software co-design on the AMD Kria KV260. Section 4 reports model-level and system-level results, including accuracy, throughput, latency, resource utilization, and power. Section 5 concludes the study and discusses future work.

2. FACE DETECTION WITH MTCNN

MTCNN is a cascaded face-detection framework composed of three subnetworks, namely P-Net, R-Net, and O-Net, in which candidate face regions are progressively generated, refined, and classified in a coarse-to-fine manner [2]. P-Net performs dense scanning over the input image to generate candidate face proposals, R-Net refines and filters these proposals, and O-Net performs final face classification and bounding-box regression. This cascaded structure enables efficient multi-face detection because a large number of background regions are rejected in the early stages before more computationally expensive refinement is applied in the later stages. Fig. 1 illustrates the overall pipeline of the original MTCNN framework.

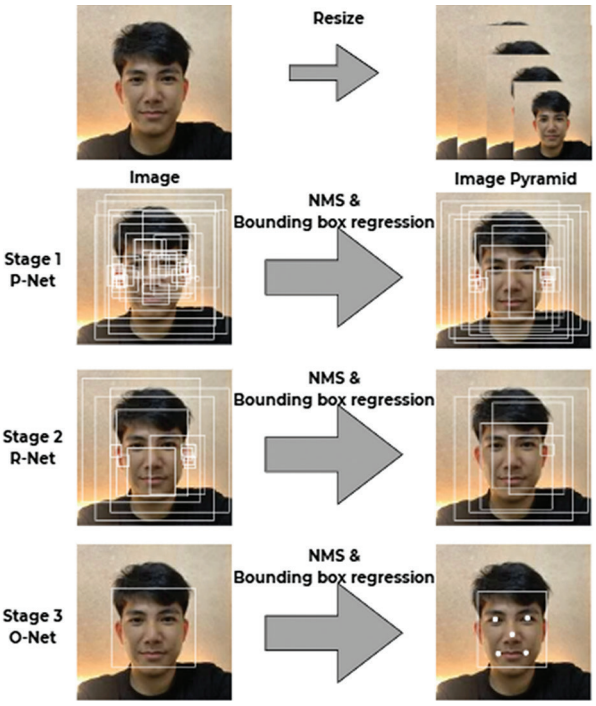


Fig. 1. Pipeline of the cascaded framework based on a three-stage MTCNN

Given an input image I, MTCNN constructs a geometric image pyramid to support face detection across multiple scales. Let $I^{(l)}$ denote the image at pyramid level l, obtained by repeatedly resizing the original image using a scale factor γ, as expressed in (1):

$$I^{(l)} = Resize(I,s_l), s_l = \gamma^l, \gamma = 0.707, \quad l = 0,1,\ldots,L, \tag{1}$$

P-Net scans each pyramid level to generate candidate face boxes, followed by non-maximum suppression (NMS). The retained candidates are then cropped and resized to 24×24 for R-Net refinement, and the surviving proposals are resized to 48×48 for final classification and bounding-box regression in O-Net. A final NMS stage is subsequently applied to remove duplicate detections.

Although this cascaded design is effective, the original MTCNN is not well suited to edge deployment. First, repeated multi-scale processing substantially increases the computational burden of P-Net because the detector must evaluate many windows across multiple image scales. Second, the standard convolution operations used throughout the cascade are computationally expensive and require significant memory bandwidth. Third, the workload is unevenly distributed across stages, since O-Net contains the largest number of parameters and contributes the greatest latency. In addition, direct deployment on low-power platforms is constrained by the cost of floating-point inference, limited on-chip memory, and the sensitivity of cascaded refinement to quantization error [13-16]. These limitations motivate the lightweight redesign and hardware-software co-design strategy presented in Section 3.

3. PROPOSED EDGE-ORIENTED MTCNN DESIGN

In the lightweight configuration deployed in this study, the MTCNN cascade is redesigned for real-time face detection and cropped-face extraction on the target edge platform. The proposed method introduces a detection-oriented O-Net, filter reduction, depthwise-separable convolutions, SE-based channel recalibration, and an INT8-oriented deployment workflow, followed by hardware-software co-design on the AMD Kria KV260.

3.1. DETECTION-ORIENTED LIGHTWEIGHT REDESIGN

The proposed framework preserves the original P-Net/R-Net/O-Net cascade structure while redesigning each stage to reduce model complexity. In the deployed lightweight configuration, the optimized O-Net performs only final face classification and bounding-box regression, without the landmark-prediction branch of the original MTCNN. This modification is more consistent with the target application of real-time face detection and cropped-face extraction, while also reducing parameter count and inference cost.

To enable efficient edge deployment, the original cascade is redesigned through filter reduction, the replacement of selected standard convolutions with depthwise-separable convolutions, and SE-based channel recalibration. In P-Net, the first convolution is reduced from 10 to 8 filters, followed by depthwise-separable blocks with 16 and 32 output channels. In R-Net, the early convolutional widths are reduced from 28 and 48 to 16 and 32 channels, respectively, and the subsequent feature-extraction path is reorganized using depthwise-separable convolutions. In O-Net, the first convolution is reduced from 32 to 16 filters, and the subsequent feature extractor uses depthwise-separable blocks with 32, 48, and 64 output channels. SE blocks are introduced in all three subnetworks to mitigate the representational loss caused by structural compression.

Assuming identical output feature-map dimensions and neglecting bias operations and implementation overhead, the ratio of multiply-accumulate operations required by depthwise-separable convolution to those of standard convolution, with a $K\times K$ kernel, M input channels, and N output channels, is given by (2):

$$\rho_{DS} = \frac{K^2M+MN}{K^2MN} = \frac{1}{N}+\frac{1}{K^2}. \tag{2}$$

This factorization decomposes the original K^2MN coupling into a depthwise spatial term and a pointwise channel-mixing term, thereby reducing multiply-accumulate cost while preserving the effective receptive field. Fig. 2 illustrates the lightweight P-Net architecture used in the proposed cascade. A 12×12×3 input is processed by a convolutional stem and two depthwise-separable convolution blocks, after which the feature tensor is recalibrated by an SE branch before entering the classification and bounding-box prediction heads.

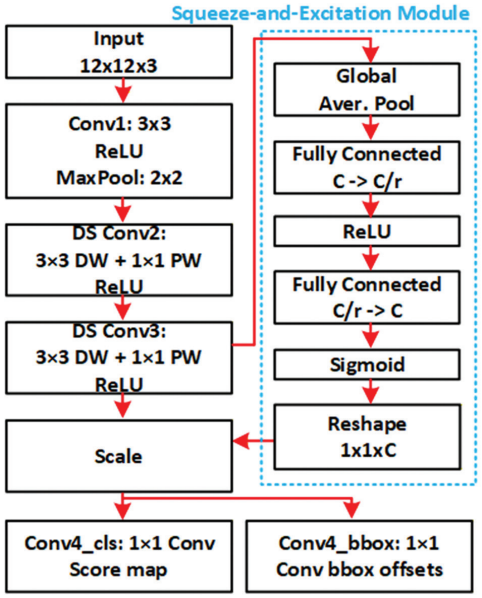


Fig. 2. Proposed lightweight P-Net architecture with depthwise-separable convolution and SE channel recalibration

3.2. SE-BASED CHANNEL RECALIBRATION AND INT8-ORIENTED DEPLOYMENT

SE blocks are introduced to compensate for the representational loss caused by aggressive slimming [12]. Let $U\in R^{H\times W\times C}$ be the output tensor of a compact feature block. The SE transformation is defined as

$$\tilde{U}_c(i,j) = s_c U_c(i,j), s = \sigma(W_2\delta(W_1 z)), \tag{3}$$

where z is the C-*dimensional* channel descriptor obtained by global average pooling, $\delta(\cdot)$ denotes the Rectified Linear Unit (ReLU) activation function, and $\sigma(\cdot)$ represents the sigmoid activation function [12]. The matrices W_1 and W_2 correspond to the two fully connected layers of the excitation mechanism, with dimensions $(C/r) \times C$ and $C \times (C/r)$, respectively, where r denotes the channel-reduction ratio. Equation (3) formalizes the role of SE in this work: it adaptively reweights channels after compression so that compact layers retain useful channel selectivity. In Fig. 2, this process is represented by the squeeze path, bottleneck excitation, sigmoid gating, and channel-wise scaling preceding the output heads. In the proposed design, SE acts as an accuracy-recovery mechanism rather than as a compression method. This role is important because feature degradation in P-Net can propagate through the cascade and reduce proposal quality in later stages.

The optimized model is prepared for an INT8-oriented inference path to reduce storage cost, memory traffic, and arithmetic complexity on the embedded target. Using uniform affine quantization,

$$x_q = clip(\lfloor \frac{x}{s_x} \rceil + z_x, q_{min}, q_{max}), \hat{x} = s_x(x_q - z_x), \quad (4)$$

where x denotes the original real-valued tensor, x_q represents its quantized integer representation, and $\hat{x}$ denotes the reconstructed approximation obtained through dequantization. The parameter $s_x > 0$ is the quantization scale, z_x is the integer zero-point, and q_{min} and q_{max} define the permissible quantization range. The rounding operator maps values to the nearest integer, while the clipping function restricts quantized values to the supported integer range. The deployment workflow uses uniform affine quantization, where the real-valued tensor is represented by an INT8 tensor together with a scale factor and a zero-point.

In this study, post-training quantization (PTQ) is used as the default quantization strategy for deployment preparation, whereas quantization-aware training (QAT) is considered only when PTQ causes unacceptable performance degradation. Unless otherwise stated, the reported parameter counts refer to the structurally optimized lightweight MTCNN architecture. Post-training quantization (PTQ) changes the numerical representation of model weights and activations without altering the number of model parameters. The KV260 latency and throughput results presented in Section 4 correspond to the current HLS-based hardware implementation and are reported separately from the model-level quantization preparation. This distinction clearly separates model-level quantization readiness from the arithmetic mode of the current hardware implementation.

Taken together, (1)-(4) define the algorithmic basis of the proposed lightweight MTCNN. Equation (1) defines the image-pyramid scaling process, whose scaling factor controls the number of processed pyramid levels, Equation (2) lowers convolutional complexity, Equation (3) restores channel discrimination after slimming, and Equation (4) supports low-precision deployment.

This combination is well suited to real-time edge face detection because it addresses the major bottlenecks jointly rather than optimizing only the network backbone.

3.3. HARDWARE-SOFTWARE CO-DESIGN ON THE AMD KRIA KV260

Fig. 3 shows the deployment of the optimized MTCNN cascade on the AMD Kria KV260, which is based on the Zynq UltraScale+ multiprocessor system-on-chip (MPSoC). Input video frames are streamed to the platform, where the customized three-stage cascade performs face detection before forwarding the detected face regions to downstream processing.

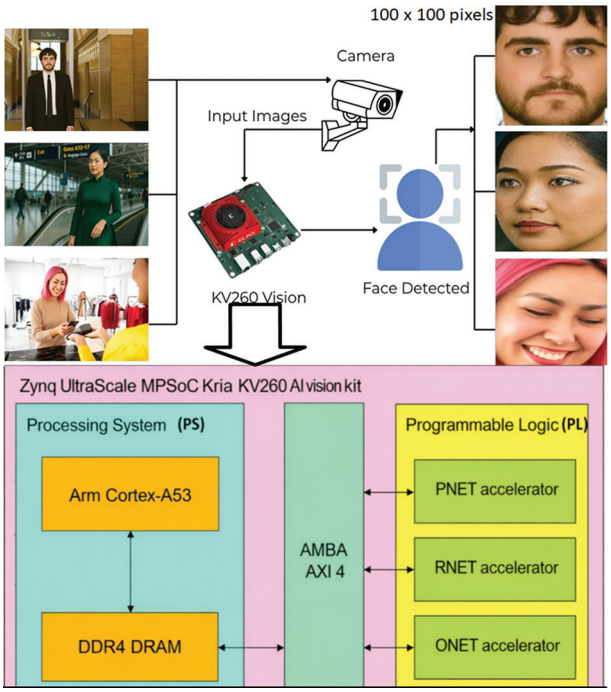


Fig. 3. System architecture of the proposed face detection framework on the Kria KV260 Vision Kit

At the system level, the Processing System (PS) handles frame I/O, control scheduling, memory management, image resizing, image-pyramid generation, inter-stage proposal cropping and resizing, and post-processing tasks such as NMS, whereas the Programmable Logic (PL) implements P-Net, R-Net, and O-Net as dedicated Vitis HLS kernels. The PS and PL communicate through AXI4 over shared double data rate (DDR) memory, thereby keeping control-intensive tasks in software and offloading convolution-dominant computation to hardware. This design is consistent with prior FPGA-based inference architectures, in which throughput depends on the coordinated optimization of computation and data movement. In the PL, pipelining, loop unrolling, and dataflow increase concurrency, whereas FIFO stream buffers allow adjacent stages to overlap at the cost of additional BRAM usage. Because P-Net is compact, its weights can be stored on-chip, whereas the larger R-Net and O-Net weights must be fetched from external DDR, making memory bandwidth and transfer scheduling critical to end-to-end performance.

Assuming that computation and memory transfers can overlap almost completely, while control overhead remains additive, the latency of each processing stage can be approximated by

$$T_t \approx \max(T_{t,\,comp}, T_{t,\,mem}) + T_{t,\,ctrl} \quad (5)$$

where $T_{t,\,comp}$ is the compute time, $T_{t,\,mem}$ is the memory-transfer time, and $T_{t,\,ctrl}$ is the control overhead associated with PS-side orchestration. Equation (5) shows that arithmetic reduction alone is insufficient for real-time edge deployment. Even when the network is structurally compressed, the achievable system throughput remains bounded by memory movement and control overhead if these terms are not concurrently optimized. This observation is fully aligned with the study's target problem: real-time embedded face detection is governed jointly by algorithmic efficiency and system-level dataflow behavior.

4. EXPERIMENTAL EVALUATION AND RESULTS

4.1. EXPERIMENTAL SETUP

This section evaluates the framework at both model and system levels for real-time edge deployment. At the model level, the evaluation assesses whether the lightweight design and INT8-oriented optimization reduce computational and memory cost while preserving competitive detection accuracy. The lightweight MTCNN is also compared with the original MTCNN on the WIDER FACE validation set, a standard face detection benchmark with large variations in scale, pose, occlusion, and illumination [23]. The evaluation uses two datasets: the WIDER FACE validation set for comparison with the original MTCNN and prior detectors, and an in-house dataset representing the target edge deployment setting. The in-house dataset contains over 8,000 annotated images with more than 15,000 face instances and five-point landmark labels; however, in the present detection-oriented deployment, the lightweight model is trained and evaluated only for face classification and bounding-box regression. Training samples follow the standard MTCNN three-class strategy: intersection over union (IoU) < 0.3 for negatives, 0.4 ≤ IoU < 0.65 for part samples, and IoU ≥ 0.65 for positives. The dataset is split into 80% training, 15% validation, and 5% testing. Images are normalized as (x−127.5)×0.0078125, with a batch size of 32 and 4 workers. Weights are initialized with Xavier uniform, biases with 0.1, and training uses Adam. Each customized subnetwork is trained for 29 epochs, with the best checkpoint chosen from validation results. During inference, cascade thresholds are set to 0.6/0.6/0.7 for P-Net/R-Net/O-Net, and the image-pyramid scaling factor is 0.707. These values are used as the evaluation settings in this study.

At the system level, the optimized cascaded detector is deployed on the AMD Kria KV260 platform, where P-Net, R-Net, and O-Net are implemented as Vitis HLS kernels and connected through AXI-based communication between the processing system and programmable logic. For comparison, the same optimized model is also evaluated on an Intel i7 CPU and an NVIDIA GTX 1060 GPU, helping distinguish gains from model optimization and hardware acceleration. Deployment feasibility is further assessed through post-implementation hardware utilization and power analysis, including look-up table (LUT), flip-flop (FF), digital signal processing slice (DSP) and RAM usage, with power estimated using Vivado tools. Kernel-level latency is also measured for each cascade stage to show the computational distribution across P-Net, R-Net, and O-Net. Together, these results evaluate detection accuracy, model compactness, throughput, resource utilization, and power consumption, with the settings and metrics summarized in Table 1.

Table 1. Experimental settings, target platforms, and evaluation metrics used in this study

Category	Item	Value
Dataset	Training corpus	In-house face dataset (>8,000 images)
	Benchmark dataset	WIDER FACE validation set
	Total annotations	>15,000 annotated faces with five landmarks per face
Dataset split	Train/Validation/Test	80% / 15% / 5%
Data loading	Batch size	32
	Number of workers	4
Input size	P-Net/R-Net/O-Net	12×12×3 / 24×24×3 / 48×48×3
Inference thresholds	P-Net/ R-Net/ O-Net	0.6 / 0.6 / 0.7
Image pyramid	Scaling factor	0.707
Quantization	PTQ calibration set	Up to 500 images per model
Processor platform	Intel Core	i7-14650HX @ 2.2 GHz
FPGA platform	AMD Kria KV260	Zynq UltraScale+ MPSoC
FPGA clock	Kernel frequency	100 MHz
Evaluation metrics	Model-level	AP, classification accuracy, parameter count
	System-level	Latency, FPS, LUT/FF/BRAM/DSP/URAM, Power consumption

4.2. MODEL-LEVEL RESULTS

Fig. 4 shows that the proposed lightweight MTCNN improves model compactness while maintaining effective detection performance. As shown in Fig. 4(a), the redesigned cascade reduces the parameter count from 6,632 to 1,846 in P-Net, from 100,178 to 79,174 in R-Net, and from 389,040 to 116,182 in O-Net, with the largest reduction in O-Net. As a result, the total parameter count decreases from 495,850 to 197,202. Fig. 4(b) shows that this compression causes only a modest drop in detection performance. Although stage-wise classification accuracy decreases after optimization, for example, from 98.4% to 94.8% for O-Net, the proposed model still achieves 96.3% AP under the WIDER FACE validation protocol, compared with 97.8% AP for the original MTCNN. In addition, increasing the image-pyramid scaling factor from 0.6 to 0.707 produces a more densely sampled image pyramid and may increase the number of pyramid levels and the computational workload of P-Net, depending on the input-image dimensions and stopping criterion. Overall, these results indicate that the proposed refinements, including depthwise-separable convolutions, substantially reduce model complexity and memory overhead with only limited impact on detection capability.

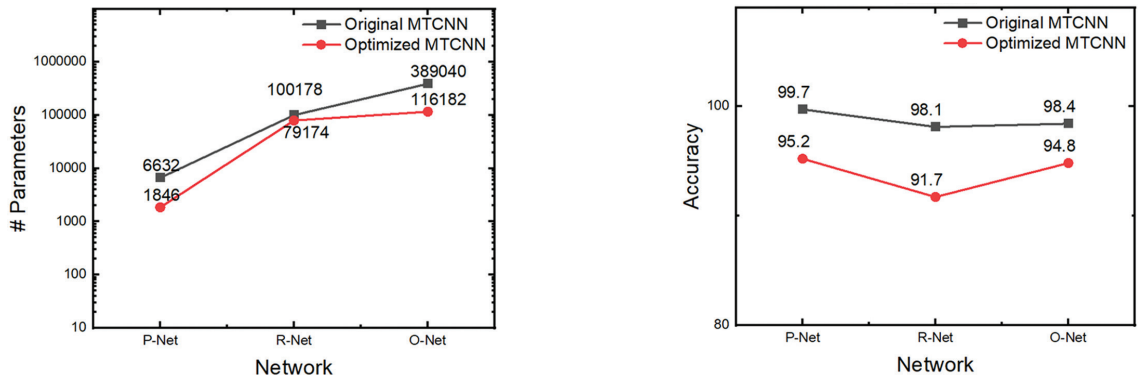


Fig. 4. Comparison of the original and optimized MTCNN across cascade stages. (**a**) Parameter counts for P-Net, R-Net, and O-Net. (**b**) Stage-wise accuracy, showing a modest performance reduction after optimization

Fig. 5 provides a qualitative comparison of the detector with and without the SE module. The SE-enabled configuration produces cleaner detections and more stable bounding-box placement across different scenes. These qualitative results suggest that SE helps preserve feature selectivity after model compression. In the non-SE configuration, shown on the left, the three-stage cascade follows the standard MTCNN progression from dense P-Net candidate proposals to R-Net refinement and the final O-Net output. As illustrated, the early stage generates multiple candidate boxes, including several redundant or off-target responses, which are only gradually suppressed in the later stages.

By contrast, the SE-enabled results shown on the right exhibit cleaner detections and more stable bounding-box placement across images with different viewpoints, backgrounds, and scene compositions. This qualitative difference indicates that SE helps preserve channel-wise feature selectivity after structural slimming with depthwise-separable convolutions, thereby improving proposal quality in the early cascade stages and reducing error propagation to downstream refinement. Overall, the figure provides visual evidence that the addition of SE helps the compressed detector maintain reliable face localization while remaining suitable for real-time edge-oriented deployment.



Fig. 5. Qualitative comparison of face-detection results with and without the SE module in the proposed lightweight MTCNN accelerator. Left: non-SE cascade outputs; right: SE-enabled outputs with more stable face localization.

4.3. SYSTEM-LEVEL RESULTS ON THE KV260

While the previous results demonstrate the proposed system's advantages at the model level, the following analysis examines its system-level behavior to evaluate deployment feasibility on the target edge platform. As shown in Fig. 6(a), the accelerator is implemented in Vivado as a Zynq UltraScale+ MPSoC design, where the P-Net, R-Net, and O-Net kernels are connected to the processing system through AXI interconnect and SmartConnect. Fig. 6(b) shows that the design fits within the target device resources, although it is memory-intensive, using 126.5/144 BRAMs, 502/1,248 DSPs, 91,292/117,120 LUTs, 115,497/234,240 FFs, and 16/64 URAMs. Fig. 6(c) reports an estimated total on-chip power of 3.853 W, including 3.544 W dynamic and 0.309 W static power. Most dynamic power is consumed by the processing system at 2.419 W, while the programmable logic uses 1.125 W.

Table 2 provides the stage-wise hardware resource utilization of the optimized kernels. P-Net is the lightest stage, whereas O-Net is the most memory-intensive. Specifically, P-Net uses 24,312 LUTs, 26,764 FFs, 5 BRAMs, and 162 DSPs. R-Net uses 28,959 LUTs, 33,700 FFs, 35.5 BRAMs, and 191 DSPs. O-Net uses 25,608 LUTs, 37,178 FFs, 86 BRAMs, and 149 DSPs. The integrated PS-PL system uses 91,292 LUTs, 115,497 FFs, 126.5 BRAMs, and 502 DSPs. These system-level values are not expected to equal the simple sum of the three kernel-level values because the complete design also includes AXI interconnect, PS-PL control logic, memory interfaces, and other integration overhead. Overall, the results confirm that the proposed PS-PL integration is feasible on the KV260, although interconnect and control overhead must be considered.

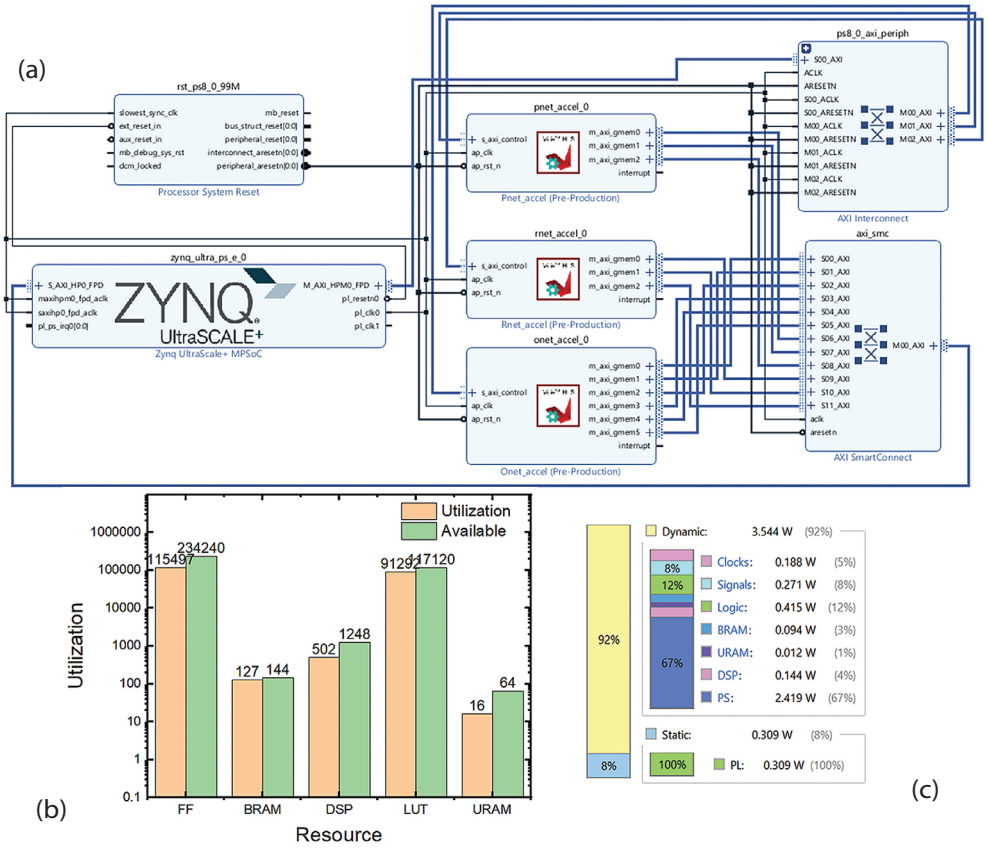


Fig. 6. Integrated design and implementation results of the proposed MTCNN accelerator (**a**) Vivado block design showing the MPSoC, P-Net, R-Net, and O-Net kernels, and the AXI interconnect. (**b**) Overall hardware resource utilization of the integrated PS-PL implementation (**c**) Vivado-estimated on-chip power breakdown

Table 2. Stage-wise hardware resource utilization of the optimized MTCNN kernels on the AMD Kria KV260

Stage	LUT	FF	BRAM	DSP
Optimized P-Net	24,312	26,764	5	162
Optimized R-Net	28,959	33,700	35.5	191
Optimized O-Net	25,608	37,178	86	149
PS-PL system	91,292	115,497	126.5	502

Note: The first three rows report standalone kernel resource usage, whereas the last row reports the integrated PS-PL implementation on the KV260.

Beyond the model-level gains, the results further show that the proposed hardware-software co-design translates these improvements into practical runtime benefits at the system level. On the AMD Kria KV260/ARM platform, the complete implementation sustains 30 FPS as an integrated end-to-end deployment result for the cascaded detector. This figure should be interpreted as system-level throughput, including preprocessing, inter-stage data movement, kernel invocation,

and post-processing, rather than as the latency of an isolated CNN kernel. For reference, the same optimized model achieves approximately 50 FPS on an Intel Core i7 CPU and about 70 FPS on an NVIDIA GTX 1060 GPU, confirming that the redesigned network remains efficient across heterogeneous platforms.

A more detailed view is provided in Table 3, which shows that the impact of the proposed hardware optimization is not uniform across the cascade but instead increases with stage complexity. For clarity, the hardware acceleration factors reported in Table 3 are measured relative to the corresponding unoptimized FPGA implementation, not relative to CPU software execution. Accordingly, the reported speedups of 1.43×, 5.00×, and 101.56× for P-Net, R-Net, and O-Net, respectively, are computed from the ratio between the original FPGA latency and the optimized FPGA latency of each stage. In particular, the 101.56× speedup for O-Net corresponds to the reduction from 358 ms to 3.525 ms on FPGA, whereas the CPU latency values are included only for runtime reference. This pattern suggests that the proposed architectural refinements and FPGA-level parallel execution strategies yield only modest benefits for lightweight stages, whereas their effect becomes much more significant as the computational burden increases.

Table 3. Stage-wise standalone kernel latency before and after hardware optimization on the AMD Kria KV260

Network	Latency (ms)			Hardware acceleration improvement
	Software on Intel Core i7-14650HX @ 2.2 GHz	Hardware on FPGA @ 100 MHz		
		Original model	Optimized model	
P-Net	1.2	0.050	0.035	1.43×
R-Net	1.5	4.047	0.810	5.00×
O-Net	2.6	358	3.525	101.56×

Note: Hardware acceleration improvement is measured relative to the unoptimized FPGA implementation of the same network stage, rather than relative to CPU execution.

This stage-dependent behavior is also reflected in the latency results. For P-Net, the original FPGA implementation already outperforms the CPU baseline, reducing latency from 1.2 ms to 0.050 ms, while the optimized version provides only a small further improvement to 0.035 ms. In contrast, R-Net is initially slower on the unoptimized FPGA design, with a latency of 4.047 ms versus 1.5 ms on the CPU, but optimization reduces it to 0.810 ms, allowing the FPGA to surpass the CPU. The largest improvement is seen in O-Net, where latency drops sharply from 358 ms to 3.525 ms. Although this is still slightly higher than the CPU runtime of 2.6 ms, it significantly reduces the main bottleneck in the cascade. This is especially important for end-to-end deployment, since accelerating the slowest stage has the strongest effect on real-time feasibility. However, these values represent standalone kernel latency, so total system latency will be higher once data transfer and integration overhead are included.

4.4. CONTEXTUAL COMPARISON WITH REPRESENTATIVE DETECTORS

Table 4 compares representative face detectors in terms of accuracy and average throughput under the reported edge-oriented setting. Among the classical baselines, Haar Cascade achieves approximately 72% accuracy at 25 FPS on the CPU, whereas DLIB HOG improves accuracy to approximately 82% but runs more slowly at approximately 20 FPS [24], [25]. RetinaFace provides a stronger deep-learning baseline, reaching approximately 96% accuracy, although only a CPU throughput of 20 FPS is reported in this comparison [26]. In this study, the original MTCNN achieves the highest detection accuracy of 97.8%. The standalone unoptimized FPGA kernel latencies reported in Table 3 sum to 362.097 ms (0.050 + 4.047 + 358 ms). This calculated value is not a measured end-to-end frame rate, because it does not account for processing across image-pyramid levels, multiple candidate proposals, data transfers, or other system-level overhead. The original MTCNN throughput on the KV260 is therefore not reported as an end-to-end FPS measurement. The original model achieves a reported CPU throughput of 5.3 FPS. By contrast, the optimized MTCNN maintains near-baseline accuracy at 96.3% while increasing throughput to 30 FPS on the KV260, 50 FPS on the CPU, and 70 FPS on the GPU. Overall, the results reflect the expected accuracy-efficiency trade-off: classical lightweight methods are faster but less accurate, whereas deep-learning methods are generally more accurate but more computationally expensive. However, this comparison is contextual only, because the reported values were obtained using different datasets, hardware platforms, and evaluation protocols.

Table 4. Contextual accuracy-throughput comparison of face detection methods. The original and optimized MTCNN results were measured in this study under the settings in Table 1, while the Haar Cascade, DLIB HOG, and RetinaFace results are literature-based values summarized from [24-26]

Method	Accuracy (%)	Avg. FPS		
		KV260	CPU	GPU
Original MTCNN (original deployment)	97.8	N/R	5.3	N/A
Optimized MTCNN (optimized deployment)	96.3	30	50	70
Haar Cascade [24]	~72	N/A	25	N/A
DLIB HOG [25]	~82	N/A	~20	N/A
RetinaFace [26]	~96	N/A	20	N/A

Note: N/R indicates that end-to-end throughput was not reported. The standalone FPGA kernel latencies are presented in Table 3 but do not establish end-to-end frame rate.

5. CONCLUSION

This study presented a hardware-software co-design framework for the deployment of a lightweight, detection-oriented MTCNN-based face detector on resource-

constrained edge platforms. The proposed design combines filter reduction, depthwise-separable convolutions, SE blocks, and an INT8-oriented workflow for deployment on the AMD Kria KV260. Under the reported evaluation settings, the optimized model reduces the parameter count from 495,850 to 197,202, achieves 96.3% AP on the WIDER FACE validation set, and sustains 30 FPS on the KV260. Within the scope of the present deployment, these results indicate that the proposed framework is a feasible solution for real-time edge face detection with reduced model complexity and practical runtime efficiency. Future work will focus on further reducing memory traffic and control overhead, improving hardware-aware quantization support in the deployed implementation, and extending the framework toward more complete MTCNN functionality when required by the target application.

ACKNOWLEDGMENT

The authors sincerely thank Nam-Phat Tran and Ngoc-Tram Le Nguyen for their support and assistance with part of the verification work for this study.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

FUNDING

This research received no external funding.

DECLARATION OF THE USE OF AI TOOLS

The authors used Grammarly for language editing. All AI-assisted content was reviewed and edited by the authors, who take full responsibility for the integrity and accuracy of this work. No AI tool is listed as an author, and AI tools are not cited as primary sources.

REFERENCES

[1] A. Kumar, A. Kaur, M. Kumar, "Face Detection Techniques: A Review", Artificial Intelligence Review, Vol. 52, No. 2, 2019, pp. 927-948. doi: 10.1007/s10462-018-9650-2

[2] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, "Joint Face Detection and Alignment Using Multi-Task Cascaded Convolutional Networks", IEEE Signal Processing Letters, Vol. 23, No. 10, 2016, pp. 1499-1503. doi: 10.1109/LSP.2016.2603342

[3] Z. Bao, J. Wang, "Design of Real-Time Face Detection FPGA System Based on MTCNN", Proceedings of the 6th International Conference on Artificial Intelligence and Pattern Recognition (AIPR '23), Xiamen, China, 22-24 September 2023, pp. 498-505. doi: 10.1145/3641584.3641658

[4] L.-H. Ma, H.-Y. Fan, Z.-M. Lu, D. Tian, "Acceleration of Multi-Task Cascaded Convolutional Networks", IET Image Processing, Vol. 14, No. 11, 2020, pp. 2435-2441. doi: 10.1049/iet-ipr.2019.0141

[5] K. Abdelouahab, M. Pelcat, J. Sérot, F. Berry, "Accelerating CNN Inference on FPGAs: A Survey", arXiv:1806.01683, 2018.

[6] K. Guo, S. Zeng, J. Yu, Y. Wang, H. Yang, "A Survey of FPGA-Based Neural Network Inference Accelerators", ACM Transactions on Reconfigurable Technology and Systems, Vol. 12, No. 1, 2019, p. 2, pp. 1-26. doi: 10.1145/3289185

[7] AMD, "Vitis High-Level Synthesis User Guide (UG1399, v2023.1)", Technical Report UG1399, 2023.

[8] R. Kastner, J. Matai, S. Neuendorffer, "Parallel Programming for FPGAs", arXiv:1805.03648, 2018.

[9] J. de Fine Licht, M. Besta, S. Meierhans, T. Hoefler, "Transformations of High-Level Synthesis Codes for High-Performance Computing", IEEE Transactions on Parallel and Distributed Systems, Vol. 32, No. 5, 2021, pp. 1014-1029. doi: 10.1109/TPDS.2020.3039409

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv:1704.04861, 2017.

[11] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21-26 July 2017, pp. 1251-1258. doi: 10.1109/CVPR.2017.195

[12] J. Hu, L. Shen, G. Sun, "Squeeze-and-Excitation Networks", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18-23 June 2018, pp. 7132-7141. doi: 10.1109/CVPR.2018.00745

[13] B. Rokh, A. Azarpeyvand, A. Khanteymoori, "A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification", ACM Transactions on Intelligent Systems and Technology, Vol. 14, No. 6, 2023, p. 97, pp. 1-50. doi: 10.1145/3623402

[14] J. Cheng, J. Wu, C. Leng, Y. Wang, Q. Hu, "Quantized CNN: A Unified Approach to Accelerate and

Compress Convolutional Networks", IEEE Transactions on Neural Networks and Learning Systems, Vol. 29, No. 10, 2018, pp. 4730-4743. doi: 10.1109/TNNLS.2017.2774288

[15] P. Kluska, M. Zięba, "Post-Training Quantization Methods for Deep Learning Models", Intelligent Information and Database Systems, Lecture Notes in Computer Science, Springer, Vol. 12033, 2020, pp. 467-479. doi: 10.1007/978-3-030-41964-6_40

[16] M. Sailesh, K. Selvakumar, N. Prasanth, "A Novel Framework for Deployment of CNN Models Using Post-Training Quantization on Microcontroller", Microprocessors and Microsystems, Vol. 94, 2022, p. 104634. doi: 10.1016/j.micpro.2022.104634

[17] B. Liu, D. Zou, L. Feng, S. Feng, P. Fu, J. Li, "An FPGA-Based CNN Accelerator Integrating Depthwise Separable Convolution", Electronics, Vol. 8, No. 3, 2019, p. 281. doi: 10.3390/electronics8030281

[18] L. Bai, Y. Zhao, X. Huang, "A CNN Accelerator on FPGA Using Depthwise Separable Convolution", IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 65, No. 10, 2018, pp. 1415-1419. doi: 10.1109/TCSII.2018.2865896

[19] Y. Ma, Y. Cao, S. Vrudhula, J.-S. Seo, "Performance Modeling for CNN Inference Accelerators on FPGA", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 39, No. 4, 2020, pp. 843-856. doi: 10.1109/TCAD.2019.2897634

[20] C. Huang, S. Ni, G. Chen, "A Layer-Based Structured Design of CNN on FPGA", Proceedings of the IEEE 12th International Conference on ASIC, Guiyang, China, 25-28 October 2017, pp. 1037-1040. doi: 10.1109/ASICON.2017.8252656

[21] L. Lu, Y. Liang, Q. Xiao, S. Yan, "Evaluating Fast Algorithms for Convolutional Neural Networks on FPGAs", Proceedings of the IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 30 April-2 May 2017, pp. 101-108. doi: 10.1109/FCCM.2017.64

[22] M. A. Arshed, S. Mumtaz, M. Ibrahim, C. Dewi, M. Tanveer, S. Ahmed, "Multiclass AI-Generated Deepfake Face Detection Using Patch-Wise Deep Learning Model", Computers, Vol. 13, No. 1, 2024, p. 31. doi: 10.3390/computers13010031

[23] S. Yang, P. Luo, C. C. Loy, X. Tang, "WIDER FACE: A Face Detection Benchmark", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27-30 June 2016, pp. 5525-5533. doi: 10.1109/CVPR.2016.596

[24] M. T. El-Melegy, H. A. M. Haridi, S. A. Ali, M. Abdelrahman, "Optimal Score Fusion via a Shallow Neural Network to Improve the Performance of Classical Open Source Face Detectors", Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Prague, Czech Republic, 25-27 February 2019, pp. 660-667. doi: 10.5220/0007691206600667

[25] T. Edirisooriya, E. Jayatunga, "Comparative Study of Face Detection Methods for Robust Face Recognition Systems", Proceedings of the 5th SLAAI International Conference on Artificial Intelligence, Colombo, Sri Lanka, 6-7 December 2021, pp. 1-6. doi: 10.1109/SLAAI-ICAI54477.2021.9664689

[26] J. Deng, J. Guo, E. Ververas, I. Kotsia, S. Zafeiriou, "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13-19 June 2020, pp. 5202-5211. doi: 10.1109/CVPR42600.2020.00525