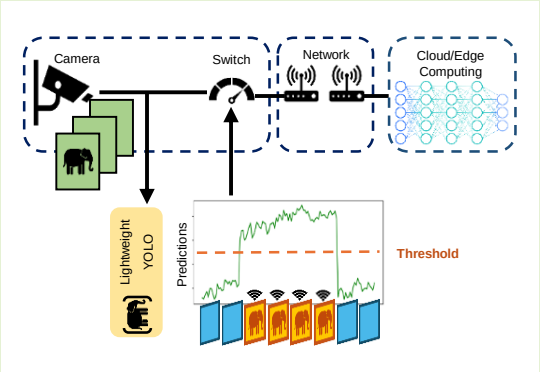
Intelligent Sensing Framework: Near-Sensor Machine Learning for Efficient Data Transmission

Wenjun Huang, Arghavan Rezvani, Hanning Chen, Yang Ni, Sanggeon Yun, Sungheon Jeong, Guangyi Zhang, Mohsen Imani

***Abstract*— Applications in the Internet of Things (IoT) utilize machine learning to analyze sensor-generated data. However, a major challenge lies in the lack of targeted intelligence in current sensing systems, leading to vast data generation and increased computational and communication costs. To address this challenge, we propose a novel sensing framework to equip sensing systems with intelligent data transmission capabilities by integrating a highly efficient machine learning model placed near the sensor. This model provides prompt feedback for the sensing system to transmit only valuable data while discarding irrelevant information by regulating the frequency of data transmission. The near-sensor model is quantized and optimized for real-time sensor control. To enhance the framework's performance, the training process is customized and a "lazy" sensor deactivation strategy utilizing temporal information is introduced. The suggested framework is orthogonal to other IoT frameworks and can be considered as a plug-in for selective data transmission. The framework is implemented, encompassing both software and hardware components. The experiments demonstrate that the framework utilizing the suggested module achieves over 85% system efficiency in terms of energy consumption and storage, with negligible impact on performance. This framework has the potential to significantly reduce data output from sensors, benefiting a wide range of IoT applications.**



***Index Terms*— Energy Efficiency, Internet of Things, Near-Sensor Computing, Intelligent Sensing, Machine Learning.**

I. Introduction

THE prevalence of ubiquitous sensors is currently experiencing an exponential surge, both in terms of their quantity and the vast amount of data they generate. Despite the rapid growth, existing approaches to sensor data processing and transmission cannot keep pace due to their algorithmic and architectural limitations [1]. In numerous IoT applications, data collected by sensors are analyzed using machine learning (ML) models [2]–[5]. As the volume of data continues to grow, many applications opt to send the data to more computationally powerful nodes, such as edge or cloud computing nodes, to execute the learning algorithms. In either scenario, a large volume of data is transmitted at a high rate to ensure that all necessary information is captured and processed for various tasks. The significant amount of data conveyed in both scenarios places high demands on energy and storage resources, resulting in considerable resource pressure and wastage [6]. This is especially problematic for applications that require a relatively complex and expensive ML model. Figure 1 depicts a typical IoT system for video monitoring systems, where dense data generated by the camera is continuously analyzed using complex ML models. In the system, visual signals captured by surveillance cameras are transmitted continuously to a costly ML model, which may be hosted on a central server, such as a cloud or edge computing node. Depending on the intended purposes, the ML model performs various tasks, including but not limited to classification, object detection, and segmentation [7], [8].

Many studies attempted to alleviate the energy and storage pressures in IoT applications from multiple perspectives, e.g., computing offloading, resource allocation, etc. Traditional methods have shown substantial progress in tackling these issues. Certain research efforts leveraged the Lyapunov optimization algorithm [9] to identify the optimal decision [10].

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation (NSF) under Grants #2127780, #2319198, #2321840, #2312517, and #2235472, the Semiconductor Research Corporation (SRC), the Office of Naval Research through the Young Investigator Program Award, and Grants #N00014-21-1-2225 and N00014-22-1-2067. Additionally, support was provided by the Air Force Office of Scientific Research under Award #FA9550-22-1-0253, along with generous gifts from Xilinx and Cisco.

All the authors are with the Department of Computer Science at the University of California, Irvine, CA 92697. Mohen Imani is the corresponding author (email: m.imani@uci.edu). Wenjun Huang and Arghavan Rezvani contributed equally to this work.

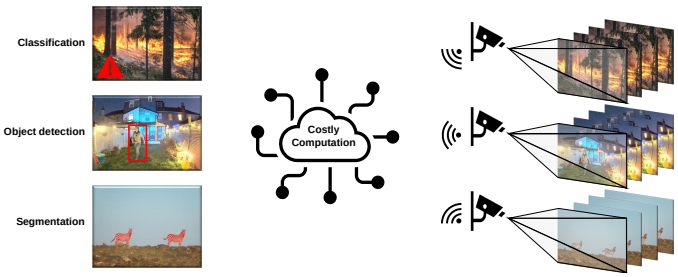


Fig. 1. Application scenarios of an intelligent system.

Others framed resource allocation and computing offloading as optimization challenges [11]–[15]. However, these approaches exhibit certain limitations. Firstly, they require knowledge of the underlying model, which proves challenging due to the intricate and dynamic nature of IoT systems. Secondly, they are vulnerable to stuck at local optima. Some research [16]–[20] have introduced intelligent offloading strategies grounded in deep learning (DL). Furthermore, some research have placed emphasis on the optimization of hardware structures, thereby enhancing the efficiency of edge computing [21]–[23].

Different from the work above, which uses ML/DL algorithms to automate offloading and resource allocation, some research proposes solutions to reduce data generated by the sensor. For example, in the realm of computer vision, *analyze-then-compress* (ATC) approaches present an alternative strategy in which front-end devices extract and transmit features to a central server. Depending on the specific scenario in which it is being applied, ATC approaches utilize a variety of traditional feature extraction algorithms, ranging from hand-crafted methods (e.g., [24]–[26]) to information theory-based methods [27], [28]. In recent years, more advanced deep-learning-based methods have garnered significant attention. Several early layers of DNN are deployed on the front-end devices for extracting highly compact and representative features. In the face recognition task, for example, the face of an individual can be represented by features with several hundred dimensions [29]–[31]. By representing data in such features, the amount of data that needs to be transmitted can be significantly decreased. Additionally, only a few lightweight operations are required to be performed on the central server.

However, a notable limitation of DNN-based ATC methods is their restricted capacity for generalization. Given the meticulous design of DNN architectures, the features they extract and transmit to the central server are often highly abstract and tailored specifically to the intended task. However, visual signal carrying pertinent information typically undergoes a sequence of downstream tasks for comprehensive analysis. Consequently, the inherent challenge arises from the deficiency in generalization, rendering it difficult to design a backbone network capable of extracting features suitable for all such tasks. Moreover, in numerous scenarios, it becomes useful to retain visual signals for subsequent analysis or future reference. The transmission of excessively abstract features significantly complicates the process of reconstructing the original visual signal on the server side. Although front-end devices possess the capability to store the original signals, their constrained storage capacity poses a challenge.

In addition, all the efforts mentioned above, whether from an IoT or ML perspective, still need to process all the data generated from the sensor, neglecting the fact that in many IoT applications (e.g., fire alarm, wildlife monitoring, crime surveillance [32], healthcare [33]), only a small fraction of sensor activity typically contains valuable information. Hence, it is unnecessary to run a costly service, such as a large-scale DNN model, that handles a continuous and complete stream of sensor data, whether on the edge or in the cloud. This is because the service specifically targets only that small fraction of valuable data, yet it still requires processing substantial amounts of irrelevant information.

Spiking Neural Networks (SNNs) and event cameras, on the other hand, generate data only when there is a change in the scene, reducing the amount of data needed for transmission. However, in a static scene, an event camera would barely generate any data, effectively rendering it blind to stationary information. This limitation restricts its applicability, particularly for tasks involving slowly moving objects. Moreover, the spatial resolution of event cameras is generally lower compared to high-resolution frame-based cameras, which can be a limiting factor for applications that require detailed spatial information. Event cameras can also be sensitive to noise, especially in low-light conditions, resulting in spurious events that add complexity to the data processing. Last but not least, the price of event cameras is generally higher than that of traditional RGB cameras, which can limit their applicability for widespread deployment.

Observing the limitations of the approaches previously discussed, in this paper, we rethink and redesign the sensing system, proposing a new framework that is orthogonal to previous research directions. Rather than reducing the data representation or determining where and how data should be relocated for service execution, our framework focuses on reducing the amount of data sent out from the sensor side by identifying valuable information. Our framework, acting as a "filter", can be applied before any aforementioned approaches, and easily be integrated into any system as a plug-in.

Our proposed framework consists of a few components. First, we deploy a lightweight model near the sensor to detect whether a frame contains useful information, which we refer to as a frame of interest (FOI), and only send out those FOIs. The model helps mitigate the huge amount of unnecessary analysis of costly ML models over the central server. Although this process can also be deployed before the costly ML models at the same place, our near-sensor model offers substantial savings in transmission costs, encompassing energy, bandwidth, and more. To enable intelligent sensing, the near-sensor model should be fast enough to process frames in real-time and provide feedback. With the help of this feedback, our framework produces selective and sparse data. Furthermore, we enhance the overall performance of the framework by introducing several effective schemes to mitigate potential misdetections of the lightweight model, which we explore in the following sections.

In this work, we describe the following contributions:

- We propose a new framework that improves IoT system

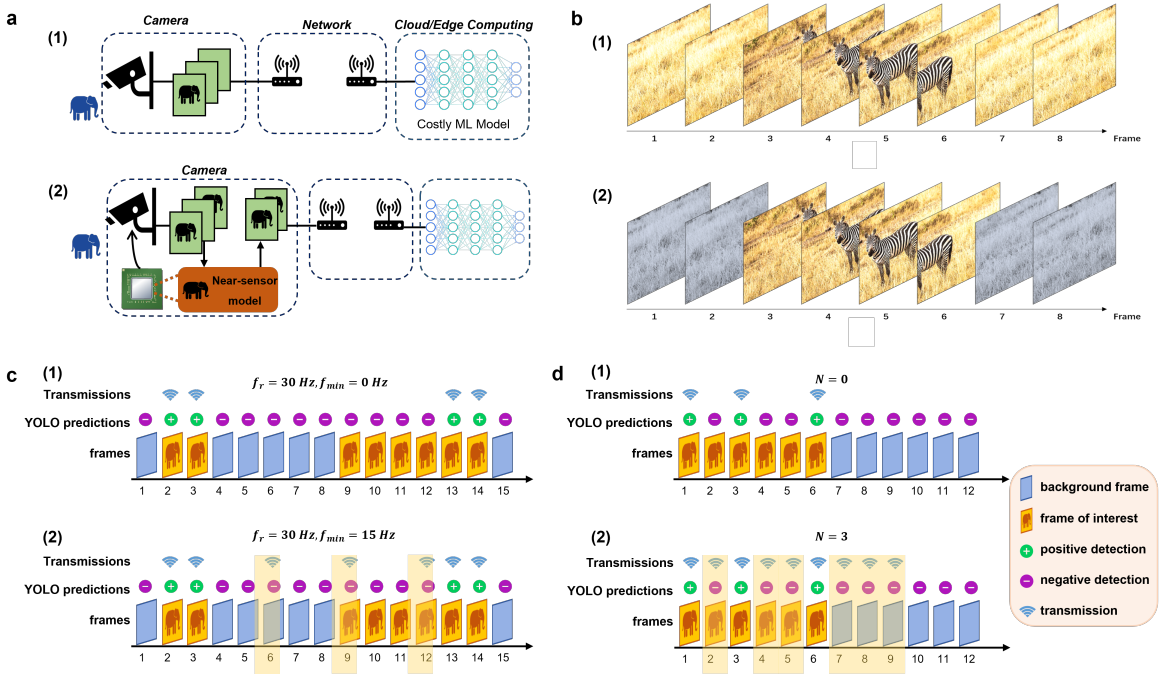


Fig. 2. Motivation and design of our proposed intelligent sensing module. **a.** General system framework of conventional systems and our system. **b.** Visualization of the data transmission in our system. **c.** Illustration of minimum data transmission frequency (denoted by f_{min}) in our system. f_r denotes the camera's refresh rate. **d.** Illustration of lazy sensor deactivation scheme in our system, N is the number for deactivation count.

energy and storage efficiency orthogonal to the previous approaches. It can be readily inserted into any existing system, serving as an intelligent data generation "filter". We call the sensor exploiting this framework an "intelligent sensor" in the rest of the paper.

- To illustrate our framework, we design a modified DNN model tailored to near-sensor computing.
- We introduce schemes for alleviating possible misdetections of the near-sensor model, including non-zero minimum transmission frequency and lazy deactivation. We also conduct a thorough investigation into their impact on overall system performance.
- We implement the framework encompassing both software and hardware components. Our experiments demonstrate that utilizing our intelligent framework leads to a substantial reduction in energy and storage consumption in sensing systems.

II. Methods

A. Framework Overview

Figure 2**a** illustrates the framework of a conventional system and our framework. In Figure 2**a**(1), the conventional sensor captures and transmits all the frames to the costly models, regardless of the presence of useful information in the frames. On the contrary, the intelligent sensor equipped with our framework, as shown in Figure 2**a**(2), utilizes a lightweight model near the sensor to detect and control the FOI transmission. The model is deployed on an edge computing device integrated into the camera, connecting to the image sensor. Specifically, the camera captures a continuous stream of frames, which are then fed to the lightweight model for real-time predictions. With the presence of FOI, the camera raises the data transmission frequency, and the frames are transmitted to the central server for more sophisticated operations; if the frame is detected as background (with no interest), the camera will turn off the data transmission. Figure 2**b** provides a visualized example, where the transmitted frames are presented in color while the discarded frames are shaded in gray. The system adopting our framework, as demonstrated in Figure 2**b**(2), outperforms conventional systems depicted in Figure 2**b**(1) by exclusively transmitting frames containing a zebra, resulting in a reduction of storage and energy consumption by half in this particular instance.

This is because transmitting only the necessary FOIs to the central server reduces the number of inferences needed by the complex ML model on the central server, which is the primary source of energy consumption. This reduction is achieved while introducing only a negligible energy overhead associated with the near-sensor model. This is in contrast to previous approaches that would transmit all frames to the server based on the camera's refresh rate, resulting in significant energy waste due to performing inference on numerous unnecessary frames.

In this work, we concentrate on the effect of our proposed framework on energy consumption reduction. Each element in the framework is elaborated on in the following sections.

B. Near-sensor Model

The near-sensor model is tasked with distinguishing FOIs from all other frames. One way to tackle this problem is by using a classifier. However, the frames captured by a sensor may contain multiple objects of interest with varying scales and positions, while classifiers are typically trained on images that contain a single, centered object (such as

those found in CIFAR-10, CIFAR-100 [34], and ImageNet [35]). These classifiers have limitations in detecting multiple objects with varying scales and positions. As a result, a deep object detection model is often employed instead. Among different object detection models, YOLO [36], a single-stage detector, is selected. Compared with two-shot detectors (e.g., R-CNN, Fast R-CNN, Faster R-CNN, R-FCN [37]–[40]), YOLO is lightweight, faster, and with comparable accuracy in a suitable scenario [41], [42]. These features make YOLO a good candidate for being embedded into the sensor.

The output layer of YOLO contains bounding box predictions concatenated to the class prediction and objectness confidence. However, the goal of our intelligent sensor is to detect the existence of objects of interest, regardless of their position in the frame. Therefore, we can only keep the objectness confidence in the output, which can be used further to determine FOI. We set a threshold for the objectness confidence, and only the frames with confidence exceeding this threshold are transmitted. As increasing the threshold, the detection becomes stricter, resulting in fewer frames being considered FOI. Our framework's definition allows us to customize the YOLO model in the following ways:

1) Model Optimization: The architectures of YOLO series contain several repetitive blocks. Although these blocks contribute to the model capacity, they make the model power-hungry and slower during inference. For example, YOLOv5 model family has five variations: x-large, large, medium, small, and nano. While each model shares the same structure, they differ in the network's depth and the number of filters in different layers (width). Since our model does not predict bounding boxes, we can modify its depth and width to create a more lightweight model that still achieves comparable performance on our task. In other words, in contrast to the YOLO model, which predicts both the class and location of an object, our proposed near-sensor model requires only class prediction. This simplification allows us to reduce the number of model parameters without compromising the accuracy of the class prediction task.

In our experiments, we utilized three YOLO-based models, namely YOLOv5n, YOLOv5nm, and YOLOv5ns. YOLOv5n stands for Yolov5 nano, which is the smallest introduced YOLOv5 model. By modifying the depth and width of the YOLOv5n, we achieved more lightweight models, which we called nano-medium (YOLOv5nm) and nano-small (YOLOv5ns). In YOLOv5nm, the depth and width are half of the depth and width of the YOLOv5n, and in YOLOv5ns, this ratio is one-third for depth and one-fourth for width.

As mentioned earlier, the output of YOLO model not only contains the confidence but also concatenates the bounding box information, which is not required in our proposed module. Consequently, we can remove the part of the model associated with the bounding box during the inference to reduce the model size.

2) Inference Simplification: YOLO utilizes non-max suppression (NMS) algorithm as the final step to pick the most appropriate bounding box for the object among all of the predicted boxes for that specific object. NMS algorithm starts with selecting the box with the highest objectness score among all, removing all the boxes with high overlap with the selected box, and repeating these steps iteratively. However, since the sensing scenario does not require bounding boxes, we can simplify this step. Instead of running the NMS algorithm, we only keep the highest objectness confidence. If there is one confidence value greater than the threshold, it indicates the presence of at least one object in the prediction. Therefore, by solely comparing the highest confidence value with the threshold, we can achieve comparable performance , resulting in reduced inference time.

3) Model Quantization & Loss Function Customization: Model quantization is another well-known approach to accelerating model inference. It involves using fewer bits to store model parameters while maintaining nearly the same level of accuracy [43]–[46]. Aggressive quantization leads to a highly lightweight model, but at the cost of reduced accuracy compared to the original model. On the other hand, less aggressive quantized models experience minimal accuracy loss, but they are not as lightweight as aggressively quantized models [47]. The amount of tolerable accuracy loss varies across different tasks. For this work, we utilized the **kmeans-lut** quantization which is a Look-up-table (LUT) based quantization [48], where LUT is generated by K-Means clustering.

Moreover, refining the loss function can enhance the performance of the model when subjected to intensive quantization. The conventional YOLOv5 has three loss terms:

$$L = l_{obj} + l_{cls} + l_{bbox} \tag{1}$$

where l_{obj}, l_{cls}, and l_{bbox} are objectness confidence loss, classification loss, and bounding box loss, respectively. Among the loss terms, reducing the l_{obj} and l_{cls} loss terms contributes to accurate object detection and classification, resulting in improved performance of our model. In contrast, the l_{bbox} loss term, which corresponds to the precise bounding box position, has a negative impact on our model. This is because it forces the model to make a compromise during the gradient descent search, making it more difficult for the model to converge to the optimal. By removing the l_{bbox} term, our near-sensor model can prioritize the detection of FOIs without considering the bounding box generation, enabling the model to achieve a higher degree of quantization while maintaining a comparable level of accuracy. Therefore, the following loss function was adapted for training the near-sensor model with a faster convergence to improve accuracy in our task:

$$L = l_{obj} + l_{cls} \tag{2}$$

C. Data Transmission Frequency

The prediction of the near-sensor model regulates the frequency of data transmission, thereby reducing the volume of data transmitted to the central server. If the camera records FOIs, it should be configured to transmit all FOIs to the server, with a frequency equivalent to the camera's refresh rate. Conversely, when the camera captures background frames, it should lower the data transmission rate to save energy. This reduced frequency is referred to as the *minimum transmission frequency*. The minimum transmission frequency can vary between zero and the camera's refresh rate. If the minimum

transmission frequency is set to match the camera's refresh rate, all frames captured by the camera are forwarded to the server, indicating that the transmission is unaffected by the predictions of the lightweight model. In this scenario, the volume of data transmitted to the server is identical to that of conventional systems. Conversely, when the minimum transmission frequency is set to zero, any frames identified as background frames would not be transmitted to the server. Figure 2**c**(1) demonstrates the data transmission of our framework. The blue frames represent the background and the yellow frames with an elephant depict FOIs. A positive or negative sign is used to present the near-sensor model predictions; The frames that are marked with a positive sign represent the prediction of FOIs. The frames being transmitted to the server are indicated by the Wi-Fi icon. Only the frames that are recognized as FOI (frames 2, 3, 13, and 14), are transmitted.

However, even though the lightweight model displays a high level of accuracy, it is still inevitable to misdetect some FOIs as background frames, and these misdetected frames are all discarded when the minimum transmission frequency is zero since the data transmission is completely halted. This wrong discard can be alleviated by increasing the minimum transmission frequency, which means that even if the frames are detected as background frames, they are still transmitted to the server regularly at a lower non-zero frequency. An example demonstrating the effect of increasing the minimum transmission frequency is shown in Figure 2**c**(2). When the minimum transmission frequency equals zero (Figure 2**c**(1)), all the frames detected as background are discarded by the intelligent sensor. This significantly reduces the amount of data transmitted while also losing some useful information (e.g., frames 9 - 12). To reduce the number of missing FOIs, we increased the transmission frequency in Figure 2**c**(2). In the figure, the camera's refresh rate $f_r = 30$ Hz, and the minimum transmission frequency f_{min} is set to $f_r/2$ (i.e., 15 Hz). Under this setting, even if the transmission frequency is tuned down, the sensor would also send one frame every two frames. From the figure, we can observe that although the prediction of the lightweight model maintains the same, we transmit more FOIs to the server (frame 9 and frame 12).

D. Lazy Sensor Deactivation

Since FOIs contain valuable information, in this work, the priority is given to transmitting all FOIs rather than mistakenly transmitting a background frame. Therefore, we define misdetections as the FOIs which are not transmitted. Considering the fact that the frames in a video have temporal correlation, we assume that if the camera captures an FOI, the following frame is likely to be an FOI as well. Thus, in order to reduce the misdetection of FOIs, inspired by [49], we proposed a scheme for lazy sensor deactivation, which considers the detection results of neighboring frames. However, unlike the work in [49] which schedules observation points over the target execution, our scheme entails monitoring the number of consecutive background frames detected by the near-sensor model. The camera maintains a high transmission frequency until the count (C_1) of consecutive background detection reaches a pre-defined number (N). Once the number is met, the camera tunes down the transmission frequency and resets the count. The count is reset to zero whenever an FOI is identified. The adoption of our lazy sensor deactivation scheme enables the detector to rectify the misdetection of a single frame by utilizing the adjacent frame's information. In comparison to the detector without the lazy sensor deactivation scheme, utilizing our approach preserves more FOIs since an occasional misdetection cannot affect the transmission frequency. Decisions for tuning the transmission frequency are made based on a few adjacent frames. Figure 2**d** provides an example that demonstrates the advantages of implementing lazy sensor deactivation. In this example, the value of N is set to 3. When compared to the system that does not utilize lazy deactivation (shown in Figure 2**d**(1)), the implementation of lazy deactivation (Figure 2**d**(2)) also transmits the FOIs that are misdetected by the near-sensor model, as demonstrated by the transmission of frames 2, 4, and 5.

The utilization of the lazy sensor deactivation scheme incurs two costs from a storage perspective. The first cost arises when a negative sample is mistakenly identified as an FOI, leading to the reset and restart of the count. In the worst-case scenario, a single negative sample misdetection results in storing $2N$ additional frames. Nonetheless, this cost is acceptable as our primary concern lies in preserving the completeness of FOIs. The inclusion of a few extra negative samples following FOIs does not influence the pertinent information we aim to retain. Moreover, given that N is not an excessively large value, our storage capacity can handle these rare occurrences.

The second cost inherent in the lazy sensor deactivation scheme manifests in the recovery of the transmission frequency to a high level and the subsequent repetition of counting following each period of frequency decrease. Given that a large proportion of frames comprise background and such frames often appear in the form of segments, the frames following a low transmission frequency period are more likely to be background as well. As a result, in a long sequence of background frames, the detector stores N more frames after each low transmission frequency period. To mitigate the redundancy following each period, we introduced one more count (C_2) to monitor the number of consecutive low transmission frequency periods. This count is used to calculate N_{new} for consecutive background frames:

$$N_{new} = \max(1, \frac{N}{2^{C_2}}) \tag{3}$$

Upon tuning down the transmission frequency, C_2 increments by 1. However, the detection of an FOI interrupts the consecutive low transmission frequency periods, resetting the count C_2 to zero. At the start of each period, equation (3) determines the number (N_{new}) for that particular period. Using the count C_2 for consecutive low transmission frequency periods gradually decreases the threshold from N to 1 in the long run, leading to greater storage and energy savings than the vanilla scheme.

Algorithm 1 outlines the pseudocode of the proposed scheme, which incorporates minimum transmission frequency and lazy sensor deactivation. The pseudocode 6 - 12 indicate

the code for lazy sensor deactivation and 14 - 20 indicate the code for minimum transmission frequency, where f_r is the camera's refresh rate, f_{min} is the minimum transmission frequency, and C_3 is a count used to determine whether a frame should be transmitted in a minimum transmission frequency period.

E. Dual-camera Collaboration

Recording and analyzing valuable information necessitate the utilization of high-resolution images, thereby engendering a predilection for high-resolution cameras. Nevertheless, the sustained operation of such cameras for near-sensor computing proves to be energy burdensome, given their elevated power consumption. Our objective is to furnish dependable performance while concurrently minimizing energy consumption. Therefore, we integrated an additional low-resolution, and power-efficient, camera into the sensor configuration.

During periods devoid of FOIs, the high-resolution camera remains inactive to conserve power, while the low-resolution camera is engaged in executing the near-sensor model, as elucidated in Section II-B. When an FOI is detected by the near-sensor model utilizing the low-resolution camera, the high-resolution camera is activated, capturing and subsequently transmitting the pertinent frames. During this phase, the low-resolution camera is deactivated, given the superior quality of the frames obtained by the high-resolution camera.

The power consumption breakdown of using dual-camera collaboration on the sensor side is depicted in Figure 3. The incorporation of a power-efficient low-resolution camera results in significant power savings (highlighted in shadow), even compared to using our module with a single high-resolution camera. Given the infrequent occurrence of FOIs, our dual-camera collaboration scheme proves to be highly effective in mitigating energy consumption at the sensor side over an extended duration.

Note that our dual-camera collaboration requires an additional low-resolution camera, which increases the cost per device. The price of an image sensor depends on several factors, such as resolution, sensor size, technology, dynamic range, and noise suppression. While a basic CMOS sensor might cost a few dollars, high-end specialized sensors can be significantly more expensive, potentially reaching thousands of dollars. In our system, a basic sensor is sufficient for FoI detection, resulting in an increase of only 1/1000 to 1/100 over the original expenses. On the other hand, these basic sensors support at least 30 frames per second (fps), which is sufficient for our task and does not impact efficiency.

III. Experiments

A. Experimental Setup

In this work, we trained and evaluated our framework in the context of animal detection using the Microsoft Common Objects in Context (MS COCO) dataset [50], which is widely used for object detection tasks. In this context, the images in the dataset were selected and relabeled. The images containing at least one object belonging to the animal category are considered FOI and are labeled 1. The remaining

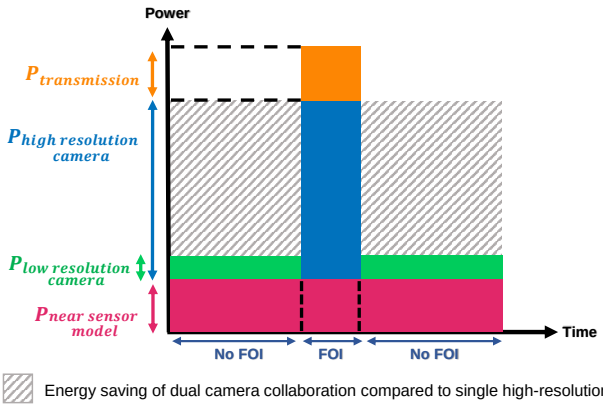


Fig. 3. Energy consumption breakdown on the sensor for the framework utilizing dual-camera collaboration.

frames are considered background and labeled as 0. The near-sensor lightweight model detects and transmits FOIs while filtering out the background frames. The detected frames are transmitted to a more sophisticated model, in our case a well-trained Fast R-CNN model, to perform advanced operations. The framework is implemented using PyTorch [51]. In accordance with the scenario, we ordered the data in the testset with a specific logic: FOIs and background frames are presented in a fragmented manner, appearing consecutively and alternating with each other. The frames in fragments are ordered randomly.

B. Parameter Evaluation Metrics

The goal of the framework is to detect the animals in all FOIs while minimizing the system's energy consumption and occupying minimal storage. In essence, our module aims to minimize the miss detection rate, defined as the fraction of FOIs that are not detected by our near-sensor model:

$$P_{miss} = \frac{n_{miss}}{n_{FOI}}, \tag{4}$$

where n_{miss} is the number of missed FOIs by the near-sensor model, and n_{FOI} is the total number of FOIs in the stream.

In addition, we prioritized the percentage of transmission reduction achieved by our module in comparison to sending all frames captured by the camera, which is defined as:

$$P_{trans} = \frac{n_{trans}}{n_{frames}}, \tag{5}$$

where n_{trans} is the number of transmitted frames, and n_{frames} is the total number of frames in the testset. The energy consumption of the system, including transmission and inference energy of the Fast R-CNN model, is closely related to the number of transmitted FOIs; thus the percentage of transmission reduction serves as a key indicator of the effectiveness of our framework. In addition to impacting energy consumption, P_{trans} also reflects the amount of storage that can be saved on the server.

It is worth emphasizing that our framework involves trade-offs among its various parameters. Altering the values of these parameters can lead to different performances with respect to missed detection frames and the percentage of transmission

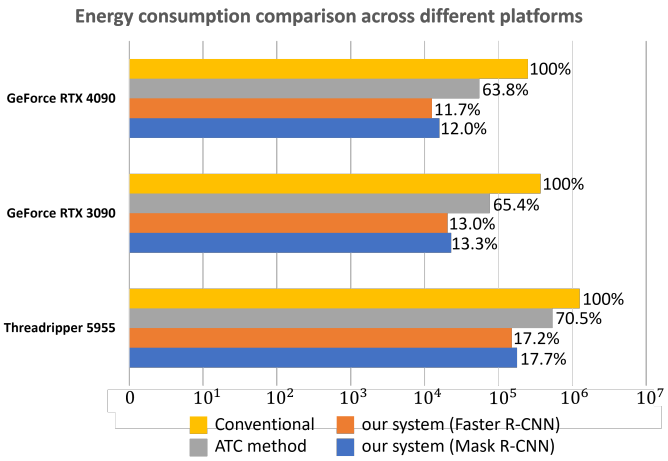


Fig. 4. Energy consumption of the baselines and the system adopting our framework. The experiments are conducted with $M = 20$ (the total number of the frames $n_{total} = 21336$, $P_{miss} = 3\% \pm 0.6\%$). The conventional system implements Fast R-CNN on the server.

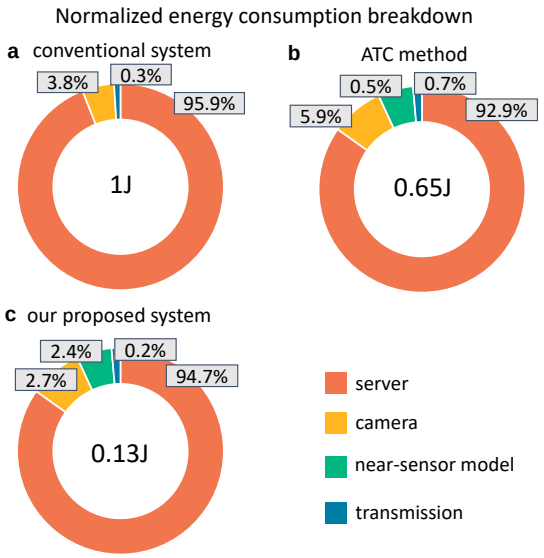


Fig. 5. Normalized energy consumption breakdown. **a.** Conventional system. **b.** ATC method. **c.** The system with our framework.

reduction. For instance, the most extreme scenario is to maintain the transmission frequency equal to the camera's refresh rate and transmit all frames to the server. Although this setting would result in zero missed detection frames, it would also lead to the highest possible energy consumption. In the following, we analyze the influence of each parameter on the performance of our framework utilizing the proposed module and discuss the trade-offs between these parameters.

This study explores the impact of four key parameters on our system's performance:

- the confidence threshold (T) of YOLO
- the ratio (M) of the number of background frames to the number of FOIs
- minimum transmission frequency (f_{min})
- the count (N) at which the sensor deactivates

C. Results

The comparison of the conventional system, the ATC method, and the system with our framework are shown in Figure 4 (The x axis is illustrated in log scale). For the system with our framework, we implemented two models (i.e., Mask R-CNN [52], Faster R-CNN) on the server, sorted in descending order of model complexity. For the conventional system, we used the least complex model, i.e., Fast R-CNN. For the ATC method, we implemented Mask R-CNN by adopting the approach described in [53]. This is because, in ATC methods, all downstream tasks rely on the same abstract features, requiring the complex server-side model to perform inference for all tasks. The energy consumptions are measured on three platforms (GeForce RTX 4090, GeForce RTX 3090, and AMD Threadripper 5955).

The energy consumption of conventional systems comprises three components: energy consumption by the sensor, transmission energy, and inference energy consumed at the server. Both the ATC method and the system with our framework introduce an additional component: near-sensor model energy consumption. Despite this additional energy consumption, both the ATC method and our framework reduce the overall system energy consumption. This reduction is achieved because both methods perform preliminary processing on low-power devices, which facilitates subsequent inference. As shown in the figure, our proposed framework helps consume less than 18% energy in all settings compared to the conventional system, and less than 25% energy compared to the ATC method.

Although the ATC method also employs a near-sensor model as a feature extractor, it transmits the features of all frames to the server for inference. In contrast to the ATC approaches, our framework does not transmit abstract features of frames but only transmits the original frames containing useful information, therefore, significantly reducing the number of frames sent to the server. While the size of data transmitted per frame may be larger compared to the ATC approaches, our framework involves the transmission of fewer frames, ultimately resulting in a reduction in the overall amount of data transmitted. Additionally, the complex machine learning model on the server in the ATC method only has access to abstract features, therefore, the downstream tasks, such as object detection and segmentation, rely on a single model. However, our near-sensor model sends the original FoIs to the server, allowing for the deployment of complex models specifically tailored to each task. This results in superior performance for each individual task.

The normalized energy consumption breakdown is depicted in Figure 5. The conventional system, which consumes the most energy, is normalized as 1, with the others adjusted accordingly. Despite our framework introducing a negligible energy portion to the system (near-sensor model), it reduces the need for server-side inferences, resulting in a substantial decrease in overall system energy consumption compared to both the conventional system and the ATC method. it is capable of saving 87% on the energy consumption of the conventional system and keeps valuable information. While the ATC method eases server-side inferences, the number of

inferences remains high, leading to a higher server energy consumption compared with our system. On the other hand, our dual-camera collaboration reduces camera energy consumption compared to both the conventional system and the ATC method.

D. Parameter Impact Analysis

Figure 6**a** and **b** present heatmaps illustrating the impact of the key parameters on P_{miss} and P_{trans}. For both metrics, a lower value indicates better performance. Regarding P_{miss}, a higher value of T leads to a notable increase in the number of missed FOIs. However, incorporating a lazier deactivation scheme and increasing the minimum transmission frequency can mitigate the adverse effects associated with a higher value of T. When examining the left panel in Figure 6, N exerts a dominant influence on P_{miss}. As N increases, accuracy improves significantly, leading to a decrease in the percentage of misdetections. Conversely, raising the minimum transmission frequency has a predominantly negative impact on P_{trans}. However, adopting a high value of T can reduce the amount of data transmission.

We also examined the relationship between M and P_{trans} and found that our framework exhibits a clear advantage as M increases, in Figure 7. Specifically, as M increases from 5 to 50, our approach can save energy ranging from 65% to 92% compared to the conventional system. In contrast, the energy consumption of both the conventional method and the ATC method increases proportionally as the amount of data grows.

The Spearman correlation coefficients [54] of the parameters are presented in Figure 8. In the figure, a positive coefficient indicates a positive correlation between two variables, while a negative coefficient indicates a negative correlation. The magnitude of the coefficient reflects the strength of the association between the variables. Specifically, T shows a significant positive correlation with P_{miss}, indicating that as T increases, P_{miss} also increases. Conversely, N exhibits a significant negative correlation with P_{miss}, indicating that higher values of N are associated with lower values of P_{miss}. On the other hand, the coefficient between N and P_{trans} is only 0.044, suggesting that changes in N have little influence on the percentage of data transmission. This observation aligns with our analysis in Section II-D. Moreover, as the near-sensor model operates continuously, the energy reduction in our framework primarily stems from the decrease in data transmission and the resulting reduction in server-side inference. Given that N has a negligible effect on P_{trans}, it also has minimal impact on overall energy consumption. Additionally, an increase in f corresponds to an increase in P_{trans}, demonstrating a strong direct positive correlation.

E. Model Customization Analysis

We compared the performance of various lightweight near-sensor models mentioned in Section II-B.1. We evaluated the trade-off between the sensitivity and specificity of these models using receiver operating characteristic (ROC) curves and area under the curve (AUC), as illustrated in Figure 9**a**. Additionally, Table I displays the number of parameters and GFLOPS of each model. While the AUC of YOLOv5ns is only slightly lower than that of YOLOv5n, the reduction in model size is significant, with the number of parameters decreasing to only 6.2% of the latter. Furthermore, the detrimental effect resulting from the reduction in model size can be alleviated by incorporating the lazy sensor deactivation scheme and elevating the minimum transmission frequency.

TABLE I
MODEL PARAMETERS

Model Name	No. of Parameters	GFLOPS
YOLOv5 n	1,765,270 (100%)	4.2
YOLOv5 nm	433,190 (24.5%)	1.1
YOLOv5 ns	108,806 (6.2%)	0.4

TABLE II
DESIGN ACCELERATION ON AMD-XILINX ZCU104

	LUT	FF	BRAM	URAM	DSP
Total	84.9K	146.5K	224	40	844
Available	230.4K	460.8K	312	96	1728
Utilization	36.87%	31.80%	71.79%	41.67%	48.84%

We also investigated the influence of quantization on the model performance. YOLOv5n trained on the original loss is quantized into different bit precisions, i.e. 16-bit float point (*fp16*), 8-bit integer (*int8*), 5-bit integer (*int5*), 4-bit integer (*int4*). The performance of both the *fp16* and *int8* quantized models remains unaffected. However, as illustrated by Figure 9**b**, when we further reduce bit precision to *int5*, a slight degradation in AUC is observed (from 0.97 to 0.96), and a degradation in performance is noticeable when the model is quantized to *int4* (from 0.97 to 0.93).

As discussed in Section II-B.3, the simplified loss function reduces task difficulty, allowing the model to become more lightweight or be more aggressively quantized. Figure 9**c** illustrates the impact assessment of our tailored loss function, demonstrating that with the tailored loss, the model can achieve intensive quantization while maintaining comparable performance levels. The model trained on the adapted loss achieves a higher AUC score under the same level of aggressive quantization (*int4* quantization) compared to the model trained on the original loss. It achieves the same AUC as the model training on the original loss with the precision float *fp32*. Since *fp32* requires 32 bits (4 bytes) per parameter, while *int4* only requires 4 bits (0.5 bytes) per parameter, jointly adopting the quantization and the customized loss can make the model 8x smaller without losing performance.

F. Hardware Implementation

The setup on the sensor side is depicted in Figure 10**a**. A high-resolution camera (④), a low-resolution camera (③), and a Wi-Fi adaptor (②) are connected to the FPGA board (①) via cable to capture and transmit the frames to the server. In addition, a screen is utilized for visualizing the information captured by the camera.

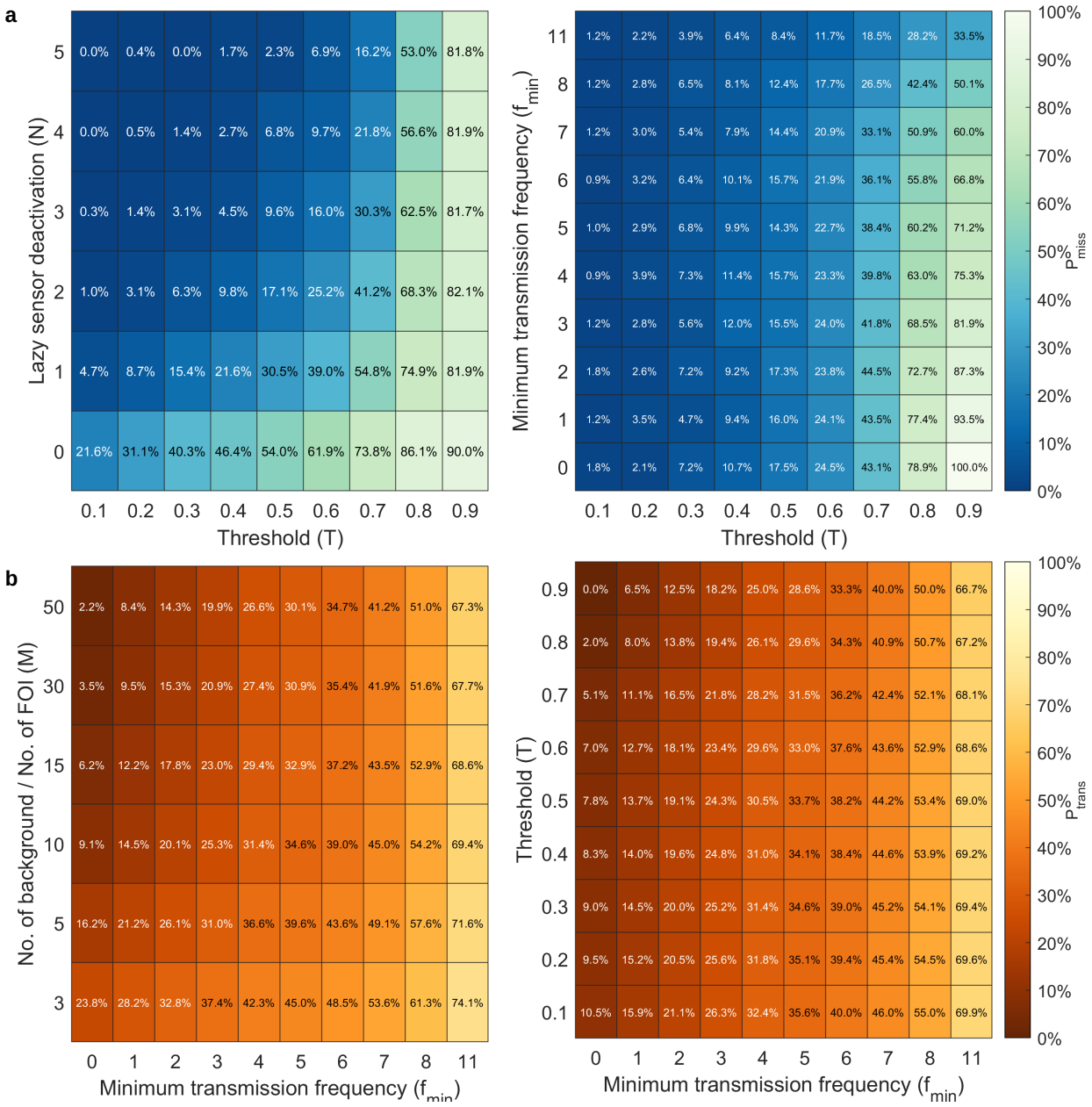


Fig. 6. Performance evaluation. **a.** Heatmaps that display the miss rate P_{miss} with different parameter combinations (threshold (T), the ratio (M) of the number of background frames to the number of FOIs, minimum transmission frequency (f_{min}), and the count (N) at which the sensor deactivates). **b.** Heatmaps that display the percentage of transmission P_{trans} with different parameter combinations.

To meet the requirements of the proposed scenario, the near-sensor model is deployed on a resource-limited low-power edge-level FPGA: AMD-Xilinx Zynq UltraScale+ MPSoC ZCU104 (ZCU104) [55]. FPGAs are semiconductor devices that are based on a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. Through hardware programming (such as Verilog or HLS), we can implement an ML accelerator on FPGA. The host program, executed on the ARM Cortex-A53 processor on the ZCU104's processing system (PS), was developed in Python. The communication between the processing system (PS) and the programmable logic (PL) is established through the AMBA Advanced eXtensible Interface (AXI). Here PS side is a host ARM processor and PL side is a reconfigurable logic. Our architecture design is implemented on the top of PL (reconfigurable logic).

To leverage hardware acceleration, we utilized the AMD-Xilinx deep learning unit (DPU) intellectual property (IP) as our hardware accelerator on the ZCU104's programmable logic (PL) side. Our model was integrated into the DPU using the Vitis AI framework [56]. Vitis AI is an ML compiler framework developed by AMD-Xilinx that automatically maps ML operations (such as convolution and fully connected layers) into Xilinx hardware IP. The Vitis AI version that we choose is 2.0. Furthermore, the cameras are connected to the host ARM CPU, which facilitates communication with the cloud server. TCP protocol is used as the communication protocol. Table II, we present the FPGA resource utilization result. In Figure 10**b**, we present the accelerator placement

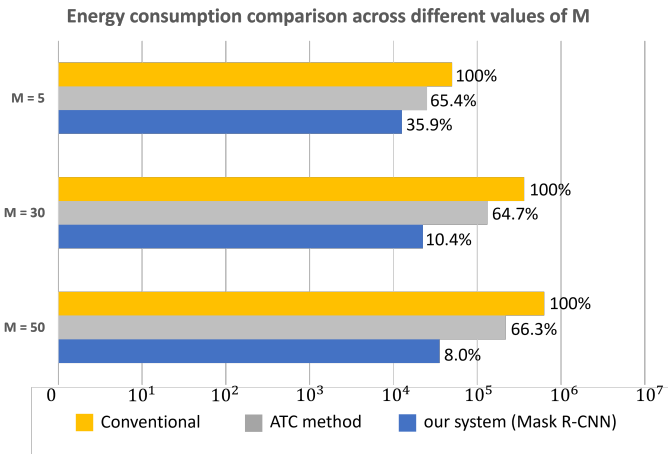


Fig. 7. Energy consumption comparison across different values of M. All servers are equipped with GeForce RTX 3090.

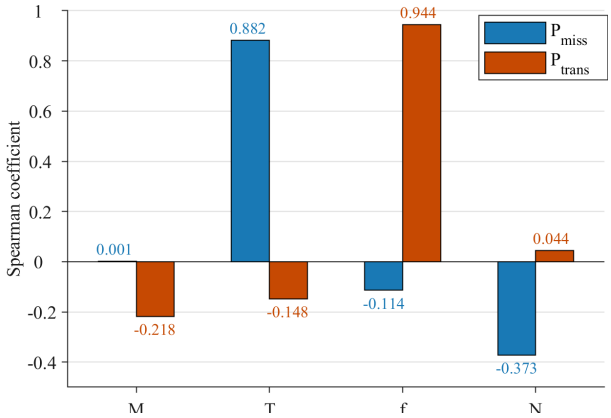


Fig. 8. Spearman coefficient of the parameters. The magnitude of the coefficient reflects the strength of the association between P_{miss}, P_{trans} and the variables (**T:** the confidence threshold of the model, **M:** the ratio of the number of background frames to the number of FOIs, **f:** minimum transmission frequency, **N:** The count at which sensor deactivates.)

layout on AMD Xilinx ZCU104 FPGA. The overview of our hardware platform is shown in Figure 11.

Considering the constraints of resources such as power and space, we sometimes need to reduce the acceleration performance of deep processing units (DPU) [57], [58]. For instance, in Table II, we select the parallelism for input, output, and pixel processing of convolution operations to be 16, 16, and 8, respectively. If the goal is to reduce power consumption and resource utilization, one strategy is to decrease computation parallelism. Another strategy involves employing knowledge distillation and quantization to minimize model size, thereby reducing the computational overhead of edge hardware accelerators [5], [59]. In this work, we concentrate on accelerating the near-sensor framework on edge FPGAs. However, we may also consider other AI computing platforms such as Google Edge TPU and NVIDIA Jetson Nano [60]. These chips facilitate easier programming of machine learning models but compromise the capability for hardware resource reconfiguration.

G. Other Applications

In addition to visual monitoring, our proposed framework can be readily applied for multiple other tasks, such as audio processing, and Radar monitoring.

For the audio processing task, we used the UrbanSound8K dataset [61], a public audio dataset for urban sound classification applications. It contains 10 classes, including car horn, gunshot, and dog bark. We focused on the gunshot and siren classes as the audio of interest. The dataset was reorganized and relabeled following the strategy outlined in Section III-A. The near-sensor model detects the audio of interest, while the server model classifies the specific class of that audio segment. A frequency-domain filter bank is applied to the audio signals, which are windowed in the time domain, to generate Mel spectrograms. These spectrograms are then fed into a CNN for classification. Compared to conventional methods, the system adopting our framework maintains comparable accuracy while consuming only 25% of the energy.

For radar monitoring, we evaluated framework using the CRUW dataset [62], a public camera-radar dataset designed for autonomous vehicle applications. The radar images in this dataset are captured by the TI AWR1843, which operates at approximately 30W [63]. The dataset was processed following the procedure outlined in Section III-A. Under the same deployment settings, the system using our framework achieved comparable performance while consuming only 18% of the energy required by the conventional system.

H. Security and privacy analysis

While our framework offers advantages such as reduced energy consumption and minimized bandwidth requirements, it necessitates an investigation of its security and privacy implications. Security considerations encompass data encryption both in transit and at rest. On the other hand, privacy concerns entail data minimization through near-sensor processing and anonymization techniques, user consent, and transparency regarding data usage. Compared with ATC methods that transmit abstract features, our framework transmits the original frames, thus sacrificing data encryption during transmission. However, it's worth noting that the conventional system also lacks data encryption during data transmission. This weakness can be mitigated by employing encryption techniques tailored specifically to image data. Additionally, both ATC methods and our framework introduce an extra near-sensor model component. Since the model is situated near the sensor and only processes incoming data locally, it does not leak any information, ensuring data safety.

IV. Limitations

While our framework is highly effective, there are a few limitations to consider for further improvements in the future works:

1) **Initial Training and Labeled Data Requirement:** Deploying the framework necessitates training the near-sensor model with labeled data. In some scenarios, obtaining labeled data may be challenging, or it might

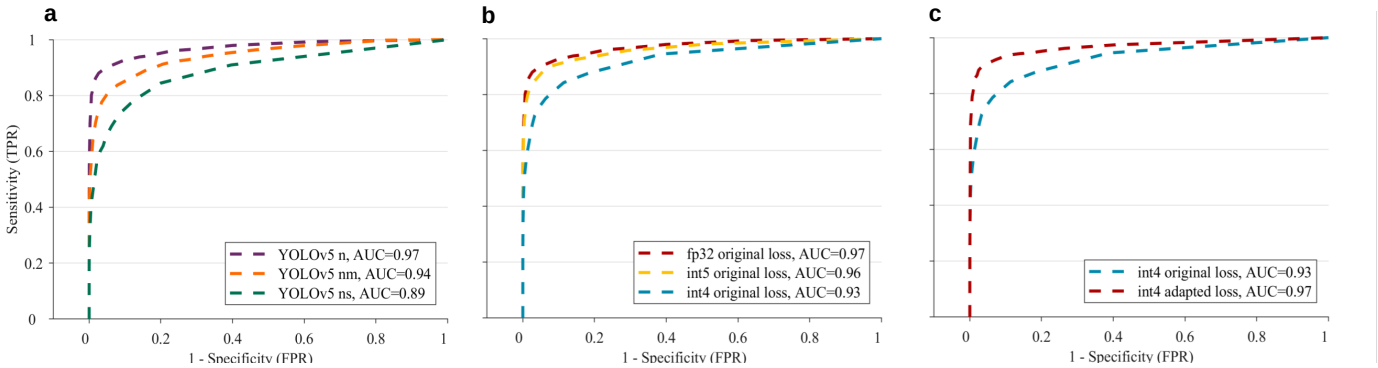


Fig. 9. Model comparison. **a** ROC curves of three lightweight models. **b** ROC curves of the models with different quantization trained by original loss. **c** The ROC curves of the model subjected to *int4* quantization, trained with our adapted loss function and the original loss function.

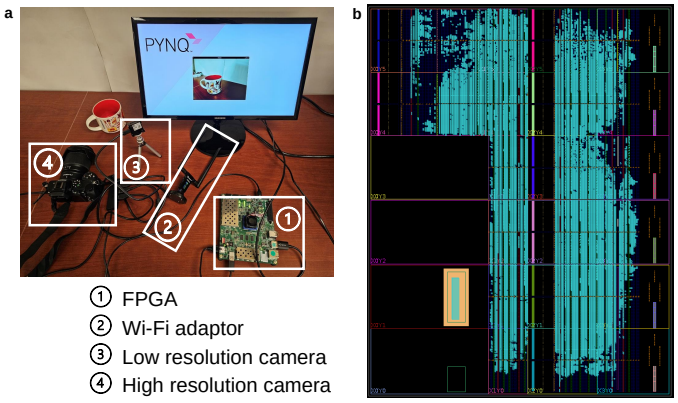


Fig. 10. Experiment setup. **a.** Sensor side setup. **b.** Accelerator placement layout on AMD Xilinx ZCU104 FPGA.

be difficult to fully cover the data distribution of sensor data. This introduces an initial cost associated with the deployment of the near-sensor module. However, after this initial cost, the framework can save substantial energy, especially on the server side.

2) **Accuracy Trade-offs:** Due to the lightweight nature of the near-sensor model, its accuracy may be lower than that of the original, more complex model. Although we have proposed schemes such as lazy deactivation and maintaining a non-zero transmission frequency to mitigate possible misdetections, the near-sensor model's accuracy is still slightly lower than the server-side model. While this loss of accuracy is negligible in many scenarios, it becomes critical in applications where high accuracy is paramount. In such cases, the near-sensor model may need to be less lightweight, which would reduce the energy savings.
3) **Environmental Constraints:** The performance of our framework is dependent on the ratio of background frames to FOIs. In environments where this ratio is lower, the energy savings and efficiency improvements may not be as significant.

In future work, we aim to enhance the performance of the near-sensor model while maintaining its low energy consumption and reducing the initial deployment cost of the framework.

V. Discussion

In this article, we introduce a novel framework for intelligent sensing that addresses some of the challenges associated with analyzing large-scale sensor data using complex ML models. Our framework is designed based on the observation that in many IoT applications, only a small proportion of sensor data conveys information of interest. Therefore, our framework intelligently selects the data generated by the sensors and only transmits and analyzes the data with useful information. It employs a near-sensor model to detect information of interest and control the data transmission, and a complex model located in the server to implement more sophisticated inference. The near-sensor model and the minimum transmission frequency are beneficial to reduce the energy and storage requirements, with a focus on decreasing the transmission frequency when no useful information is detected.

We set up the system with our framework on a low-power FPGA and evaluated the performance. The experimental results demonstrate that our framework significantly reduces total energy consumption and storage usage to less than 10% of that of conventional systems while retaining over 95% of useful information. Furthermore, we customized the model architecture and the loss function to suit our specific scenario and implemented quantization to achieve additional model compression. By jointly applying the customized loss function and quantization, the near-sensor model achieves an 8x reduction in size without any loss in performance.

We also investigated the key factors influencing the framework's effectiveness. Instead of completely halting data transmission when no FoI is detected, we maintain a non-zero minimum transmission frequency. This ensures regular, low-frequency transmission to the server even in the absence of FoIs, thereby benefiting the integrity of the useful information. Additionally, our lazy sensor deactivation scheme leverages the temporal correlation between adjacent frames, achieving a balance between resource consumption and accuracy. Furthermore, our proposed framework demonstrates greater effectiveness as the ratio of background frames to FoIs increases.

In addition to the evaluations under the visual monitoring scenario, we extended the framework to other tasks, e.g., audio processing, and Radar monitoring. The results show the

versatility and applicability of our framework under different scenarios.

We also discussed the limitations of our current framework and outlined potential directions for future improvements.

DATA AVAILABILITY STATEMENT

The dataset Microsoft COCO object detection for this study can be found in [50]. The raw data supporting the conclusion of this article will be made available by the authors, without undue reservation.

ACKNOWLEDGMENT

We express our gratitude to Andrew Ding for the discussion and meticulous proofreading to improve the earlier version of the manuscript.

APPENDIX I
SUPPLEMENTARY MATERIAL

Video Demo

Our research includes a video demonstration showcasing the results. In the demo, our model detects the animals appearing in the frames. The video can be accessed at the following link: `https://drive.google.com/file/d/1-IpRLfd8Ym38p8APCJgxNq5igiK5ARa5/view?usp=sharing`

Algorithm 1 Intelligent data transmission

Require: YOLO prediction(y), Lazy sensor deactivation count(N), Camera refresh rate(f_r), Minimum transmission frequency(f_{min}), $C_1, C_2, C_3 = 0, 0, 0$

Ensure: Transmission decision(D)

if $y == 1$ **then**
  $C_1, C_2, C_3 = 0, 0, 0$
  return $D = 1$
else
  if $C_3 == 0$ **then**
    $C_1 = C_1 + 1$
    if $C_1 \leq \max(1, \frac{N}{2^{C_2}})$ **then**
      return $D = 1$
    else
      $C_1, C_2, C_3 = 0, C_2 + 1, C_3 + 1$
      return $D = 0$
    end if
  else
    $C_3 = C_3 + 1$
    if $C_3 == f_r / f_{min}$ **then**
      $C_1, C_3 = C_1 + 1, 0$
      return $D = 1$
    else
      return $D = 0$
    end if
  end if
end if

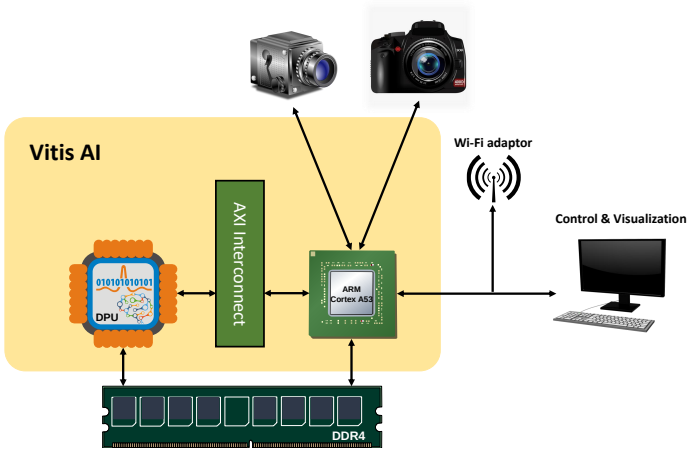


Fig. 11. Hardware platform overview.

REFERENCES

[1] M. M. Sadeeq *et al.*, "Iot and cloud computing issues, challenges and opportunities: A review," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 1–7, 2021.

[2] L. Yang *et al.*, "Iot data analytics in dynamic environments: From an automated machine learning perspective," *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105366, 2022.

[3] Y. Djenouri *et al.*, "Sensor data fusion for the industrial artificial intelligence of things," *Expert Systems*, vol. 39, no. 5, p. e12875, 2022.

[4] S. Yun, H. Chen, R. Masukawa, H. Errahmouni Barkam, A. Ding, W. Huang, A. Rezvani, S. Angizi, and M. Imani, "Hypersense: Hyperdimensional intelligent sensing for energy-efficient sparse data processing," *Advanced Intelligent Systems*, p. 2400228, 2024.

[5] Y. Ni *et al.*, "Heal: Brain-inspired hyperdimensional efficient active learning," *arXiv preprint arXiv:2402.11223*, 2024.

[6] V. Tsakanikas *et al.*, "An intelligent model for supporting edge migration for virtual function chains in next generation internet of things," *Scientific reports*, vol. 13, no. 1, p. 1063, 2023.

[7] M. Kumari *et al.*, "Deep learning techniques for remote sensing image scene classification: A comprehensive review, current challenges, and future directions," *Concurrency and Computation: Practice and Experience*, p. e7733, 2023.

[8] H. Chen *et al.*, "Taskclip: Extend large vision-language model for task oriented object detection," *arXiv preprint arXiv:2403.08108*, 2024.

[9] G. Wang *et al.*, "Event-triggered online energy flow control strategy for regional integrated energy system using lyapunov optimization," *International Journal of Electrical Power & Energy Systems*, vol. 125, p. 106451, 2021.

[10] C.-F. Liu *et al.*, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, 2019.

[11] Z. Sun *et al.*, "Cloud-edge collaboration in industrial internet of things: A joint offloading scheme based on resource prediction," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 014–17 025, 2021.

[12] Y.-H. Chiang *et al.*, "Joint cotask-aware offloading and scheduling in mobile edge computing systems," *IEEE Access*, vol. 7, pp. 105 008–105 018, 2019.

[13] M. Chen *et al.*, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[14] Y. Wang *et al.*, "Cooperative task offloading in three-tier mobile computing networks: An admm framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.

[15] Z. Zheng *et al.*, "A stackelberg game approach to proactive caching in large-scale mobile edge networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5198–5211, 2018.

[16] Y. Sun *et al.*, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1960–1971, 2018.

[17] S. Yu *et al.*, "Computation offloading for mobile edge computing: A deep learning approach," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–6.

[18] Z. Ali *et al.*, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149 623–149 633, 2019.
[19] M. Issa *et al.*, "Hyperdimensional hybrid learning on end-edge-cloud networks," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*, 2022, pp. 652–655.
[20] H. Yang *et al.*, "Brainiot: Brain-like productive services provisioning with federated learning in industrial iot," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2014–2024, 2021.
[21] A. Biswas *et al.*, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.
[22] C. Lammie *et al.*, "Low-power and high-speed deep fpga inference engines for weed classification at the edge." *IEEE Access*, vol. 7, pp. 51 171–51 184, 2019.
[23] H. Chen *et al.*, "Scalable and interpretable brain-inspired hyperdimensional computing intelligence with hardware-software co-design," in *2024 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2024, pp. 1–8.
[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
[25] H. Bay *et al.*, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 404–417.
[26] S. Leutenegger *et al.*, "Brisk: Binary robust invariant scalable keypoints," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2548–2555.
[27] N. Tishby *et al.*, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.
[28] L. Xiang *et al.*, "Compressed data aggregation for energy efficient wireless sensor networks," in *2011 8th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks*. IEEE, 2011, pp. 46–54.
[29] F. Schroff *et al.*, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
[30] Y. Sun *et al.*, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.
[31] A. M. Ghosh *et al.*, "Edge-cloud computing for internet of things data analytics: Embedding intelligence in the edge with deep learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2191–2200, 2021.
[32] B. Yogameena *et al.*, "Deep learning-based helmet wear analysis of a motorcycle rider for intelligent surveillance system," *IET Intelligent Transport Systems*, vol. 13, no. 7, pp. 1190–1198, 2019.
[33] W. Huang *et al.*, "Exploration of using a pressure sensitive mat for respiration rate and heart rate estimation," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2021, pp. 298–301.
[34] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.
[35] J. Deng, "A large-scale hierarchical image database," *Proc. of IEEE Computer Vision and Pattern Recognition, 2009*, 2009.
[36] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[37] R. Girshick *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
[38] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[39] S. Ren *et al.*, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.
[40] J. Dai *et al.*, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.
[41] L. Deng *et al.*, "Lightweight aerial image object detection algorithm based on improved yolov5s," *Scientific Reports*, vol. 13, no. 1, p. 7817, 2023.
[42] M. Zahrawi *et al.*, "Improving video surveillance systems in banks using deep learning techniques," *Scientific Reports*, vol. 13, no. 1, p. 7911, 2023.
[43] A. Alqahtani *et al.*, "Literature review of deep network compression," in *Informatics*, vol. 8, no. 4. MDPI, 2021, p. 77.
[44] A. Goel *et al.*, "A survey of methods for low-power deep learning and computer vision," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–6.
[45] C. N. Coelho *et al.*, "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," *Nature Machine Intelligence*, vol. 3, no. 8, pp. 675–686, 2021.
[46] I. Chakraborty *et al.*, "Constructing energy-efficient mixed-precision neural networks through principal component analysis for edge intelligence," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 43–55, 2020.
[47] H. Wu *et al.*, "Integer quantization for deep learning inference: Principles and empirical evaluation," *arXiv preprint arXiv:2004.09602*, 2020.
[48] F. Cardinaux *et al.*, "Iteratively training look-up tables for network quantization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 860–870, 2020.
[49] G. Caravagna *et al.*, "Lazy security controllers," in *International Workshop on Security and Trust Management*. Springer, 2012, pp. 33–48.
[50] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.
[51] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
[52] K. He *et al.*, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[53] Y. Matsubara *et al.*, "Split computing for complex object detectors: Challenges and preliminary results," in *Proceedings of the 4th International Workshop on Embedded and Mobile Deep Learning*, 2020, pp. 7–12.
[54] C. Spearman, "The proof and measurement of association between two things." 1961.
[55] A. Xilinx, "Zynq ultrascale," 2023, https://www.xilinx.com/products/boards-and-kits/zcu104.html [Accessed: 12/12/2023].
[56] V. Kathail, "Xilinx vitis unified software platform," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2020, pp. 173–174.
[57] H. Chen *et al.*, "Hypergraf: Hyperdimensional graph-based reasoning acceleration on fpga," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2023, pp. 34–41.
[58] H. Lee *et al.*, "Comprehensive integration of hyperdimensional computing with deep learning towards neuro-symbolic ai," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.
[59] H. E. Barkam *et al.*, "Reliable hyperdimensional reasoning on unreliable emerging technologies," in *2023 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.
[60] W. Huang *et al.*, "Ecosense: Energy-efficient intelligent sensing for inshore ship detection through edge-cloud collaboration," *arXiv preprint arXiv:2403.14027*, 2024.
[61] J. Salamon *et al.*, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1041–1044.
[62] Y. Wang *et al.*, "Rethinking of radar's role: A camera-radar dataset and systematic annotator via coordinate alignment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2815–2824.
[63] W. Li *et al.*, "Real-time fall detection using mmwave radar," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 16–20.