Journal of Real-Time Image Processing (2024) 21:157
https://doi.org/10.1007/s11554-024-01538-1

RESEARCH

Energy efficiency assessment in advanced driver assistance systems with real-time image processing on custom Xilinx DPUs

Güner Tatar[1,2] · **Salih Bayar**[2]

Received: 24 May 2024 / Accepted: 11 August 2024
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2024

Abstract
The rapid advancement in embedded AI, driven by integrating deep neural networks (DNNs) into embedded systems for real-time image and video processing, has been notably pushed by AI-specific platforms like the AMD Xilinx Vitis AI on the MPSoC-FPGA platform. This platform utilizes a configurable Deep Processing Unit (DPU) for scalable resource utilization and operating frequencies. Our study employed a detailed methodology to assess the impact of various DPU configurations and frequencies on resource utilization and energy consumption. The findings reveal that increasing the DPU frequency enhances resource utilization efficiency and improves performance. Conversely, lower frequencies significantly reduce resource utilization, with only a borderline decrease in performance. These trade-offs are influenced not only by frequency but also by variations in DPU parameters. These findings are critical for developing energy-efficient AI-driven systems in Advanced Driver Assistance Systems (ADAS) based on real-time video processing. By leveraging the capabilities of Xilinx Vitis AI deployed on the Kria KV260 MPSoC platform, we explore the intricacies of optimizing energy efficiency through multi-task learning in real-time ADAS applications.

Keywords Deep learning processing unit · Multi-task learning networks · MPSoC-FPGA · Hardware accelerator · Vitis AI · Heterogeneous computing

1 Introduction

The interest in artificial intelligence (AI) has increased significantly due to the successes of Convolutional Neural Networks (CNNs) across various applications. These technologies are now commonly used in sectors such as healthcare [1, 2], finance [3], and consumer electronics [4]. AI's role in developing ADAS for autonomous vehicles stands out. This area includes real-time image and video processing, leading to the development of more sophisticated and in-depth NN models. These models are designed for essential computer vision and sensor fusion tasks, crucial for ADAS effectiveness.

The growing complexity and increasing parameter demands of these NN models require robust computational hardware. A combination of software parallelization libraries such as Pthreads and Message Passing Interface (MPI), alongside innovative hardware-software co-design strategies, is used to address these needs. These strategies employ powerful hardware like ASICs, MPSoCs with integrated FPGAs, and GPUs, significantly enhancing inference speed and flexibility compared to software-only solutions.

MPSoC-FPGAs and GPUs are favored for their excellent adaptability and parallel processing capabilities, vital for handling dynamic and high-speed image and video analysis tasks. Their benefits include performance and superior energy efficiency, making FPGAs a better choice than GPUs, which tend to consume more energy. The availability of programming tools like FINN [5], hls4ml [6], and Vitis AI [7] has simplified the use of high-level programming for FPGA-based AI applications, increasing the viability of

✉ Salih Bayar
salih.bayar@marmara.edu.tr

Güner Tatar
gtatar@fsm.edu.tr; gunertatar@marun.edu.tr

[1] Faculty of Engineering, Department - Electrical and Electronic Engineering, Fatih Sultan Mehmet Vakıf University, Sutluce Mah. karaağaç cad. no:12/A, 34445 Beyoğlu, İstanbul, Turkey

[2] Faculty of Engineering, Department of Electrical and Electronic Engineering, Marmara University, Recep Tayyip Erdoğan Külliyesi Aydınevler Mah., 34854 Maltepe, İstanbul, Turkey

Published online: 22 August 2024

MPSoC-FPGAs for comprehensive software and hardware development in real-time applications.

AMD Xilinx provides a range of reconfigurable DPU architectures, such as B512, B1152, and B4096, to meet different application needs. The choice of a DPU model significantly affects performance; for instance, the B4096 model is optimized for handling complex, real-time processing tasks effectively, while smaller DPUs might be adequate for less demanding tasks. Other essential performance factors include DPU frequency, BRAM, UltraRAM usage, and ALU parallelism. This study enhances the existing literature by providing a detailed analysis of performance, power, and energy consumption across various DPU configurations. Our findings offer valuable insights into improving the efficiency and effectiveness of real-time image and video processing applications in ADAS for autonomous vehicles.

2 Background

CNN integration within ADAS leverages its superior computer vision capabilities, which are essential for optimal functionality. Deep learning (DL) models like CNNs and DNNs excel in processing and analyzing image and video data from onboard sensors, inherently learning significant features and eliminating the need for manual feature engineering. These models recognize complex patterns, facilitate end-to-end learning, simplify system architecture, and enhance adaptability across various driving conditions. Our research focuses on developing DL-based ADAS algorithms, utilizing custom DPU overlays to exploit DL's transformative potential for advancing autonomous driving technologies [8].

This objective requires software optimizations on hardware accelerator platforms such as FPGAs, GPUs, or ASICs, known for their efficiency and parallel processing capabilities. Effective hardware-software co-design within MPSoC architectures that integrate ARM and FPGA components is crucial. This approach allows the DPU to manage computationally intensive tasks while the ARM processor handles less demanding tasks, optimizing resource utilization and performance for ADAS applications. Recent scholarly research underscores a growing preference for application-specific platforms such as FPGAs and ASICs [9].

Our study selected the Kria KV260 MPSoC-FPGA for its suitability for autonomous driving, particularly ADAS, due to its computational capability, energy efficiency, and adaptability. The reconfigurable nature of the Kria KV260 facilitated detailed frequency and power analyses across seven different hardware setups, helping us identify the most suitable configuration for ADAS. The board's high-speed interfaces and specialized hardware accelerators meet our system requirements effectively. Additionally, the flexible development environment of the Kria KV260 allows for customizable hardware setups, including DPUs and software configurations. We implemented a multi-task learning ADAS algorithm across seven hardware architectures, demonstrating the board's capability and flexibility in managing complex ADAS tasks.

3 Related work

The existing literature extensively discusses DL solutions that prioritize power consumption and efficiency. Focusing on research that utilizes similar hardware and applies to comparable use cases is crucial. Our research aims to enhance autonomous vehicle technology by optimizing energy consumption and efficiency by exploring various DPU architectures. Consequently, our literature review encompasses studies and developments within this domain.

In [10], the researchers compared the Learned Approximate Matrix Profile (LAMP) algorithm on the Xilinx Ultra96-V2 and the Raspberry Pi 3 platforms. Using DPU architectures of various dimensions, they demonstrated that their design outperforms the Raspberry Pi 3 in performance efficiency and power consumption metrics. Goel et al. [11] introduced EnergyNN, a framework designed to forecast the energy consumption performance of models executed on DPUs. Their methodology incorporates a nonlinear regression model with an error rate of 9.9% across 16 renowned CNN models. However, their framework is limited to specific models running on the DPU without considering variations across different DPU architectures. This limitation suggests an area for further research into adapting EnergyNN or developing new methodologies to account for the diverse characteristics and capabilities of various DPUs.

A noteworthy investigation by [12] examines the consequences of reducing the voltage supply to FPGAs to decrease the power consumption attributed to both the FPGA and the embedded DPU. The research reveals that a reduction in voltage leads to a substantial decrease in power consumption, resulting in an enhancement of the efficiency metric GOPs/W by approximately 3x. However, a critical observation from this study is the negative effect of voltage reduction on computational accuracy. We may suggest changing the DPU size and reducing the frequency to mitigate this effect.

Vandendriessche et al. [13] conducted a significant study exploring the relationship between power consumption, operating frequencies, and DPU dimensions. Their research focused on the effect of frequency scaling on the Xilinx DPU integrated within a ZCU104 platform. Findings indicate that smaller DPU configurations, which utilize fewer resources yet operate at higher frequencies, can perform inference tasks at speeds equivalent to or surpassing those of larger

configurations that consume more resources at lower frequencies. Additionally, larger DPU configurations exhibit reduced energy consumption when operating at elevated frequencies. Acknowledging that DPUs are influenced by many parameters, including ALU parallelization, DSP usage, and the selection of activation functions, is crucial. Thus, while this study provides valuable insights into frequency analysis, it represents a partial view of the complex parameter space affecting DPU performance and efficiency.

Employing DPUs with strategic efficacy, Li et al. [14] introduced an innovative architecture termed Receptive Field Attention YOLO (RFA-YOLO), which incorporates a feature fusion neck network alongside an attention mechanism. Their methodology exhibits commendable performance, rivaling that of GPUs even when operating at reduced frequencies, and achieves a significant reduction in power consumption, estimated at approximately 89.72%. Despite acknowledging the dependency of power consumption on operational frequency, their publication omits specific frequency data. Furthermore, their experimental framework utilizes the standard overlay provided by the Xilinx Vitis-AI platform, suggesting a reliance on pre-established computational environments without modifications to the underlying hardware configuration. Zhao et al. [15] developed an FPGA-compatible floating-point data format designed to match the storage efficiency of the int8 format without sacrificing inference accuracy or necessitating fine-tuning. Their findings demonstrate that this format maintains the same level of accuracy as local floating-point representations across multiple neural networks while sustaining robust computational performance. Specifically, their methodology performs at a DPU frequency of 250 MHz when benchmarked against similar research. However, their publication needs a comparative analysis of enhanced models and provides more information regarding the specifications of current DPU technologies. In their study, Jiang et al. [16] utilized a custom accelerator to enhance the inference speed of a modified VGG16 model. They applied quantization to the model using CHaiDNN [17], an AI accelerator that uses Xilinx's High-Level Synthesis (HLS) technology. By implementing various techniques and modifying the accelerator's frequency, they could attain power savings of up to 40%.

This study breaks new ground by comprehensively analyzing diverse DPU sizes and parameters, including ALU parallelization, RAM utilization, Separable Depth-Wise Convolution (SDWC), channel augmentation, and activation function selection. This research is significant as it is the first to explore the intersection of high-performance and low-power consumption considerations while accommodating the various parameters outlined, particularly in advancing ADAS technology. Additionally, our research emphasizes multi-task learning, which AMD Xilinx provides [18], integrating crucial computer vision tasks such as multi-object detection and segmentation, which are essential for ADAS. This study also offers a comparative analysis with other leading-edge research, providing a broader context for evaluating the advancements presented.

4 Contributions

Our study substantially contributes to understanding DPU architectures in ADAS for multi-task learning. We first evaluate the frequency performance of four DPU architectures (B1152, B2304, B3136, and B4096), each operating at different clock speeds (200MHz, 275MHz, and 300MHz). This analysis is critical to comprehending how various DPUs perform under different conditions within ADAS applications.

Moreover, we delve into the energy efficiency of these DPUs in battery-powered ADAS devices, exploring their power consumption at different frequencies.

Our findings have significant practical implications, providing critical insights into the cost-effectiveness of each DPU. This aids professionals in choosing the most suitable architecture for their needs, thereby optimizing the energy consumption of their ADAS devices. Furthermore, we carefully examine the resource demands and computational limits of these DPUs, which are pivotal for deploying efficient multi-learning models in devices with constrained resources. This comprehensive assessment significantly advances our knowledge of energy-efficient DPU architectures and their crucial role in embedded systems, where resource optimization is paramount.

4.1 Paper organization

This paper is organized as follows: Sect. 5 describes our experimental setup, hardware and software components, and details on the data sets and training methods. Section 6 analyzes the experimental results, comparing the DPUs' performances with detailed graphical and numerical data on resource and power consumption. Finally, Sect. 7 discusses the implications of our findings and outlines future research directions.

5 Experimental setup

We enhance power efficiency and inference performance by examining various DPU sizes and architectures within our multi-task learning model. Our system was implemented on the Kria KV260 Vision development board based on the MPSoC-FPGA architecture. The hardware design, called "overlay," was realized using the Vivado 2022.1 integrated development environment. We employed the DPUCZDX8G (DPU) v4.0 architecture for this hardware design. To

reconfigure the hardware design on the development board, we created the SD card image using Petalinux 2022.1.

Performance evaluations were conducted with the highest precision and focused on a multi-task learning model that included segmentation and object detection. All models were precisely tested under similar conditions, with power consumption and inference performance being the crucial criteria. Our experimental setup, carefully designed, included a Logitech c920 camera, the Kria KV260 development board, and an HD monitor connected via HDMI. The setup is depicted in Fig. 1. The Kria KV260 is linked to the HD monitor through an HDMI cable. The development board and the PC share a network connection via a USB cable and Ethernet. This configuration allows the video feed captured on the PC to be transferred to the development board via the camera and then processed and outputted, as illustrated in Fig. 2. Detailed procedures for processing the video image are provided in Sect. 5.5.

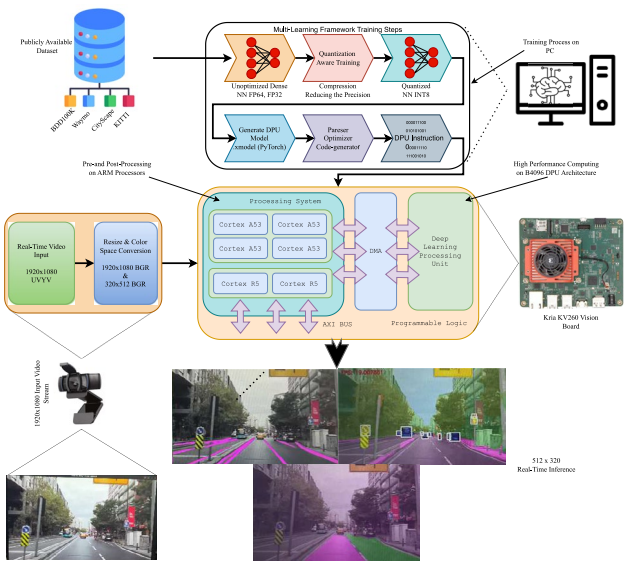


Fig. 2 Detailed block diagram of the multi-learning model used for the study

5.1 Hardware design

Our hardware setup integrates components `zynq_ultra ps_e`, `dpu_trd`, and auxiliary `IPs` in the Vivado 2022.1 environment, as shown in Fig. 3. We designed unique configurations for each case in Table 1, creating specific SD card images. The designs, particularly for compact DPU models like B1152, focus on achieving high performance at lower frequencies, requiring precise parameterization due to limited resources. Our design uses the Vitis-AI compiler for efficient memory instruction processing and leverages on-chip memory to minimize external memory dependency, increasing data reuse and accelerating performance. We

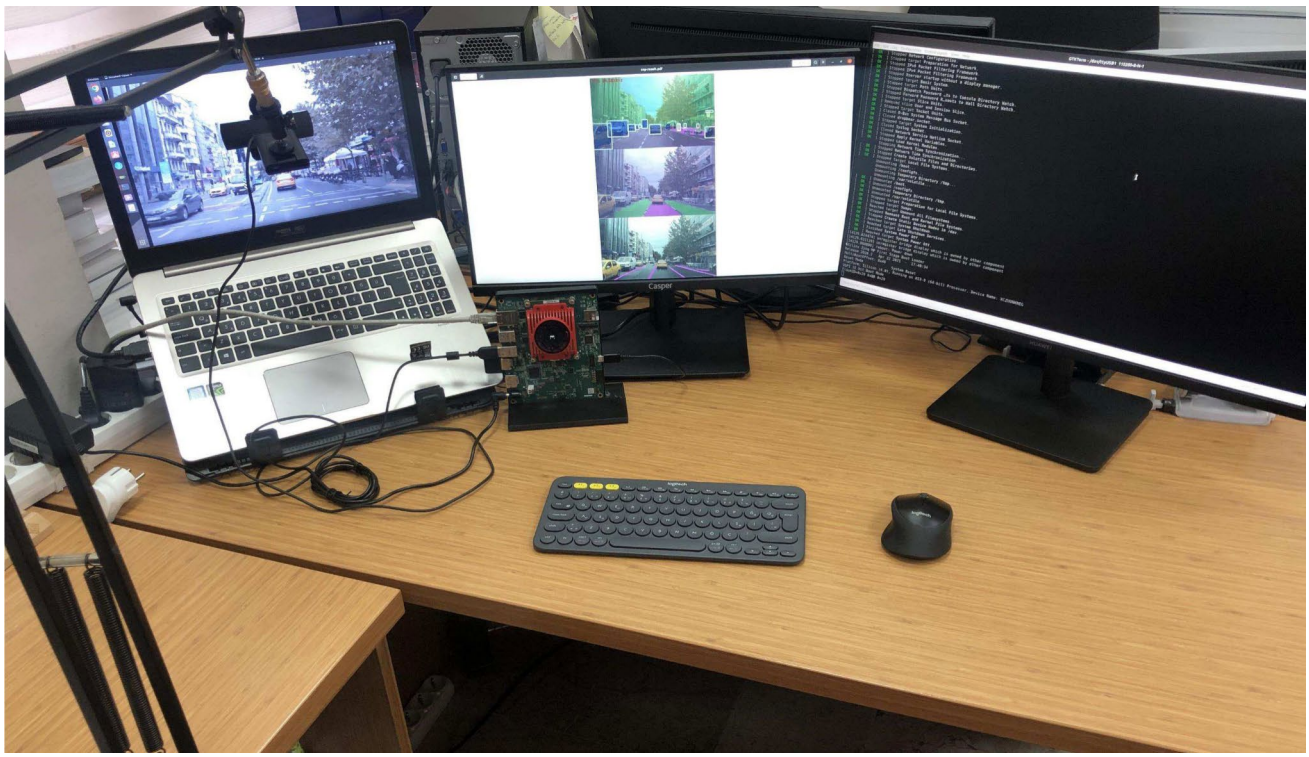


Fig. 1 Experimental setup to obtain the results of the proposed study

Table 1 DPU configuration cases in the proposed study

DPU configuration	Configuration 1	Configuration 2	Configuration 3	Configuration 4	Configuration 5	Configuration 6	Configuration 7
DPU architecture	B1152		B4096		B3136		B2304
DPU_Clk MHz	275	300	275	300	200	300	300
Channel augment	Enabled	Disabled	Enabled				Disabled
RAM usage	High	Low	Low	High			
Ultra-RAM per DPU	32	0	64		48		
DSP48 usage	High	Low	High				
Depth-wise convolution	Enable						
Average pooling	Enable						
DSP48 Max cascade length	2	4	4				
Convolution ReLU type	ReLU + ReLU6		ReLU + ReLU6	ReLU + LeakyReLU + ReLU6	ReLU + ReLU6	ReLU + LeakyReLU + ReLU6	ReLU + ReLU6

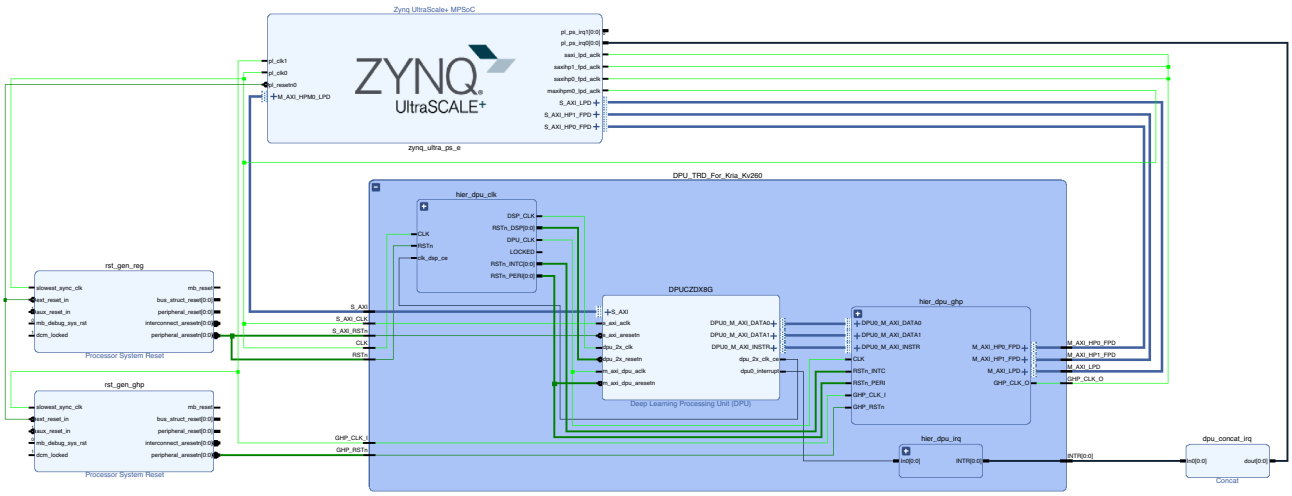


Fig. 3 Custom overlay design in the proposed study

optimized DPU architectures for different levels of parallelism, impacting resource utilization and performance. We prioritize compact, efficient designs suitable for resource-constrained devices.

Additionally, we optimized the system using a Double Data Rate (DDR) configuration and carefully selected DSP cascade lengths to balance resource consumption and timing. Our most significant DPU setup uses extensive DSP resources and employs URAM for better support in larger architectures. We examined seven DPU configurations, optimizing BRAM and URAM use to enhance performance, as detailed in Table 1, adjusting memory units and URAM blocks for effective resource allocation.

We analyze the adaptability and performance metrics of the Kria KV260 platform across various configurations. This analysis encompasses an assessment of resource utilization, including LUTs, LUTRAM, FFs, BRAM, URAM, DSP, and BUFG, alongside the specifications of the DPU, such as its frequency and architecture. The configurations reveal a strategic allocation of resources, with utilization rates varying to demonstrate the platform's capacity for customization to specific computational tasks. Notably, DPU frequencies are set at 200 MHz, 275 MHz, and 300 MHz for different configurations, and DPU architectures vary among B1152, B2304, B3136, and B4096, indicating a tailored approach to achieving a balance between computational efficiency and power consumption. This configuration variability highlights the Kria KV260's versatility, enabling it to address various applications, from high-throughput processing to energy-efficient deployments. The analysis portrays the platform as

a highly adaptable solution, capable of being optimized to fulfill diverse application requirements, thereby proving its suitability for various computational tasks across multiple domains.

5.2 Data sets

We trained our model using four different data sets: BDD100K [19], Waymo [20], KITTI [21], and CityScapes [22]. Initially, BDD100K and Waymo were employed for diverse driving scenes, while KITTI was utilized for object detection across four classes. CityScapes was used for 16 classes for semantic segmentation and three different classes for drivable area segmentation. Table 2 details the data set distribution for training and inference.

To improve accuracy and efficiency, we consolidated data sets into categories: CityScapes and BDD100K for segmentation, BDD100K, and Waymo for object detection. This strategic integration and accurate data partitioning for training, testing, and validation yielded promising results. We evaluated the performance of our model using mean Average Precision (mAP) (1) for object detection and mean Intersection over Union (mIoU) (2) for segmentation:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{1}$$

Here, AP_k is the AP of class k, n is the number of classes,

$$mIoU = \frac{\left(\frac{1}{n_{cl}}\right)\sum_i n_{ii}}{\left(t_i + \sum_j n_{ji} - n_{ii}\right)} \tag{2}$$

In these evaluations, n_{cl} represents the number of classes, t_i denotes the total number of pixels in class i, n_{ii} indicates true positives, and n_{ji} accounts for false negatives.

5.3 Training procedure

Developing a multi-task model for ADAS required a careful selection of loss functions and a high-performance GPU to handle large data sets. We split the model into heads, training them independently to reduce computational load. The backbone head was initialized using the ImageNet data set. The training involved object detection, segmentation, and backbone encoder heads. Initially, the training focused on setting up image weights until the loss function converged. We utilized Quantization-Aware Training (QAT) to optimize weight and activation functions. Pre-processing included resizing input images for multi-object detection. After pre-training, the model underwent fine-tuning with labeled data, updating shared backbone and task-specific heads parameters.

Various hyperparameters were tested to optimize the model, with performance monitored using a validation set. For object detection, *focal_loss2d* and soft L_1 loss were used, addressing class imbalance and outliers, respectively. Stochastic Gradient Descent (SGD) refined the model with a learning rate of 1×10^{-6} and momentum of 0.9. A batch size of 32 and binary cross-entropy loss *BCELoss* was used in the SSD [23]. The segmentation head used *LovaszSoftmaxLoss*, optimized with SGD and a learning rate of 1×10^{-3}, and a batch size of 16. The model incorporated seven custom DPU overlay designs, each tailored for different tasks to maximize efficiency and performance. Equations (3), (4), and (5) detail the mathematical specifications of SGD and *LovaszSoftmaxLoss*.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{3}$$

$$\nabla = \frac{\partial}{\partial x}\hat{i} + \frac{\partial}{\partial y}\hat{j} + \frac{\partial}{\partial z}\hat{k} \tag{4}$$

Here, ∇ is the gradient operator, η is the learning rate, $x^{(i)}$ is the training sample, and $y^{(i)}$ denotes the label.

$$\text{loss}(f) = \frac{1}{|C|}\sum_{c\in C}\overline{\Delta_{J_c}}(m(c)) \tag{5}$$

Where Δ_{J_c} is Jaccard loss extension, $m(c)$ is the vector of errors and C represents class.

5.4 Quantization aware training (QAT)

QAT is an essential technique for optimizing DL models for deployment in resource-constrained environments, such as embedded systems or mobile devices. We implement QAT to explicitly account for quantization effects during the training phase, enabling the model to accommodate the inherent precision loss and thereby preserving high accuracy levels when the model operates in a quantized state.

Table 2 Use of data sets

Data sets	Task	Train	Validation	Test
BDD100K + Waymo	Multi-object det	228,081	50,077	50,000
BDD100K + CityScape	Segmentation	72,975	10,500	21,525
BDDD100K	Drivable area seg	70,000	10,000	20,000
BDDD100K	Lane seg	70,000	10,000	20,000

As shown in Fig. 4, quantization reduces the precision of the numerical values within a model by converting floating-point numbers to a lower-bit representation, such as fixed-point numbers or integers. This reduction in precision facilitates more efficient computation and diminishes memory requirements. However, it may lead to accuracy degradation if not managed correctly. During QAT, we simulate quantization within the training process by incorporating it into both the forward and backward propagation stages. This integration helps the network to adapt to and mitigate the errors introduced by quantization.

Multi-task learning leverages a single model architecture to perform several tasks simultaneously, optimizing parameter usage across tasks compared to training separate models for each task. We followed the steps listed below to apply QAT in a multi-task learning framework and improve the accuracy of the inference.

1. We define a quantization plan, deciding which layers to quantize and the quantization type (e.g., during training). Parameters like scale and zero-point for each layer are optimized during the training.
2. We modify the training algorithm to incorporate quantization operations, applying quantization to weights and activations during the forward pass and dequantizing during the backward pass for gradient computation.
3. We performed post-training fine-tuning without quantization to recover any potential loss in inference accuracy.

The primary operations in QAT are quantization and de-quantization, described by the following equations:

- Quantization

$$Q(x) = \text{round}\left(\frac{x}{\text{scale}}\right) + \text{zero_point} \tag{6}$$

- De-quantization

$$D(q) = (q - \text{zero_point}) \times \text{scale} \tag{7}$$

Here, x represents the real-valued input, while q denotes the quantized value. Parameters scale and zero_point are learned during training and adjust the quantization process to fit the model's distribution of values.

We employ QAT within our model, applying a nuanced approach to manage each task's unique sensitivity to quantization errors. This process may necessitate task-specific quantization adjustments or varied quantization strategies across different network segments, ensuring that the model remains robust against potential degradation in performance due to quantization.

Following the quantization phase, we progress to the model's compilation, utilizing the `vai_c_xir` command provided by the Vitis AI Compiler. We compile model with the specific parameters defined in the architecture configuration file (`arch.json`) that targets the desired DPU architecture. This configuration enables the compiler to apply targeted optimizations that enhance inference speed and operational efficiency on the chosen hardware platform. The specific command executed during this phase is `vai_c_xir -xmodel input_model.xmodel -arch /path/to/arch.json -output_dir./compiled_model`.

Here, `-xmodel` defines the path of the input model, `-arch` identifies the DPU configuration, and `-output_dir` sets the directory for storing the compiled model. The compilation process deploys the optimized model onto the DPU, ensuring it is well-prepared for efficiently executing real-time data processing tasks within various applications. This step marks the transition from model preparation to active deployment, highlighting our comprehensive approach to developing and deploying high-performance AI solutions in our edge device.

5.5 Multi-task learning network

The efficacy of the multi-task learning model in practical implementations is evident from its significant contributions to resource optimization and performance accuracy. Recent literature highlights a burgeoning interest in multi-task learning, where networks simultaneously tackle multiple tasks, thus enhancing learning efficiency and boosting

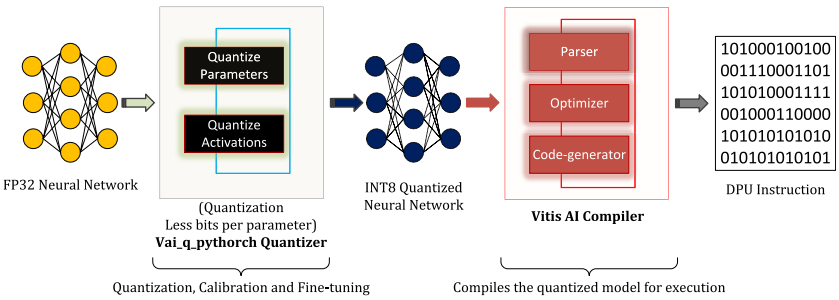


Fig. 4 Quantization and compilation of FP32 to INT-8 bit-width

prediction accuracy through shared features [19, 24, 25]. In line with these advancements, we have devised a novel approach tailored for MPSoC FPGA architecture, ideally suited for ADAS projects that involve semantic segmentation and object detection. This strategy incorporates a CNN-based shared backbone encoder that adeptly extracts vital features from input videos, significantly advancing the multi-task learning paradigm.

Our model surpasses mere theoretical development; it is an actionable, applied solution. Figure 2 demonstrates how synthesizing insights from various tasks improves overall task performance. After extensive training on public data sets, we deploy this model on seven custom-configured DPUs, each rigorously identified and compiled according to the designated DPU-TRD on the Kria KV260 platform. This arrangement is critical, not just a technical detail, as it is crucial for effectively processing and extracting real-time camera data, thereby showcasing our research's practical relevance and application. The architecture of our system is characterized by efficiency and ingenuity. We have opted for a shared backbone encoder based on the ResNet architecture, emphasizing our commitment to compatibility with resource-constrained embedded devices. Integrating a pre-trained ResNet model boosts our model's learning efficiency. It lowers computational demands, positioning our approach as a dependable and effective solution that is ideally suited to the limitations of embedded devices.

Our multi-task learning framework is designed for quick inference, which is essential for real-time object detection. By employing the SSD [23] model, we successfully reduce feature map dimensions to more precisely locate larger objects through a 3×3 feature map extractor kernel. Utilizing the VGG16 architecture, we add convolutional feature layers that facilitate the model to perform approximately 8732 predictive detections at various scales. The system also features a multi-layer sub-network for segmentation tasks such as drivable area and line detection, integrating learnable upsampling, 3×3 convolutional layers, batch normalization, and activation functions.

To enhance detail retention beyond conventional pooling methods, we employ dilated convolution within the convolution block for tasks associated with drivable area segmentation. An upsampling step precisely restores spatial dimensions, and the integration of ResNet into the decoder refines semantic data boundaries, improving the precision in defining object boundaries. By prioritizing feature integration over simple layer stacking, our approach focuses on inference accuracy and efficiently uses our custom DPU overlays within this robust multi-task learning framework. This methodological focus responds to the reviewer's inquiry about the practical benefits of the "multi-learning model" and underscores its ability to enhance performance while managing resource use effectively.

6 Experimental results

To assess the outcomes of the performance analysis within our research, we concentrated on three primary metrics: power consumption, resource utilization, and inference efficiency. We conducted multiple analyses of this scenario, encompassing the correlation between power consumption and thread count, the relationship between instantaneous power consumption and thread count, the connection between inference performance and thread count, and the total resource utilization for each configuration examined. As expected, the observed findings indicated that embracing a larger-size DPU architecture increases resource utilization. Additionally, our results clarify that the frequency is directly associated with the inference rate. The research showed that increased frequency concomitantly enhances the inference under comparable conditions. The analytical discussion of our experimental endeavors is systematically organized under respective subheadings.

Tables 3 and 4 show the percentages of total resource consumption for seven different configurations, offering a clear view of efficiency and resource demand. Configurations 1 and 2 show very similar levels of resource consumption at 26.99% and 27.07%, respectively. This indicates they are both optimized for resource efficiency or tasks requiring lower computational loads. The proximal values suggest similarities in architectural design or intended operational functionality. Configuration 3 marks a significant increment in resource utilization to 44.80%, reflecting its orientation towards more resource-intensive tasks or a design that inherently demands more excellent computational resources. This leap intimates a tailored design for applications necessitating enhanced computational power or memory allocation. Configuration 4, with the highest resource consumption at 52.20%, underscores its allocation towards the most demanding computational tasks, possibly leveraging advanced processing capabilities or heightened parallelism. Configuration 5, positioned at 37.94%, represents an intermediate stance in resource utilization, suggesting its design compromises computational performance and resource efficiency, making it suitable for a broad spectrum of tasks with moderate resource demands. Configuration 6, with a total resource consumption of 38.46%, aligns closely with Configuration 5, indicating its suitability for moderately demanding applications while maintaining a balance between performance and resource efficiency. Lastly, Configuration 7, consuming 29.16%, appears to be optimized for tasks that require slightly more resources than Configurations 1 and 2 but remain within a relatively efficient range. The resource consumption profiles collectively underscore the

Table 3 Comparison of resource utilization of the proposed study in the cases

Kria KV260		Conf. 1		Conf. 2		Conf. 3		Conf. 4		Conf. 5		Conf. 6		Conf. 7	
Resource	Available	Utilization	Utilization (%)	Utilization	Utilization (%)	Utilization	Utilization (%)	Utilization	Utilization (%)	Utilization	Utilization (%)	Utilization	Utilization (%)	Utilization	Utilization (%)
LUTs	117,120	35,955	30.7	35,489	30.3	51,300	43.8	65,329	55.78	46,233	39.47	57,607	49.19	43,607	37.23
LUTRAM	57,600	4068	7.06	3983	6.91	7264	12.61	8041	13.96	5374	9.33	6127	10.64	4308	7.48
FFs	234,240	56,595	24.16	57,360	24.49	101,415	42.3	113,127	48.3	83,860	35.80	93,509	39.92	71,333	30.45
BRAM	144	0	0	121	84.03	67.5	46.88	89.13	79	104	72.22	108	75	77	53.47
URAM	64	32	50	0	0	64	100	64	100	48	75	48	75	48	75
DSP	1248	246	19.71	198	15.87	710	56.89	724	58.01	566	45.35	580	46.47	438	35.1
BUFG	352	4	1.14	4	1.14	4	1.14	4	1.14	4	1.14	4	1.14	3	0.85
DPU frequency		275 MHz		300 MHz		275 MHz		300 MHz		200 MHz		300 MHz		300 MHz	
DPU architecture		B1152				B4096				B3136				B2304	

Table 4 Total resource consumption for each configuration

Configurations	Total resource consumption (%)
Config. 1	26.99
Config. 2	27.07
Config. 3	44.80
Config. 4	52.20
Config. 5	37.94
Config. 6	38.46
Config. 7	29.16

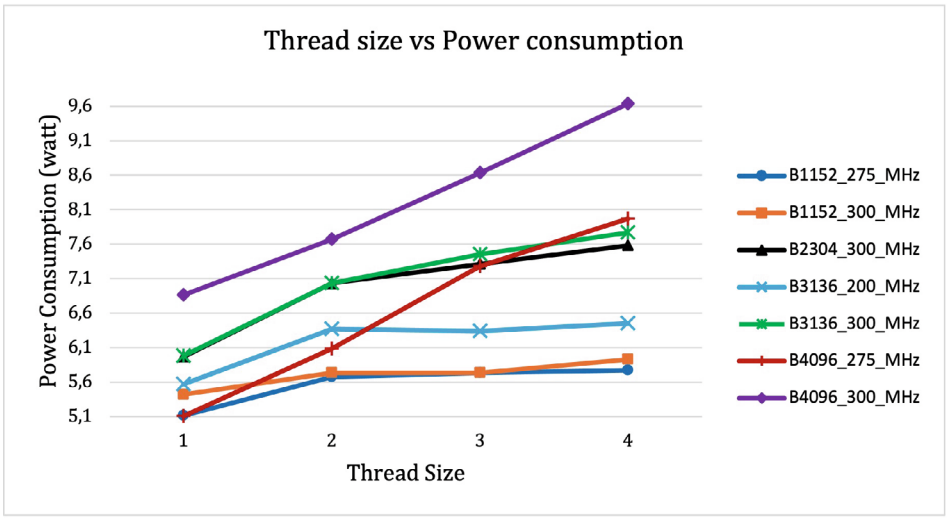
configurations' diverse architectural strategies and application targets, ranging from efficiency-oriented designs to those catering to high-performance requirements.

6.1 Power consumption and inference efficiency

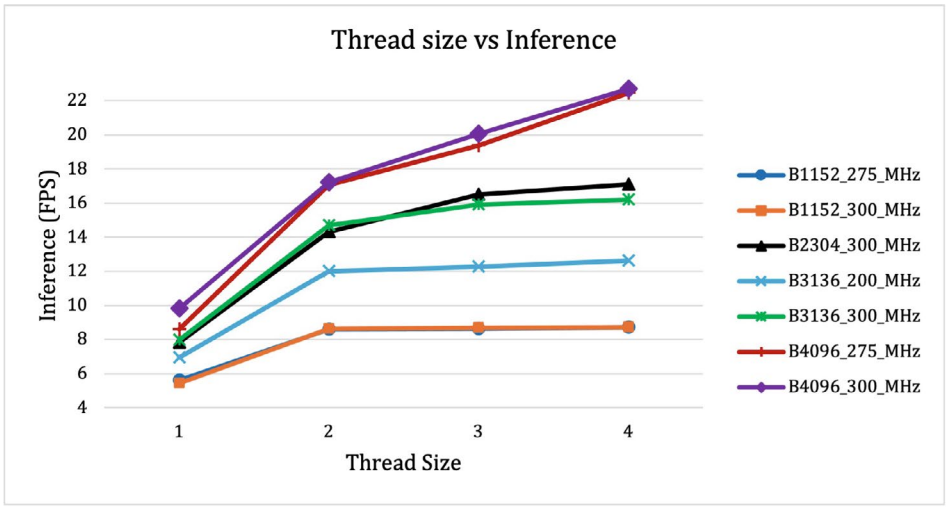
In our research, which focuses on integrating hardware and software design, the choice of DPU configurations is crucial, as it significantly affects resource utilization, inference performance, and, consequently, power consumption. This analysis focuses on variations across both the PS and PL configurations. Specifically, the hardware platform utilized, the Kria KV260, supports DPU on the PL side, complemented by a quad-core ARM Cortex A-53 processor on the PS side. This setup allows for managing FPGA parallelism via threading. Our study explores power consumption using threading for task allocation and implementing a pipeline architecture on the PL side.

A comparative assessment of various DPU configurations highlights the trade-offs between energy efficiency and computational performance. Figure 5a illustrates that the *B4096_300_MHz* configuration uses more power at all thread sizes, making it the best option for exceptional computational performance, though this comes with increased energy costs. Conversely, Fig. 5b demonstrates how all configurations enhance performance with increased thread sizes. Nonetheless, the power efficiency of the *B4096_300_MHz* diminishes with larger thread sizes, pointing to inefficient energy usage for marginal performance gains.

This comprehensive analysis underlines the necessity of selecting DPU configurations that align precisely with specific application needs, ensuring a balance between computational speed and energy consumption. It reveals that while configurations such as *B4096_300_MHz* may provide high computational throughput, alternatives like *B2304_300_MHz* offer a more energy-efficient solution without significantly sacrificing performance, especially at larger thread sizes. The findings are pivotal for optimizing DPU selection to achieve the best balance between power



(a) Threads and power consumption relation



(b) Threads and FPS relation

Fig. 5 Performance comparison of proposed study based on frequency and DPU architecture

consumption and computational efficiency in scenarios where energy sensitivity is paramount.

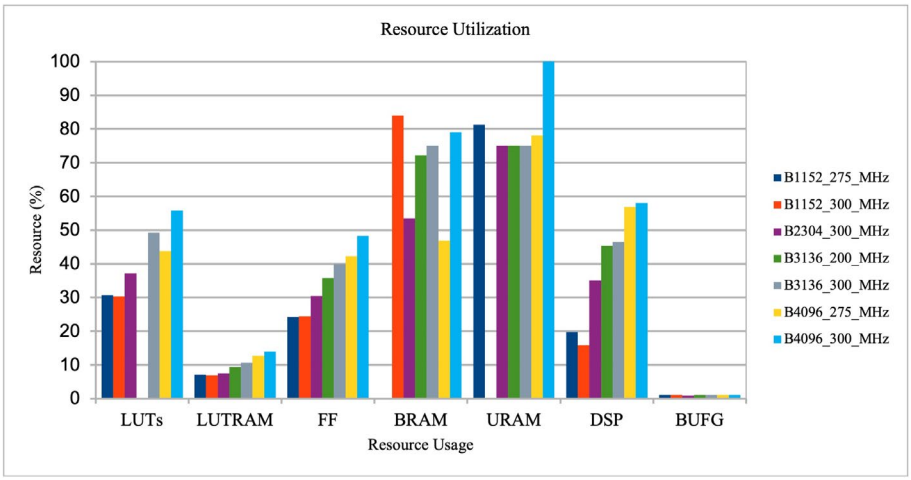
Further investigation into configurations *B1152_275_MHz*, *B3136_200_MHz*, and *B3136_300_MHz* consistently emphasizes the relationship between computational throughput and energy efficiency. Trends indicate that higher power consumption accompanies increased thread sizes, especially for configurations like *B4096_275_MHz* and *B4096_300_MHz*, which require more power for potentially higher speeds. The *B3136_200_MHz* presents a more power-efficient option with a modest drop in performance, suitable for energy-constrained applications. Meanwhile, *B1152_275_MHz* emerges as the most energy-efficient, albeit with lower computational output, ideal for scenarios where moderate inference speeds suffice and energy conservation is critical.

Choosing among these configurations necessitates carefully considering the application's performance requirements versus energy constraints. Configurations like *B4096_300_MHz* are preferable for maximum performance with less concern for energy consumption, whereas *B3136_200_MHz* offers a balanced option for those seeking a compromise between efficiency and speed. *B1152_275_MHz* is designed for applications prioritizing energy efficiency over high computational speed. This emphasizes the importance of aligning DPU selection with an application's computational demands and energy limitations to achieve an optimal balance of efficiency and performance.

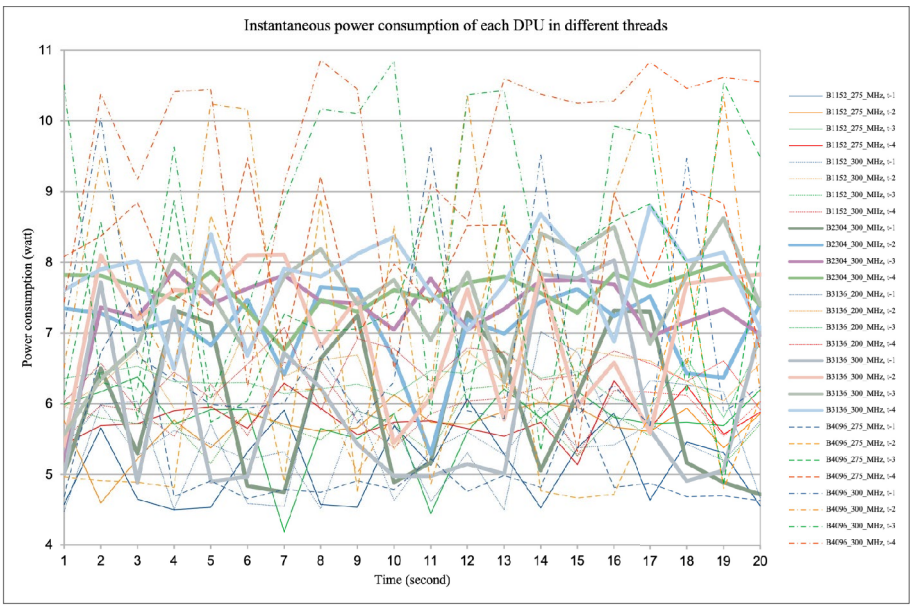
6.2 Resource utilization

In examining the resource utilization and instantaneous power consumption characteristics of various DPU architectures, our analysis reveals distinct operational profiles with significant implications for their application in computational tasks. Figure 6a delineates the utilization percentages of essential hardware resources. Notably, the *B4096_300_MHz* architecture exhibits pronounced utilization of DSP resources, indicative of its optimization for high-speed arithmetic operations key in intensive computational tasks. Conversely, the *B3136_300_MHz* configuration displays a more balanced resource allocation, indicating a design oriented towards versatility or generalized computing. The utilization percentages across all configurations reveal notable trends like;

- **LUT usage**: *B4096_300_MHz* shows the highest LUT utilization, aligning with its design for high computational throughput.
- **LUTRAM usage**: *B3136_300_MHz* demonstrates balanced usage, suggesting versatility.
- **FF usage**: *B2304_300_MHz* maintains moderate utilization, suitable for general tasks.
- **BRAM usage**: *B1152_300_MHz* shows lower usage, indicating efficient memory use.
- **URAM usage**: *B4096_300_MHz* again shows higher usage, aligning with its high-performance focus.



(a) Resource usage for each DPU architectures



(b) Instantaneous power consumption for 20 sec

Fig. 6 Resource utilization and instantaneous power consumption for each DPU architecture



(a) summer night: object and line detection

(b) summer night: segmentation

(c) summer night: drivable area segmentation

(d) winter night: object and line detection

(e) winter night: segmentation

(f) winter night: drivable area segmentation

(g) summer time: object and line detection

(h) summer time: segmentation

(i) summer time: drivable area segmentation

Fig. 7 Application results of the multi-learning network under different weather conditions using *Configuration3*

- **DSP usage**: High usage in Configuration *B4096_300_MHz* reflects its suitability for arithmetic-heavy tasks.

Figure 6b further elaborates on the dynamic power consumption patterns of these DPU architectures over 20 s across varying thread sizes. The figure displays a noticeable variability in power consumption, with distinct peaks and troughs that reflect the fluctuating workload demands. Architectures *B1152_275_MHz* and *B1152_300_MHz* are characterized by a closer clustering of power usage across all thread sizes, suggesting a commendable level of power management efficiency amidst computational demand oscillations. In contrast, the *B4096_275_MHz* and *B4096_300_MHz* configurations reveal broader variances in power consumption, especially at higher thread sizes, underscoring

Table 5 Evaluation result for *Configuration 3* (B4096_275_MHz) of the study on INT8 and FP32 model types

Multi-task learning model	Floating point (FP32)	Quantization aware training (INT8)
Segmentation mIoU (%)	59.95	57.76
Multiple object detection mAP (%)	51.2	50.1
Drivable area detection mIoU (%)	83.78	82.65
Line detection IoU (%)	44.78	45.86
Memory footprint (MB)	69.76	63.3
Performance	25 G FLOPs	708.859 GOPs

their deployment in tasks of substantial computational intensity that intermittently necessitate increased power.

The synthesized insights from these observations underscore the critical role of architectural choices in DPU design, particularly emphasizing the balance between resource utilization efficiency and power consumption dynamics. The high resource and power utilization patterns of the *B4096_300_MHz* configuration align it with high-performance applications. In contrast, the *B3136_300_MHz*'s moderate and balanced profile denotes suitability for general-purpose computing scenarios. This analysis highlights the criticality of aligning DPU architectural selection with the specific computational demands and energy efficiency requirements of the intended application, optimizing performance output and operational sustainability. Figure 7 presents the multi-learning network's application results under various weather conditions using *Configuration 3*. Although any DPU architecture could be employed for these results, *Configuration 3* was selected for demonstration purposes.

Table 5 compares two computational models-Floating Point (FP32) and Quantization Aware Training (INT8)-designed for multi-task learning applications. In evaluating performance metrics such as mIoU for segmentation and drivable area detection, mAP for multiple object detection, and IoU for line detection, the FP32 model generally exhibits slightly higher accuracy, with a segmentation mIoU of 57.95% compared to 56.62% for the INT8 model, and a drivable area detection mIoU of 82.63% versus 81.56%. However, the INT8 model performs nearly equivalently in multiple object detection (51% mAP compared to 51.29% for FP32) and even slightly outperforms in line detection (43.86% IoU against 43.65%). Additionally, the INT8 model is more memory efficient, with a footprint of 63.3 MB compared to 69.76 MB for the FP32 model, reflecting the quantization benefits in reducing resource requirements-crucial for deployment in environments with limited computational resources. Most notably, the INT8 model's computational performance is significantly higher at 708.859 GOPs compared to 25 G FLOPs for the FP32 model, underscoring the efficiency gains from quantization that favor scenarios demanding real-time processing and low-latency responses. These results indicate that while there is a marginal reduction in accuracy in some areas when using the INT8 model, the decrease is minimal relative to the substantial improvements in throughput and memory efficiency, making the INT8 model a compelling choice for applications where performance and efficiency are prioritized over small increases in accuracy. This balance is especially vital in real-time systems such as ADAS, where operational efficiency is paramount.

Table 6 offers a nuanced comparison of various state-of-the-art studies, each designed to meet specific operational demands ranging from defect detection to multi-object tracking. This comparative framework highlights the distinct approaches utilized across different studies to optimize performance, precision, and resource utilization within their respective platforms. While some studies demonstrate efficiency and robust performance on platforms like XCZU7EV and ZCU102, the reported data for critical metrics such as precision and power consumption are lacking. These omissions can complicate the assessment of their practical viability and real-world applicability. Moreover, some configurations, although powerful, entail significant resource consumption and potentially higher costs due to premium platforms.

In contrast, *Configuration 3*, utilizing the Kria KV260 platform, distinguishes itself by its cost-effective approach. With a precision of INT8 and a power usage of only 7.19 watts, it achieves a commendable balance between computational efficiency and power consumption, making it highly suitable for multi-task learning environments where operational cost and energy efficiency are pivotal. Despite its sophisticated capabilities, *Configuration 3* is deployed on one of the cheaper boards available, underscoring its value proposition in providing substantial computational power at a lower economic footprint than other entries in the table.

To further express the comparative advantage of each configuration, we introduce the Performance Index (PI) [9], defined as in (8).

$$PI = \frac{F \times T \times S}{C} \tag{8}$$

Table 6 Performance comparison of state-of-the-art studies in specified metrics

References	Task	Platform	Precision	DPU frequency [MHz]	DPU architecture	Average power [W]	Total resource consumption (%)	FPS	Cost ($)	Image size	PI (1e5)
[26]	Defect detection	XCZU7EV	INT8	300	1×4096	8.755	15.16	13	1678	416×416	0.0137
[27]	Object detection	Ultra96-V2	INT8	300	1×1600	3.29	30.89	53.3	299	320×320	0.18
[28]	Object detection	UltraScale+ MPSoC	N/A	300	N/A	N/A	33.95	110	N/A	123×128	N/A
[29]	Multi-task Imp.	KV260	INT8	300	1×4096	7.34	49.05	37.49	249	288×512	0.44
[30]	Multi-task ADAS	ZU9EG	INT8	N/A	N/A	13.7	48	55	1982	512×256	0.145
[31]	Multi-model multi-mode	ZCU102	N/A	330	4×4096	N/A	28.07	N/A	3374	224×224	N/A
[32]	Multi-task Imp.	SoC	INT8	N/A	N/A	N/A	N/A	10.5	300	572×572	0.229
[33]	Multi-task learning	Xilinx ZU9	INT8	333	2×4096	14.3	28.23	16.2	1982	480×360	0.0422
[34]	Multi-object detection & tracking	ZCU102	Fixed-16	230	4×4096	12.51	15.94	91.65	3374	416×416	0.094
Config. 3 (proposed)	Multi-task learning	KV260	INT8	275	1×4096	7.19	44.8	22.15	249	512×320	**0.582**

where: C = Hardware Cost: Lower values are better. S = Image Size: Typically, larger values indicate higher resolution and potentially better detection capabilities. T = Number of ADAS Tasks: Higher values are better, suggesting more functionalities. F = FPS: Higher values are better, indicating smoother and potentially more reliable performance.

The PI helps quantify the cost-effectiveness and efficiency of different hardware configurations, allowing us to determine which setups offer the best balance of performance to cost. As obvious from Table 6, our proposed solution gives the best PI value as 0.582 (bold in Table 6). Using PI, *Configuration 3* demonstrates superior cost-performance metrics compared to others, making it an optimal choice for scenarios that require high efficiency and budget-friendly solutions. This analysis underscores the importance of selecting the right platform and configuration to balance performance requirements with cost and power constraints.

7 Conclusion and future work

7.1 Summary of findings and implications

This study presents a novel approach to optimizing energy efficiency in configurable DPU architectures, utilizing frequency adjustments on the Zynq UltraScale+ platform to enhance computational efficiency. We have methodically detailed the training and inference processes through our methodical analysis, building upon a robust software foundation and multi-task learning model. Our empirical results, illustrated in Figs. 5 and 6, and summarized in Tables 1, 3, and 4, demonstrate the intricate balance between power consumption, processing speed, and resource utilization across different DPU configurations.

Our detailed examination of various DPU configurations and comparative analyses not only highlights the significant potential for further system optimization but also inspires future research. This potential could enhance the sustainability and efficiency of future embedded AI systems, motivating researchers to explore new avenues in the field.

7.2 Exploring the impact of hardware dependency and voltage scaling on DPU performance

We specifically utilized the AMD Xilinx Vitis AI on the Kria KV260 MPSoC platform, which, while practical, introduces a certain level of hardware dependency that might limit the broader applicability of our findings. Recognizing the importance of this issue, we have noted the necessity of extending our research to various hardware platforms. This expansion will allow us to validate and enhance the adaptability of our optimizations, ensuring that our contributions

can be applied more universally across different technological contexts.

Our current research has also successfully optimized resource utilization and performance through strategic frequency adjustments. However, we acknowledge that we have not entirely wiped into the potential of voltage scaling a promising avenue for further enhancing the energy efficiency of DPUs, especially at higher operational frequencies that typically elevate power consumption. To address this, we plan to integrate adaptive voltage scaling techniques in our future work. This method will balance computational demands with energy conservation more effectively, thereby extending the operational life of our systems and reducing the thermal impacts in energy-constrained environments. These areas, marked for future exploration, are expected to enrich our understanding and implementation of energy efficient DPU configurations, contributing significantly to the advancement of embedded AI systems.

7.3 Directions for future research

Future research should focus on refining DPU configurations to enhance energy efficiency without sacrificing performance. This includes exploring advanced machine learning algorithms and adaptive frequency scaling techniques tailored for real-time applications. Expanding the scope to encompass a broader range of DPU models and configurations will deepen our understanding of how hardware specifications impact energy efficiency.

Additionally, integrating real-world benchmarks into our testing framework is crucial. This will provide deeper insights into the practical implications of DPU optimization, reassuring our audience about the viability of our solutions in real-world scenarios. Ultimately, the ongoing exploration and technological innovations in this field are expected to drive further advancements in sustainable, high-performance computing solutions, with our findings serving as a foundational guide for future developments.

Author contributions G.T. and S.B. wrote the main manuscript text and prepared. All authors reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Yazhini, K., Loganathan, D.: A state of art approaches on deep learning models in healthcare: an application perspective. In: 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, pp. 195–200 (2019). https://doi.org/10.1109/ICOEI.2019.8862730
2. Le Nguyen, T., Do, T.T.H.: Artificial intelligence in healthcare: a new technology benefit for both patients and doctors. In: 2019 Portland International Conference on Management of Engineering and Technology (PICMET), pp. 1–15 (2019). https://api.semanticscholar.org/CorpusID:207973389
3. Pai, N., Ilango, V.: A comparative study on machine learning techniques in assessment of financial portfolios. In: 2020 5th International Conference on Communication and Electronics Systems (ICCES), pp. 876–882 (2020). https://api.semanticscholar.org/CorpusID:220568565
4. Pereira, S.W., Fishman, E.K., Rowe, S.P.: The future is now: how technology and entertainment are transforming education in the artificial intelligence era. J. Am. Coll. Radiol. **19**, 1077–1078 (2022). https://doi.org/10.1016/j.jacr.2022.06.015
5. Umuroglu, Y., Fraser, N.J., Gambardella, G., Blott, M., Leong, P.H.W., Jahre, M., Vissers, K.A.: FINN: a framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (2016). https://api.semanticscholar.org/CorpusID:10530917
6. Duarte, J.M., Han, S., Harris, P.C., Jindariani, S., Kreinar, E., Kreis, B., Ngadiuba, J., Pierini, M., Rivera, R., Tran, N.V., Wu, Z.: Fast inference of deep neural networks in FPGAs for particle physics. J. Instrum. **13**, P07027 (2018)
7. Xilinx: Vitis AI Github. https://github.com/Xilinx/Vitis-AI. Accessed 6 Mar 2024
8. Tatar, G., Bayar, S.: Real-time multi-task ADAS implementation on reconfigurable heterogeneous MPSoC architecture. IEEE Access **11**, 80741–80760 (2023). https://doi.org/10.1109/ACCESS.2023.3300379
9. Tatar, G., Bayar, S., Çiçek, İ.: Real-time multi-learning deep neural network on an MPSoC-FPGA for intelligent vehicles: harnessing hardware acceleration with pipeline. IEEE Trans. Intell. Veh. (2024). https://doi.org/10.1109/TIV.2024.3398215
10. Kalantar, A., Zimmerman, Z., Brisk, P.: FPGA-based acceleration of time series similarity prediction: from cloud to edge. ACM Trans. Reconfigurable Technol. Syst. **16**(1), Article 12, 27 pages (2023). https://doi.org/10.1145/3555810
11. Goel, S., Balakrishnan, M., Sen, R.: EnergyNN: energy estimation for neural network inference tasks on DPU. In: 2021 31st International Conference on Field-Programmable Logic and Applications (FPL), Dresden, Germany, pp. 64–68 (2021). https://doi.org/10.1109/FPL53798.2021.00019
12. Salami, B., et al.: An experimental study of reduced-voltage operation in modern FPGAs for neural network acceleration. In: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, pp. 138–149 (2020). https://doi.org/10.1109/DSN48063.2020.00032
13. Vandendriessche, J., Da Silva, B., Touhafi, A.: Frequency evaluation of the Xilinx DPU towards energy efficiency. In: IECON 2022—48th Annual Conference of the IEEE Industrial Electronics Society, Brussels, Belgium, pp. 1–6 (2022). https://doi.org/10.1109/IECON49645.2022.9968811
14. Li, C., Xu, R., Lv, Y., Zhao, Y., Jing, W.: Edge real-time object detection and DPU-based hardware implementation for optical remote sensing images. Remote Sens. **15**(16), 3975 (2023). https://doi.org/10.3390/rs15163975

15. Zhao, W., Dang, Q., Xia, T., Zhang, J., Zheng, N., Ren, P.: Optimizing FPGA-based DNN accelerator with shared exponential floating-point format. IEEE Trans. Circuits Syst. I Reg. Pap. **70**(11), 4478–4491 (2023). https://doi.org/10.1109/TCSI.2023.3300657
16. Jiang, W., Yu, H., Liu, X., Ha, Y.: Energy efficiency optimization of FPGA-based CNN accelerators with full data reuse and VFS. In: 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, pp. 446–449 (2019). https://doi.org/10.1109/ICECS46596.2019.8964717
17. Xilinx: GitHu—Xilinx/CHaiDNN: HLS based deep neural network accelerator library for Xilinx Ultrascale+ MPSoCs. GitHub. https://github.com/Xilinx/CHaiDNN
18. Xilinx/Vitis-AI: "GitHub," Jun. 06, 2021. https://github.com/Xilinx/Vitis-AI/tree/2.5/model_zoo/model-list. Accessed 5 May 2024
19. Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: BDD100K: a diverse driving dataset for heterogeneous multitask learning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2633–2642 (2018). https://api.semanticscholar.org/CorpusID:215415900
20. Sun, P., et al.: Scalability in perception for autonomous driving: Waymo Open Dataset. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2443–2451 (2019). https://api.semanticscholar.org/CorpusID:209140225
21. Fritsch, J., Kühnl, T., Geiger, A.: A new performance measure and evaluation benchmark for road detection algorithms. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, Netherlands, pp. 1693–1700 (2013). https://doi.org/10.1109/ITSC.2013.6728473
22. Cordts, M., et al.: The Cityscapes dataset for semantic urban scene understanding. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3213–3223 (2016). https://api.semanticscholar.org/CorpusID:502946
23. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision—ECCV 2016. Lecture Notes in Computer Science, vol. 9905. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
24. Caruana, R.: Multitask learning. Mach. Learn. **28**, 41–75 (1997). https://doi.org/10.1023/A:1007379606734
25. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04), Association for Computing Machinery, New York, NY, USA, pp. 109–117 (2004). https://doi.org/10.1145/1014052.1014067
26. Chi, T.-K., Chen, T.-Y., Lin, Y.-C., Lin, T.-L., Zhang, J.-T., Lu, C.-L., Chen, S.-L., Li, K.-C., Abu, P.A.R.: An edge computing system with AMD Xilinx FPGA AI customer platform for advanced driver assistance system. Sensors **24**(10), 3098 (2024). https://doi.org/10.3390/s24103098
27. Machura, M., Danilowicz, M., Kryjak, T.: Embedded object detection with custom LittleNet, FINN and Vitis AI DCNN accelerators. J. Low Power Electron. Appl. **12**(2), 30 (2022). https://doi.org/10.3390/jlpea12020030
28. Hemmati, M., Biglari-Abhari, M., Niar, S.: Adaptive real-time object detection for autonomous driving systems. J. Imaging **8**(4), 106 (2022). https://doi.org/10.3390/jimaging8040106
29. Kalapothas, S., Flamis, G., Kitsos, P.: Efficient edge-AI application deployment for FPGAs. Information **13**(6), 279 (2022). https://doi.org/10.3390/info13060279
30. Peng, J., et al.: Multi-task ADAS system on FPGA. In: 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hsinchu, Taiwan, pp. 171–174 (2019). https://doi.org/10.1109/AICAS.2019.8771615
31. Zhang, J., et al.: An FPGA-based neural network overlay for ADAS supporting multi-model and multi-mode. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, pp. 1–5 (2021). https://doi.org/10.1109/ISCAS51556.2021.9401751
32. Dellinger, F., Boulay, T., Mendoza Barrenechea, D., El-Hachimi, S., Leang, I., Bürger, F.: Multi-task network pruning and embedded optimization for real-time deployment in ADAS (2021). arXiv preprint arXiv:2101.07831
33. Fang, S., et al.: Real-time object detection and semantic segmentation hardware system with deep learning networks. In: 2018 International Conference on Field-Programmable Technology (FPT), pp. 389–392, Naha, Japan (2018). https://doi.org/10.1109/FPT.2018.00081
34. Zhai, J., Li, B., Lv, S., Zhou, Q.: FPGA-based vehicle detection and tracking accelerator. Sensors **23**(4), 2208 (2023). https://doi.org/10.3390/s23042208

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.