Energy-Efficient Fast Object Detection on Edge Devices for IoT Systems

Mas Nurul Achmadiah, Afaroj Ahamad, Chi-Chia Sun, *Member, IEEE*, and Wen-Kai Kuo, *Member, IEEE*

***Abstract*—This article presents an Internet of Things (IoT) application that utilizes an AI classifier for fast-object detection using the frame difference method. This method, with its shorter duration, is the most efficient and suitable for fast-object detection in IoT systems, which require energy-efficient applications compared to end-to-end methods. We have implemented this technique on three edge devices: 1) AMD Alveo^{TM}U50; 2) Jetson Orin Nano; and 3) Hailo-8^{TM}AI Accelerator, and four models with artificial neural networks and transformer models. We examined various classes, including birds, cars, trains, and airplanes. Using the frame difference method, the MobileNet model consistently has high accuracy, low latency, and is highly energy-efficient. YOLOX consistently shows the lowest accuracy, lowest latency, and lowest efficiency. The experimental results show that the proposed algorithm has improved the average accuracy gain by 28.314%, the average efficiency gain by 3.6 times, and the average latency reduction by 39.305% compared to the end-to-end method. Of all these classes, the faster objects are trains and airplanes. Experiments show that the accuracy percentage for trains and airplanes is lower than other categories. So, in tasks that require fast detection and accurate results, end-to-end methods can be a disaster because they cannot handle fast object detection. To improve computational efficiency, we designed our proposed method as a lightweight detection algorithm. It is well suited for applications in IoT systems, especially those that require fast-moving object detection and higher accuracy.**

***Index Terms*—AI classifier, energy efficiency, fast-moving object detection (FMOD), high-latency, real-time performance.**

I. Introduction

THE Internet of Things (IoT) is crucial for advancing computer vision by enabling seamless integration of sensors, devices, and data processing. The global proliferation of IoT over the past decade has facilitated the development of numerous new applications that utilize a wide range of devices and sensors. With the explosive growth of mobile applications and rapid development of advanced wireless technologies under the driving forces of IoT networks, a large number of IoT devices and emerging applications require ultra data rates, low-energy consumption, and spectral efficiency simultaneously [1]. The importance of improving efficiency in IoT applications is to overcome limitations in traditional systems by optimizing resource utilization and energy consumption in a way that optimization can provide innovative solutions to improve the performance of IoT systems in challenging environments [2]. More recently, visual sensors have seen their considerable boom in IoT systems because they are capable of providing richer and more versatile information. IoT networks connect cameras and sensors to edge devices, facilitating real-time data collection and analysis. This connectivity enhances the capabilities of computer vision systems, making them more efficient in tasks like object detection, facial recognition, and anomaly detection. The massive data generated by IoT devices can be processed and analyzed to train and improve machine learning models, driving innovation in automation, smart cities, healthcare, and security [3].

The field of moving object detection is so vast. Researchers have proposed many approaches to the importance of efficiency and accuracy in multitask learning by highlighting that integrating tasks, such as semantic segmentation and camera pose prediction within a single framework, not only improves the learning efficiency but also enhances the generalization ability across tasks [4]. A critical task in object detection is motion estimation. Optical flow is commonly used to estimate the object's motion [5]. Starting with the original algorithms by Lucas and Kanade (LK) [6] as well as Horn and Schunck (HS) [7], gradient-based methods have led to other improved optical flow estimation methods. However, when the image background is cluttered or the detected object is moving at high speed, the accuracy of gradient-based methods will be significantly decreased.

There are three primary categories of techniques for detecting moving objects. One technique is optical flow [8], which establishes an image's optical flow field and looks at the associated pixel's motion vector to identify moving objects. On the other hand, because of the scene's variability, the calculation is intricate and prone to detection errors [9]. However, the end-to-end method offers powerful capabilities and has achieved remarkable success, but it comes with several significant disadvantages. These include high computational and data requirements, lack of interpretability, risks of overfitting, scalability challenges, sensitivity to architecture and hyperparameters, ethical concerns, and maintenance difficulties. Addressing these issues requires advanced techniques,

Received 10 December 2024; accepted 19 January 2025. Date of publication 10 February 2025; date of current version 23 May 2025. This work was supported by the National Science and Technology Council under Grant 113-2221-E-150-026-MY3. *(Corresponding author: Chi-Chia Sun.)*

Mas Nurul Achmadiah is with the SMIM Research Center and the Department of Electro-Optics, National Formosa University, Huwei 632, Taiwan.

Afaroj Ahamad is with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan 320, Taiwan.

Chi-Chia Sun is with the Department of Electrical Engineering, National Taipei University, Taipei 237303, Taiwan (e-mail: chichiasun@mail.ntpu.edu.tw).

Wen-Kai Kuo is with the Department of Electro-Optics, National Formosa University, Huwei 632, Taiwan.

Digital Object Identifier 10.1109/JIOT.2025.3536526

careful design, and ongoing management. Therefore, it cannot deliver optimal performance at high resolutions [10].

The following method is called frame difference [11]. Detection of moving objects from a sequence of frames captured from a static camera is widely performed by the frame difference method. The objective of the approach is to detect the moving objects from the difference between the existing and reference frames [12]. The frame difference method is the standard method of motion detection. This method adopts pixel-based differences to find the moving object. Which compares the pixel information among adjacent frames. When an object passes through the frames, the differences between the frames exceed the threshold [13]. From [14] the study results show the success of the frame difference method, which is more accurate in detecting objects. In the frame difference method, the high resolution of the resulting video does not reduce the resulting performance.

Another essential task in object detection is classification. Real-time classification of fast-moving objects is a challenging task. Although image-based object classification systems have been explored for decades, classifying fast-moving objects in real-time and for a long duration is still challenging. Fast-moving objects often generate dramatic motion blurs in the images captured. The induced motion blurs cause severe image quality degradation, reducing the achievable classification accuracy. In addition, sophisticated image analysis algorithms are generally computationally expensive, and consequently, they are not suited for real-time classification [15].

Many industrial, medical, commercial, and research-related applications depend on computer vision and image processing techniques for real-time object recognition and classification. Central processing units (CPUs) are insufficient for many applications because they cannot quickly process the computations. Algorithms can be implemented in AI accelerators, field-programmable gate arrays (FPGAs), or graphics processing units (GPUs) to shorten the calculation time. Choosing the right hardware accelerator for a given application can be difficult [16].

Several generations of FPGAs, GPUs, and AI accelerators are available, and it is challenging to compare different hardware accelerators due to their technological variations. Previous works have covered the performance and technical aspects of hardware accelerators. Nevertheless, many of these presentations have flaws, such as discussing outdated technology and comparing hardware accelerators at two distinct technological levels without providing enough technical information to help choose an appropriate accelerator [17]. Currently, the most commonly used hardware is the GPU. However, GPUs have significant power consumption and high latency. Some AI processing hardware with low latency, high throughput, and low power include AI Accelerator, FPGA, and Special AI SoC.

Energy efficiency is crucial for edge devices in the IoT due to their typically limited power resources and the vast scale at which they operate [18]. Edge devices, such as sensors, gateways, and microcontrollers, are often deployed in remote or hard-to-reach locations where frequent battery replacement or recharging is impractical. Therefore, optimizing energy usage is essential to prolong device lifespan, reduce maintenance costs, and ensure the reliability of the entire IoT ecosystem. Efficient energy management in edge devices also contributes to the overall sustainability of IoT networks, reducing their carbon footprint and supporting green technology initiatives. Techniques, such as low-power hardware design and intelligent power management algorithms, are critical for minimizing energy consumption [19].

The main contribution of this article is to compare our proposed method to the end-to-end method and implement this technique on three modern edge devices for object detection and classification applications to get the best method. Specifically, we make the following contributions.

1) Comparing our proposed object detection method with end-to-end methods and finding which method has real-time processing, high accuracy, and is highly energy-efficient.
2) Perform testing on different edge devices; first, we tested on the AMD Alveo U50. Alveo is an accelerator product from AMD that uses FPGA technology. The second is Jetson Orin Nano, and the third is Hailo-8^{TM} AI Accelerator.
3) From these two results, we will compare the performance of the method and the capabilities of edge devices in the context of object detection. We categorize and report our evaluation results based on key benchmarks, including object detection accuracy, latency performance, and efficiency gains.

This article is organized as follows. Section I discusses the background of the proposed method. Section II discusses related work, which includes previous research. Section III discusses the proposed method and the steps required. Section IV shows the experimental results of the evaluated devices, which are then analyzed according to the evaluation criteria. Finally, Section V summarizes our work and concludes.

II. Related Work

In this section, we survey the previous literature related to our research. This section involves two parts. The first discusses our proposed algorithm, fast-moving object detection (FMOD), and AI classifier. The second concerns the implementation of FMOD on the AMD Alveo U50, Jetson Orin Nano, and Hailo- 8^{TM} AI accelerators.

A. Fast-Moving Object Detection and AI Classifier

Real-time object detection is crucial for latency-sensitive IoT applications, such as autonomous driving, augmented reality, and intelligent surveillance. These applications require fast and accurate processing of video streams to make timely decisions. Traditional object detection methods, while accurate, often have high end-to-end latency, making them unsuitable for real-time use [20]. Object detection and the capacity to cross current benchmarks remain unresolved issues in deep learning technology. Gradually raising CNN's computational capacity promotes extensive application [21]. The CNN-based pedestrian and localization algorithm proposed that the object

was spotted using a monovision camera, and the distance of the found object was measured. Applied to classify the found objects [22]. In [23] presented a LiDAR-based three-stage GC-net encompassing a pipeline comprising grinding, clustering, and a CNN-based classifier. Leveraging the intelligence trend, cooperative driving automation (CDA) has attended acknowledged events over the last several years.

The first real-time object detection technique was proposed by [24]. However, not all past methods are suitable for fast-moving object recognition and identification. Several investigations have successfully and highly precisely identified fast objects with great speed [25]. The acoustic approach, however, failed to recover absolute visual and relative position. Regarding the image processing technique, fixed cameras are primarily applied in many types of research for fast object identification moving [26]. Though low accuracy and processing speed are significant disadvantages, this study effectively identified and classified objects at high speeds. Other earlier research found moving objects among more intricate backdrops with success. They still need to categorize the objects [27].

The frame differencing method is a straightforward and energy-efficient approach to motion detection that compares pixel intensity differences between consecutive video frames. By examining changes in pixel values, frame differencing directly identifies motion, avoiding the need for extensive preprocessing or feature extraction. Pixels with significant changes are flagged as areas of motion. This minimalistic processing pipeline avoids the computational overhead associated with more complex methods, such as convolutional neural networks (CNNs), used in YOLO or other deep learning models. In [28] frame differencing can efficiently handle simple scenes with static backgrounds. While advanced models require extensive hardware and energy resources to train and infer object categories, frame differencing's task-specific design achieves its objectives with significantly lower energy demands. Compared to background subtraction methods, frame differencing excels in terms of energy efficiency because it does not rely on maintaining or updating a background model. Background subtraction methods often require dynamic adaptation to changing environments, which introduces additional computational steps. In contrast, frame differencing processes only two consecutive frames at a time, making it lightweight and adaptable. This advantage is highlighted in the work of [29] where the authors demonstrate the method's ability to achieve real-time motion detection with minimal hardware requirements.

When compared to deep learning-based methods, the energy efficiency of frame differencing becomes even more apparent. Deep learning models like YOLO require extensive computations across multiple convolutional layers, resulting in high power consumption. These methods also demand specialized hardware, such as GPUs or TPUs to operate in real-time, further increasing energy costs. In contrast, frame differencing can run on low-power microcontrollers without the need for hardware accelerators. In [30] underscore this advantage, emphasizing frame differencing's suitability for resource-constrained environments.

B. Implementation of FMOD on Edge Devices for Energy Efficiency in IoT Systems

FMOD, particularly object detection, is critical for applications requiring real-time analysis and immediate response. Moving object detection enables systems to identify and track objects in dynamic environments quickly, which is essential for applications like autonomous driving, where vehicles must rapidly detect and react to other cars, pedestrians, and obstacles to avoid accidents [31]. In industrial automation, fast object detection allows robots to monitor and interact with swiftly moving items on production lines, enhancing efficiency and precision. Security systems benefit from fast detection by immediately identifying and responding to potential threats, ensuring timely interventions [32]. In [33], the development of energy-efficient Artificial Intelligence (AIoT) with intelligent edge computing is discussed. This article highlights the importance of optimizing energy consumption in edge devices and cloud services when processing AIoT tasks. This article introduces a multilevel intelligent edge framework designed to improve energy efficiency by managing resources between edge devices and the cloud.

Edge devices are indispensable in optimizing the performance and practicality of fast object detection. By processing data locally on devices, such as smart cameras, drones, and mobile phones, edge computing significantly reduces latency, enabling immediate decision-making crucial for real-time applications [34]. This local processing minimizes the need for data transfer to remote servers, conserving bandwidth and ensuring that the system remains operational even with intermittent connectivity. Edge devices also enhance data privacy by keeping sensitive information on the device. Their role in fast object detection is vital in scenarios where speed and reliability are paramount, making AI applications more efficient, responsive, and capable of functioning independently of constant cloud connectivity [35].

CPUs and GPUs each have strengths and weaknesses, influencing their suitability for various computing tasks. CPUs are versatile and excel in single-threaded performance, making them ideal for general-purpose computing and tasks that require complex decision-making and low latency. However, they need help with highly parallel computations due to their limited number of cores and higher power consumption. In contrast, GPUs are designed for parallel processing with thousands of smaller cores, making them excellent for tasks like image processing, scientific simulations, and deep learning. Despite their high throughput and efficiency in handling large-scale parallel tasks, GPUs consume significant power, have higher latency for single-threaded tasks, and are often expensive. However, GPU platforms are power hungry, and due to their high cost [36], ASIC's time-to-market weakness is unacceptable on AI Accelerator edge computing devices [37].

Hailo AI accelerators offer a specialized solution to address the limitations of both CPUs and GPUs, especially in edge AI applications. These accelerators are optimized for deep learning inference, providing high performance and power efficiency in a compact form factor. They are suitable for real-time AI tasks in low-power environments like IoT devices and autonomous systems. While Hailo AI accelerators offer

significant advantages in power efficiency and latency for AI inference, they specialize in these tasks and may require additional integration effort and optimization knowledge. Overall, Hailo AI accelerators complement the capabilities of CPUs and GPUs by providing an efficient and effective solution for specific AI-driven applications [38]. FPGAs are highly configurable, allowing for custom hardware configurations optimized for specific tasks, leading to significant performance and power efficiency improvements. FPGAs can be tailored to handle parallel processing tasks with lower latency and power consumption than GPUs, making them suitable for applications where these factors are critical, such as real-time signal processing, embedded systems, and specialized AI inference tasks. Additionally, FPGAs can be reprogrammed to adapt to evolving computational requirements, providing a level of flexibility that fixed-function Application-Specific Integrated Circuits (ASICs) lack [39].

However, the complexity of FPGA programming and integration can be a disadvantage, requiring specialized knowledge and development time to leverage their capabilities thoroughly. Despite this, the adaptability and efficiency of FPGAs make them an essential component in scenarios where traditional CPUs and GPUs fall short. Apart from this, FPGA [40] has properties, such as low operation power, reconfigurability, customizable data flow, and data width. Hence, FPGA became a delightful platform for accelerating DNN architecture. In [41] suggested an FPGA scalable processor, i.e., a deep learning accelerator unit (DLAU), in which they use a technique, such as FIFO buffer and pipelines. This approach offers intercommunication reduction to DRAM while enabling the reuse of computing units when implementing neural networks. Farabet et al. [42] suggested a large-scale CNN implementation methodology that involved constructing manifold tiles inside an FPGA processor. However, due to the limitation of FPGA hardware, most prior studies focus on simplifying the neural network weight coefficient or architecture [43].

It quantizes the parameters of the neural network to reduce the capacity required by memory and the logic gates needed by the architecture. Several studies have proposed methods to reduce memory and logic resource requirements with a slight reduction in accuracy in exchange [44]. There are also prior studies enhancing the power consumption efficiency and performance of FPGA parallel processing with neural network architecture [45]. However, most prior research did not discuss and consider practical applications. Moreover, it is usually challenging to select and detect objects within the input frame directly, therefore, this research article will also propose an algorithm to detect objects of interest within the input frame.

Furthermore, the NVIDIA Jetson Orin Nano integrates a powerful CPU and GPU architecture tailored for AI workloads, providing efficient parallel processing and real-time performance with lower power consumption than traditional GPUs. This makes it particularly well suited for AI inference tasks at the edge, where power efficiency and compact form factor are critical. Additionally, the Jetson Orin Nano includes AI-specific accelerators and supports a wide range of AI frameworks, simplifying the deployment of advanced AI models in edge devices. By leveraging the capabilities of the Jetson Orin Nano, developers can achieve high-performance AI processing with improved energy efficiency and reduced latency, overcoming the traditional limitations of CPUs and GPUs. This makes it an excellent choice for applications, such as autonomous machines, robotics, intelligent cameras, and other AI-driven embedded systems where computational power and energy efficiency are paramount [46].

The NVIDIA Jetson Orin Nano offers significant advantages over Hailo AI accelerators and FPGAs, particularly for edge AI applications. Unlike Hailo AI accelerators, the Jetson Orin Nano benefits from NVIDIA's extensive software ecosystem, including the JetPack SDK and support for popular AI frameworks, such as TensorFlow and PyTorch. This makes development more straightforward and accessible. It combines CPU, GPU, and AI accelerators on a single chip, providing a versatile and integrated solution that can handle various tasks, from AI inference to general-purpose computing [46]. Compared to FPGAs, the Jetson Orin Nano is more accessible to program and deploy, thanks to NVIDIA's high-level programming languages and comprehensive development tools. It is optimized specifically for AI workloads, offering efficient performance for neural network inference and real-time processing. Additionally, its power efficiency and ability to handle multiple tasks simultaneously make it ideal for edge applications with critical performance and power constraints. The Jetson Orin Nano's blend of powerful hardware, extensive software support, and ease of use makes it a compelling choice for implementing advanced AI capabilities at the edge. This research will apply the proposed method to each device so that the results can determine the best combination of evaluations, including object detection accuracy, latency performance, and increased efficiency [47].

III. Proposed Method

The frame difference method combined with a lightweight AI classification algorithm is well-suited for FMOD in energy-constrained IoT applications due to its efficiency and performance. This method is lightweight, making it ideal for edge devices with limited power availability, such as sensors, cameras, and IoT gateways. By combining an efficient AI classification algorithm, this approach improves detection accuracy while maintaining low latency and energy consumption.

This research involved three significant procedures: 1) movement detection; 2) pre-processing; and 3) CNN or Transformer Classifier. Fig. 1 illustrated the proposed algorithm. Although movement detection and pre-processing procedures can be grouped, as they both employ and are based on image morphology techniques, dividing them into two methods is preferable because a decision check separates them, contributing excellent features to the algorithm flow. The algorithm has a loop-like structure, presenting the complete loop of processing the 3-channel color video input and output of the resulting video, including features. The image region in which the object's presence is marked with a bound box, and the class of the object is written above the box. Each video frame input represents a loop instance element of the algorithm. The algorithm starts and runs through a loop

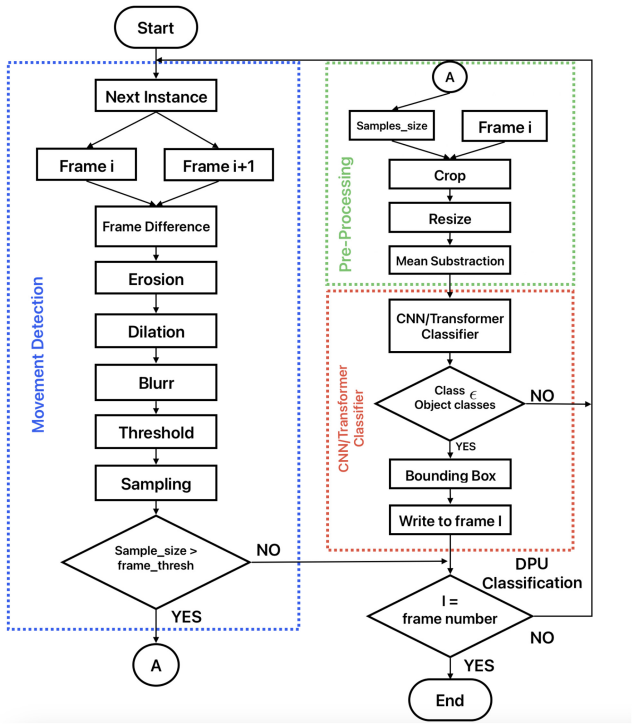

Fig. 1. Flow-chart of proposed algorithm.

Algorithm 1 Feed Image

video **read** *vFrame* → **Read a store a video frame**
Im1 ← *vFrame*
Im2 ← *vFrame*
for *f* ← 0, *frame Number* **do** → Loop whole video frame
    Im1 ← *Im2* → *First input image*
    video read vFrame → *Read next video frame*
    Im1 ← *vFrame* → *Second input image*
    ImgProc(*Im*1, *Im*2) → Execute image morphology
end for

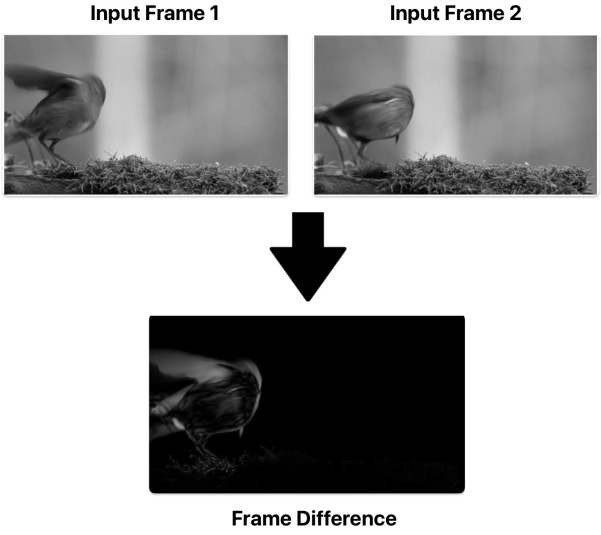

Fig. 2. Frame difference process.

Fig. 3. Image dilation process.

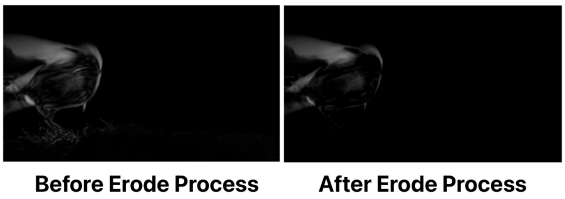

Fig. 4. Image erosion process.

instance. Afterward, at the end of the loop instance, we will check if the current frame counter has already reached the maximum value or not, which is the total number of frames of the inputted video. If the frame counter has not reached the absolute frame number, the algorithm returns to the starting point and processes the next loop instance.

A. Movement Detection Method

The movement detection process uses image morphology as its foundation. Image morphology is the method of extracting useful information from an image or a video frame using a step-by-step procedure.

Algorithm 1 illustrated each image's morphological steps. It involves covertly 3-color-channeling two consecutive frames to grayscale and feeding them as input.

The next step takes the frame's absolute abstraction of the two inputted images, i.e., frame difference, and gets one resulting image (see Fig. 2). After the frame difference process, the image might already have a glimpse of which area contains the movement. However, the background still contributes a significant amount of noise. This noise may be caused by the subtle movement of the grass in the background, rather than by the object, and should thus be eliminated. For the goal of noise reduction, morphological opening is used.

The morphological opening of an image consists of two consecutive operations: an image erosion followed by an image dilation, both of which use the same structuring element. Figs. 3 and 4 are the image dilation and erosion processes.

The next step is called image blurring (see fig. 5) the resulting image of the previous step is convoluted with a low-pass filter kernel of high-frequency content. After image blurring, the result is compared with the threshold value, and it is 0 for smaller than the threshold; otherwise, it is 255, i.e., it is called thresholding (see Fig. 6). The final step of image morphology is sampling, which involves finding the region of interest (ROI) rectangle from a frame.

Algorithm 2 illustrates pseudo-code to ROI. Firstly, find all points whose value equals 1, record their X and Y coordinates into the arrays, and determine the maximum and minimum of each array, respectively. Then the loop executes through both coordinate arrays one by one to feed maximum and minimum

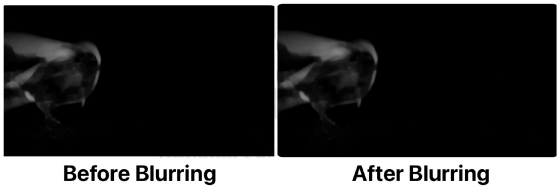

Fig. 5. Image blurring process.

Fig. 6. Thresholding process.

values, respectively. The algorithm will calculate the ROI's cropping parameters at the end.

B. Pre-Processing

After getting the result of the Movement Detection procedure and getting a greater value than the threshold value, the algorithm will continue to execute the Pre-processing procedure. The processes of pre-processing consist of the following.

1) Positioning the frame on a predetermined frame sample. Cropping the desired region from the inputted 3-channel image, After the ROI parameters are determined, the program will crop out the ROI with these parameters from the input BGR 3-channel video frame input. The kernel-internal input-output type for accelerated kernel arguments of Alveo is mostly stream-like data; therefore, the stream-architect implementation typed cropping will be used instead of the memory-mapped architecture implementation.
2) The cropping process will vary in size, so to input the cropped region into the neural network, resize it to the standard input size of the ImageNet model, which is $224 \times 224 \times 3$, with the exception of the Inception-v4 model, whose input size is $299 \times 299 \times 3$.
3) The process involves resizing the resulting image to a resolution of 224×224 and carrying out appropriate pre-processing tasks related to neural networks. The resize function offers a variety of interpolation settings, enabling the calculation of the intermediate point between two original pixels in the resized image. Interpolation settings include the nearest neighbor, bilinear, bicubic, area relation, etc. To preserve the 3-channel color feature while still enhancing performance, the result uses bilinear interpolation for the resizing function.
4) Before being fed for inference into a deep neural network, the image must be broken down into an array to be able to be fed into the inference graph of the neural network, in complement with the serialization step, which has several extra steps of pre-processing, depending on the neural network model.

Algorithm 2 Find ROI Arguments

$X_{min} \leftarrow maxRow$ - 1 $\rightarrow$ Initiate with opposite value
$X_{max} \leftarrow 0$
$Y_{min} \leftarrow maxCol$ - 1
$Y_{max} \leftarrow 0$
for r $\leftarrow$ 0, *maxRow* **do** $\rightarrow$ Search pixels with positive value
  for c $\leftarrow$ 0, *maxCol* **do**
    if $Mat(r, c) = True$ **then**
      X_{array} **append** c
      Y_{array} **append** r
      $pointCnt \leftarrow pointCnt + 1$
    end if
  end for
end for
if $pointCnt = 0$ **then**
    $X_{position} \leftarrow 0$
    $Y_{position} \leftarrow 0$
    $X_{size} \leftarrow 0$
    $Y_{size} \leftarrow 0$
return $\rightarrow$ No point found
end if
$X_{min} \leftarrow X_{array}[0] \rightarrow$ Assign with first non-zero
$X_{max} \leftarrow X_{array}[0]$
$Y_{min} \leftarrow Y_{array}[0]$
$Y_{max} \leftarrow Y_{array}[0]$
for $i \leftarrow 0$, *pointCnt***do** $\rightarrow$ Colomn searching loop
    if $X_{array}\ [i] > X_{max}$ **then**
      $X_{min} \leftarrow X_{array}\ [i]$
    else
      if $X_{array}\ [i] > X_{max}$ **then**
        $X_{max} \leftarrow X_{array}\ [i]$
      end if
    end if
end for
for i $\leftarrow$ 0, *pointCnt* **do** $\rightarrow$ Row searching loop
    if $Y_{array}\ [i] < Y_{min}$ **then**
      $Y_{min} \leftarrow Y_{array}\ [i]$
    else
      if $Y_{array}\ [i] > Y_{max}$ **then**
        $Y_{max} \leftarrow Y_{array}\ [i]$
      end if
end if
end for
$X_{position} \leftarrow X_{min} \rightarrow$ Update with final values
$Y_{position} \leftarrow Y_{min}$
$X_{size} \leftarrow X_{max} - X_{min} + 1$
$Y_{size} \leftarrow Y_{max}$ - $Y_{min} + 1$

C. CNN or Transformer Classifier

This research employs four models with artificial neural networks and transformer models to recognize and classify objects and compare them to end-to-end methods. These models used the ImageNet dataset [48], which is a large-scale visual database designed to advance research in computer vision and machine learning. The dataset contains over 14 million images, spanning more than 21 000 categories, with a subset

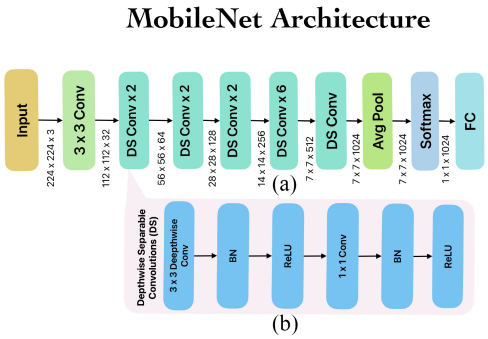

Fig. 7. Architecture diagram of mobilenet.

of 1 000 categories often used for the ImageNet large scale visual recognition challenge (ILSVRC). This subset includes approximately 1.2 million training images, 50 000 validation images, and 100 000 test images, making it a comprehensive resource for evaluating machine learning models. The specific models are ResNet50, MobileNet, Inception-v4, and ViT Base.

1) Architecture of MobileNet: MobileNet is specifically optimized for high efficiency and performance on mobile and embedded devices. Fig. 7 of the document illustrates the structure of the MobileNet architecture, highlighting its efficiency and modular design. Part (A) shows the overall architecture, which includes convolutional layers for feature extraction, depthwise separable convolutions (DS) to significantly reduce computational complexity, average pooling for spatial reduction, fully connected layers to map features to output classes, and a softmax layer for generating class probabilities. Part (B) provides a detailed view of the DS process, which splits standard convolutions into two operations: depthwise convolution, which applies a filter to each input channel independently, and pointwise convolution (1×1), which combines these filtered outputs. The DS layer also incorporates batch normalization (BN) to stabilize and accelerate training and a rectified linear unit (ReLU) for non-linear activation, enhancing the network's ability to model complex patterns.

2) Architecture of Inception-V4: Inception V4 enhances the Inception architecture by incorporating an inception module that includes parallel convolutional layers of varying widths. In Fig. 8 of the document, the architecture of the Inception-v4 model is illustrated. It includes several interconnected layers designed to optimize deep learning performance. The core components are the Inception blocks, which feature residual connections for efficient gradient flow and improved training stability. These blocks combine various filter sizes in parallel, enabling the model to extract features at multiple scales. The architecture integrates BN layers to stabilize learning and reduce overfitting, and residual scaling is applied to prevent gradients from vanishing in deeper networks. Additionally, the model incorporates hyperparameter tuning for optimization and performance improvement, which is further enhanced by using Bayesian optimization. The diagram showcases the systematic arrangement of these components to achieve robust feature extraction and classification.

3) Architecture of ResNet50: ResNet50 employs residual learning with 50 layers, incorporating skip connections to facilitate the smooth passage of gradients throughout the network, thus mitigating the issue of vanishing gradients in deep networks. Fig. 9 is the architecture of the ResNet-50 model, a deep CNN designed to address the degradation problem in training deep networks. The diagram highlights the use of residual blocks, which are the core innovation in ResNet. These blocks introduce shortcut connections that bypass one or more layers, facilitating efficient training and ensuring that deeper networks can achieve better performance without degradation. The figure differentiates between two types of shortcut connections: identity shortcuts, used when input and output dimensions are the same, and projection shortcuts, used to match differing dimensions. Downsampling is achieved between blocks with a stride of 2, enabling spatial reduction and efficient feature extraction. The structure demonstrates how ResNet-50 balances depth and computational efficiency, making it highly effective for tasks, such as image classification.

4) Architecture of Vision Transformers (ViT): Vision Transformers (ViT), especially ViT Base, signify a transition from convolutional networks to transformer-based structures for visual applications. This model employs a self-attention technique to collect and incorporate global dependencies in input images effectively. The ViT Base model provides greater precision; albeit, this comes with the drawback of longer processing time and higher computational demands. Fig. 10 is an overview of the ViT architecture. It demonstrates how images are processed as sequences of patches rather than full 2D structures. The process begins by dividing the input image into fixed-size patches, flattening them, and applying a linear embedding to each patch to project it into a consistent vector space. Positional embeddings are added to these patch embeddings to encode spatial information, ensuring the model retains awareness of patch positions within the original image. These embeddings are then fed into a standard Transformer encoder, which comprises layers of multihead self-attention and feed-forward networks, with residual connections and normalization applied throughout. This architecture leverages the scalability and effectiveness of transformers, originally designed for NLP tasks, for image recognition.

MobileNet is designed to be efficient and low-latency in resource-constrained environments. Inception V4 focuses on achieving high accuracy using more computational power. ResNet50 strikes a balance between depth and efficiency. The ViT model offers advanced accuracy with scalable complexity, making it suitable for applications that require a thorough understanding of global features. Each model has been trained using ImageNet as the base dataset. The reason for choosing MobileNet, Inception V4, ResNet50, and several ViTs is that they are leading neural network designs developed to achieve various trade-offs between accuracy, latency, and efficiency in computer vision tasks.

D. Hardware Deployment

1) AMD Alveo U50 and Hailo AI Accelerator: Fig. 11 illustrates the structure of the object detection and classification system implemented on the Alveo U50 and Hailo-8 AI Accelerator. Both of these devices possess identical processing

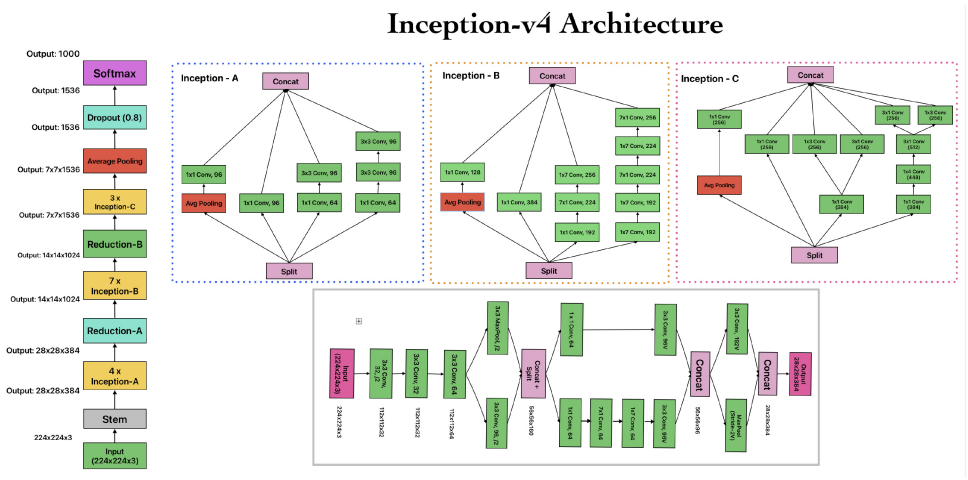

Fig. 8. Architecture diagram of inception-v4.

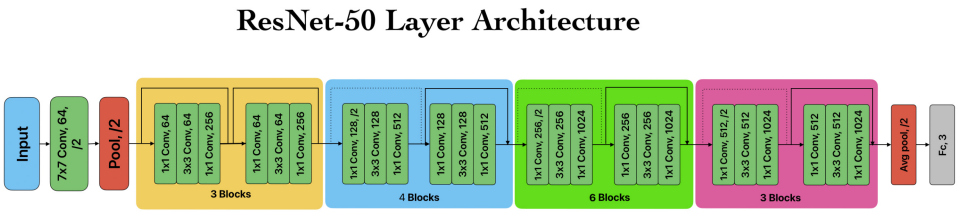

Fig. 9. Architecture diagram of ResNet50.

stages in terms of their architectural design. During the second processing phase, the system divides into two main components: 1) the host PC and 2) the device, specifically the AMD Alveo U50/Hailo-8 accelerator. The host PC is an X86 CPU, renowned for its robust performance and adaptability in managing diverse computer activities, which equips the host PC. The input video, the X86 CPU carries out preprocessing operations to prepare the video data for next processing. This step may involve analyzing the video stream, implementing fundamental improvements, or refining the data to ensure it is in an optimal state for subsequent processing by AI accelerators. In addition to the X86 processor, PC Host also utilizes its memory subsystem. Visual data is temporarily stored and buffered in the memory component during the initial processing step.

This ensures seamless data transmission and enables the processor to quickly retrieve critical data without experiencing noticeable delays. After the host PC finishes its initial processing, it transfers the data to either the Alveo U50 or Hailo-8 for additional processing. High-performance artificial intelligence (AI) applications, specifically on both devices, make them well-suited for demanding video processing workloads. The system comprises a motion detection unit and an AI classifier processing module, both of which closely interact with the high-bandwidth memory (HBM). The Motion Detection Unit is responsible for identifying and tracking motion within video frames. The device utilizes sophisticated algorithms to identify and examine motion, which is crucial for tasks, such as surveillance, automated video editing, and activity recognition. The AMD Alveo U50 or FPGA-based designs are highly

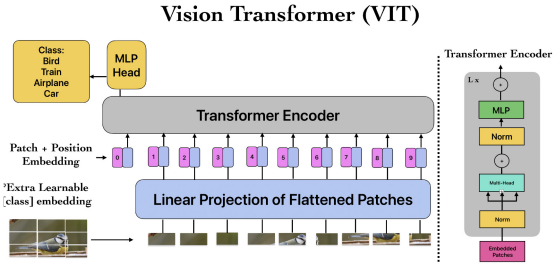

Fig. 10. Architecture diagram of VIT.

suitable for providing the substantial computational power and precision necessary for accurate motion detection. HBM stores and handles the data after motion detection.

This form of memory is crucial for efficiently managing vast quantities of data at a rapid pace, guaranteeing that processing units can quickly retrieve the necessary data. HBM enhances the speed of both reading and writing data, reduces delays, and offers the ability to analyze data in real time. The AI Classifier Processing module receives the video data once it detects motion. This module is responsible for executing a proposed algorithm model in order to categorize the identified items. The synergy among the Motion Detection Unit, AI Classifier Processing Module, and High Bandwidth Memory guarantees a seamless and effective data processing workflow. The AI classifier sends the analyzed video back to the host PC. Subsequently, the host PC does the conclusive processing task of categorizing the identified objects.

2) NVIDIA Jetson Orin Nano: The Fig. 12 provides a detailed representation of the architecture of the Jetson Orin

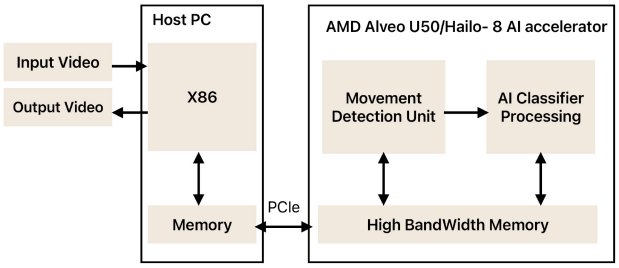

Fig. 11. System architecture of Hailo and AMD Alveo *TM* U50.

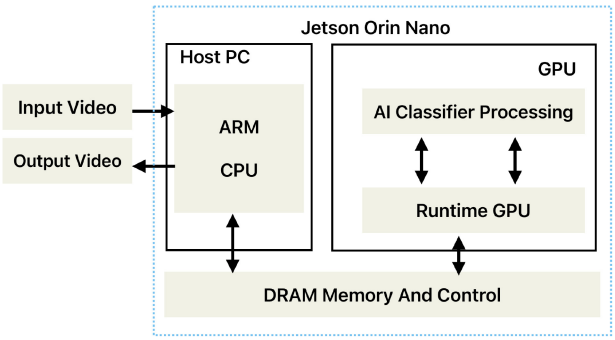

Fig. 12. System architecture of Jetson Orin Nano.

Nano-based video processing system for object detection. It illustrates the interconnections among different components to accomplish effective video processing and AI classification. The system comprises two primary components: 1) the host PC and 2) the GPU, both of which have significant roles in the whole workflow. The host PC receives the video input first.

The ARM components and the host PC's CPU initially process the video. ARM processors specifically design themselves to efficiently handle fundamental processing tasks, offering a harmonious blend of performance and power efficiency. On the other hand, the CPU oversees complex computational processes, ensuring adequate preprocessing of the input video before it reaches the Jetson Orin Nano. The purpose of this pre-processing stage is to prepare the data for further AI processing. Once the host PC completes its initial processing, it transmits the data to the GPU via DRAM memory and control. GPUs are specialized modules specifically intended for AI and deep learning applications, making them well-suited for activities like real-time video processing and object detection. In addition, the memory and DRAM control modules have a significant impact.

This module facilitates seamless data transmission across different system elements, offering the necessary bandwidth and memory administration for fast processing. The GPU is responsible for processing the Jetson Orin Nano's core. The GPU consists of two primary components: 1) AI classifier processing and 2) GPU runtime activities. Specifically designed for executing complex deep learning models and neural networks, the AI Classifier Processing component serves a variety of tasks, such as object detection and other AI-driven video analysis activities. This component utilizes the parallel processing capabilities of GPUs to effectively manage the intricate calculations needed for contemporary AI algorithms.

The GPU Runtime section is responsible for overseeing real-time GPU operations. This entails performing the requisite computations and guaranteeing the timely delivery of the processed data. The interplay between the AI classifier processing and GPU runtime portions is crucial, as it enables the system to do complex AI tasks without sacrificing the real-time performance demanded by applications.After the GPU finishes its processing jobs, it produces a video output. The GPU then transmits the resulting video back to the host PC. The host PC receives the data and then performs the final processing of item detection and classification. The system's division of labor and efficient data handling make it well-suited for real-time video processing and AI applications, guaranteeing exceptional performance and dependability. The interactions among these components showcase the intricate nature and effectiveness of contemporary video processing systems, which are capable of flawlessly managing resource-intensive AI tasks.

IV. Experimental Result

This experiment uses four classes for object detection, such as birds, trains, airplanes, and cars. We are conducting experiments with our proposed method with four models, such as MobileNet [49], ResNet50 [50], Inception-v4 [51], and ViT Base [52] using datasets from Imagenet [48], and YOLOX [53] using datasets from MS Coco [54]. Using some of the topologies, MobileNet, InceptionV4, ResNet50, ViT Base, and YOLOX. The tested data performance is accuracy (Acc%), latency (ms), energy consumption (Joule), and efficiency (%/mW). To calculate efficiency we use the formula from [55]. At Hailo-8 AI Accelerator and AMD Alveo U50, we use PowerTOP in the process of measuring latency, power, and energy during object detection tasks. PowerTOP is a Linux-based software developed by Intel to monitor the power consumption of computers or embedded systems. At Jetson Orin Nano we use jtop. Jtop is a monitoring tool specifically designed for NVIDIA Jetson devices, such as the Jetson Orin Nano. Same with PowerTOP, it can measure latency, power, and energy during object detection tasks.

We also consider the size of the video test we use. Then, we will implement it on three devices: the Hailo-8 AI Accelerator, the Jetson Orin Nano, and the AMD Alveo U50.

A. *Hailo - 8^{TM} AI Accelerator's Performance*

The Table I presents an experimental result of the method when implemented in the Hailo-8^{TM} AI Accelerator. The Birds class, when utilizing the proposed method, obtains an accuracy of 92.6% with MobileNet. MobileNet also has the lowest latency of 35.63 ms and the lowest energy consumption of 1,221 J. As a result, it has the highest efficiency of 0.1731 at a resolution of 3840 × 2160. The ViT Base model attains the utmost accuracy of 94.9% while maintaining a reasonable level of latency and energy usage. However, InceptionV4 and ResNet50 demonstrate increased latencies and energy consumptions, resulting in reduced efficiency. The Bird class for object detection in YOLOX demonstrates a precision of

TABLE I
EVALUATION OF THE HAILO - 8^{TM} AI ACCELERATOR'S PERFORMANCE

Class Name	Method	Topology	Acc(%)	Latency (ms)	Energy (Joule)	Efficiency (%/msW)	Video Resolution
Bird	Proposed Object Detection Method	MobileNet	92.6	35.63	1,221	0.1731	3840 x 2160
		InceptionV4	84.2	47.12	1,751	0.0541	
		ResNet50	72.3	49.16	1,184	0.0806	
		ViT Base	94.9	43.36	1,750	0.0875	
Bird	Object Detection	YOLOX	67.4	48.62	2,847	0.0393	3840 x 2160
Train	Proposed Object Detection Method	MobileNet	97.7	43.80	4,849	0.046	4096 x 2016
		InceptionV4	57.4	65.23	5,892	0.025	
		ResNet50	95.7	56.71	5,094	0.045	
		ViT Base	84.4	83.93	7.580	0.0370	
Train	Object Detection	YOLOX	57.4	49.54	10,880	0.0354	4096 x 2016
Airplane	Proposed Object Detection Method	MobileNet	57.4	14.32	69.34	0.1168	3840 x 2160
		InceptionV4	44.4	16.85	67.41	0.1234	
		ResNet50	28.4	15.68	69.00	0.0631	
		ViT Base	45.6	19.01	108.41	0.1644	
Airplane	Object Detection	YOLOX	25.9	39.80	274.65	0.02762	3840 x 2160
Car	Proposed Object Detection Method	MobileNet	98.4	22.07	241.05	0.1168	4096 x 2016
		InceptionV4	94.2	33.74	907.39	0.1582	
		ResNet50	99.8	27.91	245.25	0.1825	
		ViT Base	97.7	60.10	308.98	0.1410	
Car	Object Detection	YOLOX	93.0	49.80	794.90	0.08658	4096 x 2016

67.4% with a latency of 48.62 ms and energy consumption of 2 847 J, yielding an efficiency of 0.0393 at a resolution of 3840 × 2160.

The Train class achieves the maximum accuracy (97.7%) with a latency of 43.801 ms and an energy usage of 4 849 J with MobileNet. InceptionV4 and ResNet50 exhibit elevated energy usage and latencies, leading to reduced efficiency. The Train class for Object Detection using YOLOX achieves an accuracy of 57.4% with a latency of 49.54 ms and energy consumption of 10 880 J. This results in an efficiency of 0.0354 at a resolution of 4096 × 2016.

Then, the Airplane class obtains the maximum accuracy (57.4%) with the lowest energy usage (69.34 J) and latency (14.32 ms) using MobileNet. Among all the models, ViT Base exhibits the most significant energy usage, amounting to 108.41 Joules, as well as the longest latency, at 19.01 ms. The YOLOX model in the Airplane class for object detection achieves an accuracy of 25.9%, with a latency of 39.80 ms and energy consumption of 274.65 J. This results in an efficiency of 0.02762 at a resolution of 3840 × 2160.

The Car class achieves the maximum accuracy (99.8%) with a latency of 27.91 ms and an energy usage of 245.25 J with ResNet50. MobileNet exhibits a modest level of energy usage, specifically 241.05 Joules, and latency, specifically 22.07 ms. The Car class for Object Detection using YOLOX achieves a precision of 93.0% with a response time of 49.80 ms and energy usage of 794.90 joules, resulting in an efficiency of 0.08658 at a resolution of 4096 × 2016.

In general, the performance of the Hailo-8^{TM} AI Accelerator varies greatly depending on the structure and objective, with MobileNet often being a good choice due to its balanced accuracy, latency, energy consumption, and efficiency. The combination of frame difference and classification typically achieves greater levels of accuracy compared to object detection. Additionally, MobileNet consistently demonstrates low latency and energy consumption across different classes, making it highly efficient. On the other hand, YOLOX, although it excels at object detection, has a tendency to exhibit increased energy consumption and latency.

B. Jetson's Performance

The Table II assesses the performance of the method when implemented in Jetson in multiple classes. The Bird class, MobileNet achieves a flawless accuracy of 100%. It also has a low latency of 41.38 ms and consumes just 0.3937 J of energy. This makes MobileNet the most efficient option, with an efficiency of 0.8332% /ms W. ViT Base demonstrates a remarkable accuracy of 93.72% with reasonable latency and energy usage, whereas InceptionV4 and ResNet50 exhibit lesser accuracies and modest efficiency. For object detection in the Bird class, YOLOX demonstrates a precision of 66.92% with a latency of 61.67 milliseconds and power usage of 0.674 joules, leading to an efficiency of 0.4203% /ms W at a resolution of 3840 × 2160.

The Trains class, using the proposed method, demonstrates MobileNet's consistent accuracy of 100%. This is achieved with a latency of 53.35 ms and remarkably low energy consumption of 0.0596 J. ResNet50 achieves a high level of accuracy (97.70%), but it also exhibits increased latency and energy consumption, and ViT Base demonstrates middling performance metrics. The Train class for Object Detection demonstrates that YOLOX attains a precision of 55.39% with a latency of 55.44 ms and power consumption of 0.0857 J, yielding an efficiency of 0.3568%/mW at a resolution of 4096 × 2016.

The Airplanes class obtains an accuracy of 60.23% with a latency of 53.36 ms and an energy usage of 0.035 J when employing MobileNet. ViT Base demonstrates the highest levels of latency and energy consumption, resulting in reduced efficiency. The Airplane class in Object Detection using YOLOX achieves an accuracy of 23.11% with a latency of 76.74 ms and energy consumption of 0.063 J. This results in an efficiency of 0.0137 at a resolution of 3840 × 2160.

TABLE II
EVALUATION OF THE JETSON'S PERFORMANCE

Class Name	Method	Topology	Acc(%)	Latency (ms)	Energy (Joule)	Efficiency (%/msW)	Video Resolution
Bird	Proposed Object Detection Method	MobileNet	100	41.38	0.3937	0.8332	3840 x 2160
		InceptionV4	83.68	52.43	0.59436	0.5319	
		ResNet50	75.99	61.57	0.4864	0.5239	
		ViT Base	93.72	63.58	0.53064	0.4466	
Bird	Object Detection	YOLOX	66.92	61.67	0.674	0.4203	3840 x 2160
Train	Proposed Object Detection Method	MobileNet	100	53.35	0.0596	0.6693	4096 x 2016
		InceptionV4	58.62	61.49	0.1094	0.3177	
		ResNet50	97.70	81.33	0.0875	0.3533	
		ViT Base	87.36	203.5	0.0624	0.134	
Train	Object Detection	YOLOX	55.39	55.44	0.0857	0.3568	4096 x 2016
Airplane	Proposed Object Detection Method	MobileNet	60.23	53.36	0.035	0.1044	3840 x 2160
		InceptionV4	49.99	82.31	0.090	0.0216	
		ResNet50	29.1	65.42	0.060	0.0267	
		ViT Base	46.01	67.05	0.048	0.0581	
Airplane	Object Detection	YOLOX	23.11	76.74	0.063	0.0137	3840 x 2160
Car	Proposed Object Detection Method	MobileNet	100	28.02	0.0807	1.151	4096 x 2016
		InceptionV4	100	32.79	0.2261	0.8712	
		ResNet50	100	33	0.136	0.8912	
		ViT Base	95.77	56.59	0.1432	0.4701	
Car	Object Detection	YOLOX	91.11	45.48	0.193	0.6907	4096 x 2016

The Car class achieves perfect accuracy (100%) with a latency of 28.02 ms and energy usage of 0.0807 J, resulting in the maximum efficiency (1.151%/mW) at a resolution of 4096 × 2016, as demonstrated by MobileNet. The Car class for Object Detection in YOLOX demonstrates a precision of 91.11% with a response time of 45.48 ms and power consumption of 0.193 joules, yielding an efficiency of 0.6907 at a resolution of 4096 × 2016. The performance of Jetson's device is influenced by its structure and objectives, with MobileNet continuously demonstrating high accuracy, low latency, low power usage, and high energy efficiency across many classes. The combination of frame difference and classification typically achieves greater levels of accuracy compared to object detection. MobileNet, in particular, demonstrates remarkable performance in terms of both accuracy and efficiency. Although YOLOX has strong performance in object detection tasks, it is characterized by longer response times and worse operational efficiencies when compared to alternative topologies.

C. AMD Alveo U50's Performance

The Tables III show the results of how well the method was implemented in an AMD Alveo *TM* U50 and could identify four different types of objects: birds, trains, airplanes, and cars. The ViT model's performance evaluation results are not presently accessible on Alveo devices because Vitis AI does not currently support it. Therefore, testing is restricted to four models for Alveo devices: MobileNet, InceptionV4, Resnet50, and YOLOX.

For the Bird class, MobileNet achieves high accuracy (94.39%) with low latency (7.74 ms) and energy consumption (43.74 J), leading to the highest efficiency (0.8935%/mW) at a resolution of 3840 × 2160. InceptionV4 and ResNet50 show moderate accuracies and latencies with higher energy consumptions, resulting in lower efficiencies. In the Bird class for Object Detection, YOLOX achieves an accuracy of 69.11% with a latency of 16.37 ms and energy consumption of 347.75 J, resulting in an efficiency of 0.1189 at a resolution of 3840 × 2160.

For the Trains class, MobileNet shows high accuracy (93%) with low latency (8.44 ms) and minimal energy consumption (30.91 J). ResNet50 follows with a high accuracy (90.86%) but slightly higher latency and energy consumption. InceptionV4 demonstrates lower accuracy and efficiency compared to MobileNet and ResNet50. In the Train class for Object Detection, YOLOX achieves an accuracy of 51.52% with a latency of 13.15 ms and energy consumption of 318.89 J, resulting in an efficiency of 0.135 at a resolution of 4096 × 2016.

For the Airplanes class, MobileNet achieves moderate accuracy (59.32%) with low latency (8.39 ms) and energy consumption (25.12 J). InceptionV4 and ResNet50 show lower accuracies and efficiencies, with higher latencies and energy consumptions. In the Airplane class for Object Detection, YOLOX achieves an accuracy of 29.11% with a latency of 16.88 ms and energy consumption of 350.67 J, resulting in an efficiency of 0.058 at a resolution of 3840 × 2160.

For the Car class, MobileNet shows high accuracy (97.67%) with moderate latency (10.581 ms) and low energy consumption (23.04 J), leading to high efficiency (0.78103%/mW). ResNet50 and InceptionV4 also demonstrate high accuracies with slightly higher latencies and energy consumptions. In the Car class for Object Detection, YOLOX achieves an accuracy of 84.73% with a latency of 11.28 ms and energy consumption of 321.55 J, resulting in an efficiency of 0.2673%/mW at a resolution of 4096 × 2016. So, the Alveo's performance varies across different topologies and tasks, with MobileNet consistently demonstrating high accuracy, low latency, low energy consumption, and high efficiency across various classes.

Frame difference + classification generally yields higher accuracies and efficiencies than object detection. YOLOX, while performing well in object detection, tends to have higher latencies and lower efficiencies compared to other topologies.

TABLE III
EVALUATION OF THE AMD ALVEO U50'S PERFORMANCE

Class Name	Method	Topology	Acc(%)	Latency (ms)	Energy (Joule)	Efficiency (%/msW)	Video Resolution
Bird	Proposed Method	MobileNet	94.39	7.74	43.74	0.8935	3840 x 2160
		InceptionV4	85.02	12.82	111.12	0.220	
		ResNet50	77.68	11.55	95.46	0.2134	
Bird	Object Detection	YOLOX	69.11	16.37	347.75	0.1189	3840 x 2160
Trains	Proposed Method	MobileNet	93	8.44	30.91	0.989	4096 x 2016
		InceptionV4	54.52	12.23	120.24	0.1815	
		ResNet50	90.86	11.29	89.63	0.3129	
Train	Object Detection	YOLOX	51.52	13.15	318.89	0.135	4096 x 2016
Airplanes	Proposed Method	MobileNet	59.32	8.39	25.12	0.6	3840 x 2160
		InceptionV4	51.67	12.68	97.72	0.157	
		ResNet50	31.11	9.08	72.84	0.1269	
Airplane	Object Detection	YOLOX	29.11	16.88	350.67	0.0580	3840 x 2160
Car	Proposed Method	MobileNet	97.67	10.58	23.04	0.78103	4096 x 2016
		InceptionV4	91.12	13.20	89.612	0.2902	
		ResNet50	93	13.06	66.797	0.2855	
Car	Object Detection	YOLOX	84.73	11.28	321.55	0.2673	4096 x 2016

Overall, MobileNet appears to be the most balanced option for Alveo, providing high performance across multiple metrics. So, from all the experimental studies that we examined in the various classes, we conclude that using our proposed method, the MobileNet model has consistently high accuracy, low latency, and highly efficient energy for all the devices that we tested. Otherwise, YOLOX consistently demonstrates the lowest accuracy, the lowest latency, and the lowest efficiency. If we compare our proposed algorithm and end-to-end method, our method increases average accuracy by 28.314%, an average efficiency of 3.6 times, and an average latency reduction of 39.305% compared to the end-to-end method.

Frame differencing can be more efficient because it utilizes localized operations based on pixel changes, while end-to-end methods, such as YOLO process global features of the entire image, requiring more computation and thus needing higher energy consumption. MobileNet, ResNet50, ViT Base, and InceptionV4 each have strengths and shortcomings in fast object detection tasks using the frame difference method. MobileNet demonstrates high accuracy, low latency, and exceptional energy efficiency, making it ideal for IoT systems and edge devices, but its lightweight architecture may struggle with highly complex detection tasks. ResNet50 achieves a good balance of accuracy and efficiency but has slightly higher latency and energy consumption than MobileNet, limiting its suitability for resource-constrained environments. ViT Base excels in accuracy due to its transformer-based architecture, which captures global context effectively but suffers from high latency and significant energy demands and is less suited for real-time or energy-sensitive applications. InceptionV4 delivers high accuracy and robust detection but has the highest latency and energy consumption among the models, making it unsuitable for time-critical or energy-constrained tasks. YOLOX exhibits lower accuracy, higher latency, and less energy efficiency compared to methods utilizing the frame difference approach, as the end-to-end methodology employed by YOLOX struggles to handle the computational and efficiency demands of FMOD, which are better addressed by lightweight algorithms like the frame difference method.

Some of the reasons why our method cannot achieve maximum accuracy are Frame differencing is very sensitive to environmental noise. In dynamic scenes where the background is not static, such as areas with moving trees, flowing water, or fluctuating lighting conditions, even small changes in the background can be detected as motion. This leads to a large number of false positives. In addition, any vibration or small shift in the camera position is interpreted as a difference between frames, which further increases false detections.

Frame differencing has difficulty detecting motion in certain scenarios, such as small or slow objects. When objects move very slowly, the difference in pixel intensity between consecutive frames may be too subtle to pass the detection threshold, causing the method to miss these objects altogether. and very fast-moving objects until motion blur occurs. This blurring reduces the contrast and clarity of object edges, making it difficult for the frame differencing method to effectively identify and quantify changes.

V. CONCLUSION

This study presents an improved FMOD using the frame difference method. The proposed method will offer significant advantages in implementing fast-moving object identification in diverse sectors, such as surveillance, ADAS, and human activity recognition, with high computational speed and power efficiency. The implementation of our method used four object detection classes, such as birds, trains, airplanes, and cars, and some of the topologies were MobileNet, InceptionV4, ResNet50, and ViT Base. The tested data performance is accuracy (Acc%), latency (ms), energy consumption (Joule), and efficiency (%/msW). This proposed method has been implemented on the Alveo U50, Jetson Orin Nano, and Hailo-8 AI Accelerator. The result will compare our performance method to the YOLOX with the end-to-end method.

The experimental results indicate that our method consistently demonstrated high efficiency, accuracy, and latency for the three devices that were examined. On the other hand, the end-to-end method consistently demonstrated the lowest level of accuracy, the most significant latency, and the poorest

level of efficiency on devices in all parameters. Compared to the end-to-end method, the proposed method achieves an average latency efficiency gain of 3.6 times that of the YOLOX method. Additionally, it shows an average accuracy improvement of 28.3142%. Compared to the YOLOX, the average latency reduction is 39.305%. From the results, we conclude that the faster the movement of an object, the lower the accuracy will be obtained by the YOLOX method. Of all these classes, we know the fastest objects are trains and airplanes, and the percentage of accuracy in trains and airplanes is lower than in other categories.

End-to-end object detection methods, such as You Only Look Once (YOLO), consume more energy than simpler techniques like frame differencing due to their computational complexity and resource demands. YOLO performs tasks like feature extraction, classification, and bounding box regression in a single pass. In contrast, frame differencing is a straightforward method that detects motion by comparing pixel values between consecutive frames. It doesn't require complex computations or large-scale data processing, resulting in significantly lower energy consumption.

This method innovatively enhances motion detection and object classification in energy-constrained IoT applications. Unlike traditional frame difference techniques, which rely solely on pixel-based motion detection, this method leverages AI classification to improve detection accuracy and efficiency. This hybrid approach enables real-time object recognition with higher accuracy, reduced latency, and optimized energy consumption, making it different from conventional frame difference methods and more suitable for the dynamic requirements of FMOD. So, our proposed method is a lightweight detection algorithm that is well-suited for detecting fast-moving objects and has higher accuracy.

REFERENCES

[1] Z. Lin et al., "Pain without gain: Destructive beamforming from a malicious RIS perspective in IoT networks," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 7619–7629, Mar. 2024.

[2] Z. Lin, M. Lin, T. De Cola, J.-B. Wang, W.-P. Zhu, and J. Cheng, "Supporting IoT with rate-splitting multiple access in satellite and aerial-integrated networks," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11123–11134, Jul. 2021.

[3] C. W. Chen, "Internet of Video Things: Next-generation IoT with visual sensors," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6676–6685, Aug. 2020.

[4] J. Zhang, Q. Su, B. Tang, C. Wang, and Y. Li, "DPSNet: Multitask learning using geometry reasoning for scene depth and semantics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 6, pp. 2710–2721, Jun. 2023.

[5] P. Han, J. Du, J. Zhou, and S. Zhu, "An object detection method using wavelet optical flow and hybrid linear-nonlinear classifier," *Math. Problems Eng.*, vol. 2013, no. 1, pp. 1–14, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2013/965419

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, San Francisco, CA, USA, 1981, pp. 674–679.

[7] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, 1981.

[8] C.-W. Liang and C.-F. Juang, "Moving object classification using a combination of static appearance features and spatial and temporal entropy values of optical flows," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3453–3464, Dec. 2015.

[9] S. S. Sengar and S. Mukhopadhyay, "Detection of moving objects based on enhancement of optical flow," *Optik*, vol. 145, pp. 130–141, Sep. 2017.

[10] E. şimşek and B. Ozyer, "Selected three frame difference method for moving object detection," *Int. J. Intell. Syst. Appl. Eng.*, vol. 9, no. 2, pp. 48–54, 2021.

[11] M. Zhu and H. Wang, "Fast detection of moving object based on improved frame-difference method," in *Proc. 6th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, 2017, pp. 299–303.

[12] C.-M. Tsai and Z.-M. Yeh, "Intelligent moving objects detection via adaptive frame differencing method," in *Proc. 5th Asian Conf. Intell. Inf. Database Syst.*, Berlin, Germany, 2013, pp. 1–11.

[13] N. Singla, "Motion detection based on frame difference method," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 15, pp. 1559–1565, 2014.

[14] Y. Jusman, L. Hinggis, R. O. Wiyagi, N. A. M. Isa, and F. Mujaahid, "Comparison of background subtraction and frame differencing methods for indoor moving object detection," in *Proc. 1st Int. Conf. Inf. Technol., Adv. Mech. Elect. Eng. (ICITAMEE)*, 2020, pp. 214–219.

[15] J. Sun, W. Cao, Z. Xu, and J. Ponce, "Learning a convolutional neural network for non-uniform motion blur removal," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 769–777.

[16] A. HajiRassouliha, A. J. Taberner, M. P. Nash, and P. M. Nielsen, "Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms," *Signal Process., Image Commun.*, vol. 68, pp. 101–119, Oct. 2018.

[17] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, "Medical image processing on the GPU–past, present and future," *Med. Image Anal.*, vol. 17, no. 8, pp. 1073–1094, 2013.

[18] Y. Liu and J. Zhang, "Service function chain embedding meets machine learning: Deep reinforcement learning approach," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 3, pp. 3465–3481, Jun. 2024.

[19] J. Suo, X. Zhang, W. Shi, and W. Zhou, "E3-UAV: An edge-based energy-efficient object detection system for unmanned aerial vehicles," *IEEE Internet Things J.*, vol. 11, no. 3, pp. 4398–4413, Feb. 2024.

[20] S. Guo, C. Zhao, G. Wang, J. Yang, and S. Yang, "Ec^{2}detect: Real-time online video object detection in edge-cloud collaborative IoT," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20382–20392, Oct. 2022.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[22] R. Ojala, J. Vepsäläinen, J. Hanhirova, V. Hirvisalo, and K. Tammi, "Novel convolutional neural network-based roadside unit for accurate pedestrian localisation," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3756–3765, Sep. 2020.

[23] L. Zhang, J. Zheng, R. Sun, and Y. Tao, "GC-Net: Gridding and clustering for traffic object detection with roadside LiDAR," *IEEE Intell. Syst.*, vol. 36, no. 4, pp. 104–113, Jul./Aug. 2021.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–5.

[25] R. Zhao et al., "Accelerating binarized convolutional neural networks with software-programmable FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2017, pp. 15–24.

[26] W. Li and D. Song, "Automatic bird species detection from crowd sourced videos," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 348–358, Apr. 2014.

[27] S. Tian, X. Cao, B. Zhang, and Y. Ding, "Learning the state space based on flying pattern for bird detection," in *Proc. Integr. Commun., Navig. Surveillance Conf. (ICNS)*, 2017, pp. 5B3-1–5B3-9.

[28] D. S. Alex and A. Wahi, "BSFD: Background subtraction frame difference algorithm for moving object detection and extraction.," *J. Theor. Appl. Inf. Technol.*, vol. 60, no. 3, pp. 623–628, 2014.

[29] T. Biswas, D. Bhattacharya, D. Rudrapal, S. Roy, and G. Mandal, "Motion detection in real-time surveillance using two frame differencing," in *Proc. Int. Conf. Inf. Commun. Technol. Competit. Strategies*, 2023, pp. 97–109.

[30] S. S. Sengar and S. Mukhopadhyay, "Moving object detection based on frame difference and w4," *Signal, Image Video Process.*, vol. 11, pp. 1357–1364, Oct. 2017.

[31] N. Alsharabi, "Real-time object detection overview: Advancements, challenges, and applications," *Amran Univ. J.*, vol. 3, p. 12, Nov. 2023.

[32] X. Chen and J. Guhl, "Industrial robot control with object recognition based on deep learning," *Procedia CIRP*, vol. 76, pp. 149–154, Jan. 2018.

[33] S. Zhu, K. Ota, and M. Dong, "Energy-efficient artificial intelligence of things with intelligent edge," *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7525–7532, May 2022.

[34] S. Nayak, R. Patgiri, L. Waikhom, and A. Ahmed, "A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications," *Digit. Commun. Netw.*, vol. 10, no. 3, pp. 783–804, 2022.
[35] M. K. Hasan et al., "Federated learning for computational offloading and resource management of vehicular edge computing in 6G-V2X network," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 3827–3847, Feb. 2024.
[36] C. Meng, M. Sun, J. Yang, M. Qiu, and Y. Gu, "Training deeper models by GPU memory optimization on tensorflow," in *Proc. ML Syst. Workshop NIPS*, 2017, pp. 1–8.
[37] T. Luo et al., "DaDianNao: A neural network supercomputer," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 73–88, Jan. 2017.
[38] "Hailo AI: The world's best edge AI processors." Hailo. Accessed: Jul. 1, 2024. [Online]. Available: https://hailo.ai/products/ai-accelerators/hailo-8-ai-accelerator/
[39] Y.-J. Lin and T. S. Chang, "Data and hardware efficient design for convolutional neural network," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 5, pp. 1642–1651, May 2018.
[40] S. Cao et al., "Efficient and effective sparse LSTM on FPGA with bank-balanced sparsity," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2019, pp. 63–72.
[41] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "DLAU: A scalable deep learning accelerator unit on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 3, pp. 513–517, Mar. 2017.
[42] C. Farabet et al., "Large-scale FPGA-based convolutional networks," *Scaling Up Mach. Learn., Parallel Distributed Approaches*, vol. 13, no. 3, pp. 399–419, 2011.
[43] K. Guo et al., "Angel-eye: A complete design flow for mapping CNN onto embedded FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 1, pp. 35–47, Jan. 2018.
[44] S. Bouguezzi, H. B. Fredj, T. Belabed, C. Valderrama, H. Faiedh, and C. Souani, "An efficient FPGA-based convolutional neural network for classification: Ad-mobilenet," *Electronics*, vol. 10, no. 18, p. 2272, 2021.
[45] S. Moini, B. Alizadeh, M. Emad, and R. Ebrahimpour, "A resource-limited hardware accelerator for convolutional neural networks in embedded vision applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 10, pp. 1217–1221, Oct. 2017.
[46] D. Youvan, "Developing and deploying AI applications on Nvidia Jetson Orin NX: A comprehensive guide," Jun. 2024, submitted for publication.
[47] H. V. Pham, T. G. Tran, C. D. Le, A. D. Le, and H. B. Vo, "Benchmarking Jetson edge devices with an end-to-end video-based anomaly detection system," in *Proc. Future Inf. Commun. Conf.*, 2024, pp. 358–374.
[48] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, Dec. 2015.
[49] S. Phiphiphatphaisit and O. Surinta, "Food image classification with improved mobilenet architecture and data augmentation," in *Proc. 3rd Int. Conf. Inf. Sci. Syst.*, 2020, pp. 51–56.
[50] N. A. Al-Humaidan and M. Prince, "A classification of Arab ethnicity based on face image using deep learning approach," *IEEE Access*, vol. 9, pp. 50755–50766, 2021.
[51] K. Shankar, Y. Zhang, Y. Liu, L. Wu, and C.-H. Chen, "Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification," *IEEE Access*, vol. 8, pp. 118164–118173, 2020.
[52] D. Alexey, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020, *arXiv: 2010.11929*.
[53] J. Zhang and S. Ke, "Improved YOLOX fire scenario detection method," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–8, Mar. 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/epdf/10.1155/2022/9666265
[54] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. 13th Eur. Conf., Comput. Vis. (ECCV)*, Zurich, Switzerland, 2014, pp. 740–755.
[55] J. Pan et al., "EdgeViTs: Competing light-weight CNNs on mobile devices with vision transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 294–311.

Mas Nurul Achmadiah received the Bachelor of Applied Science degree in mechatronics engineering from the Electronic Engineering Polytechnic Institute of Surabaya, Surabaya, Indonesia, and the master's degree in electronic engineering from the Tenth of November Institute of Technology, Surabaya, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree with the Electro-Optics Department, National Formosa University, Huwei, Taiwan.

She is a Lecturer with the Department of Electrical Engineering, State Polytechnic of Malang, Malang, Indonesia. Her research interests are in artificial intelligence, edge computing devices, intelligent control, and image processing.

Afaroj Ahamad received the B.Tech. degree in electronics and communication engineering from Aligarh Muslim University, Aligarh, India, in 2012, the M.Tech. degree in electronics engineering from Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, India, in 2019, and the Ph.D. degree in deep learning algorithms for edge AI from National Formosa University, Huwei, Taiwan, in 2023.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Yuan Ze University, Taoyuan, Taiwan. Previously, he served as an Assistant Professor with the Department of Computer Science and Information Engineering, Asia University, Taichung. from October 2024 to January 2025. From January 2023 to September 2024, he worked as a Senior FPGA Application Engineer with E-Elements Technology Company Ltd., Taipei, Taiwan, where he contributed to embedded system design and AI acceleration using DPU and FPGA technologies. He has extensive experience in both academic and industrial settings, with a focus on deploying efficient AI solutions for IoT and robotics applications. His research interests include deep learning, machine learning, FPGA-based AI acceleration, and IoT systems for edge computing.

Chi-Chia Sun (Member, IEEE) received the B.S. degree in computer science and engineering from National Taiwan Ocean University, Keelung, Taiwan, in 2004, the M.S. degree in electronic engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2006, and the Doktor Ingenieur degree from Dortmund University of Technology, Dortmund, Germany, in 2011.

From April 2008 to March 2011, he worked as a Research Assistant with Dortmund University of Technology. He is currently a Full Professor with the Department of Electrical Engineering, National Taipei University, Taipei. His research interests include image processing, system integration, and very large-scale integration/FPGA design.

Wen-Kai Kuo (Member, IEEE) received the Ph.D. degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000.

He has been a Professor with the Department of Electro-optics Engineering, National Formosa University, Huwei, Taiwan. His research interests are optical sensors and systems.

Prof. Kuo is a member of the Phi-Tau-Phi Honorary Scholar Society.