Exploring In-Memory Accelerators and FPGAs for Latency-Sensitive DNN Inference on Edge Servers

Ali Suvizi, Suresh Subramaniam, Tian Lan and Guru Venkataramani
Department of Electrical and Computer Engineering, The George Washington University, Washington, DC, USA
{ali.suvizi, suresh, tlan, guruv}@gwu.edu

***Abstract*—Edge servers are frequently used in latency-sensitive environments, like search-and-rescue missions involving unmanned aerial vehicles (UAV) that have real-time processing needs. In this paper, we study system designs that address the challenges of reducing latency and optimizing the power usage using Processing-in-Memory (PIM) and Field-Programmable Gate Array (FPGA). Age of Information (AoI) is a key metric to measure the data freshness for processing the images captured in real time. Our experimental results show our architecture significantly boosts computational speed and energy efficiency. Through the integration of PIM and FPGA into our edge server, latency is significantly reduced, achieving a speed-up of $92\times$ for PIM, and further to just 0.02 ms for FPGA, a sharp decrease from 43.48 ms on CPUs. Power consumption for inference tasks on LeNet-5 model is 0.36W with PIM, down from 11.57W on a CPU, and to 5.22W with FPGA. These results show the effectiveness of in-memory accelerators and FPGAs in ensuring that information remains current and actionable. In addition, our system's capability to support UAVs notably improves the real-time IoT application scalability. Specifically, our accelerator enhanced edge server can manage $1.6\times$ more UAVs for VGG-8 model and up to $71\times$ more for LeNet-5 inference tasks, compared to CPU-only, demonstrating its robustness in edge computing.**

***Index Terms*—Edge Computing, Accelerators, Deep Neural Networks, Age of Information, Processing In-Memory**

I. Introduction

Deep Neural Networks (DNNs) are now widely used in computer vision and natural language processing among various applications. In particular, Unmanned Aerial Vehicles (UAVs), commonly known as drones, and self-driving cars frequently use DNNs for image processing applications. These domains consider it essential to maintain the freshness of data, which is measured by Age of Information (AoI) [1], also referred to as age, defined as the time elapsed since the generation of the latest delivered update.

The integration of image processing and drone technology has positive effects on improving real-time IoT applications such as traffic management for smart cities [2], search and rescue systems, and surveillance and environmental modeling [3], [4]. These advancements underscore the pivotal role of DNNs in enhancing the autonomy, efficiency, and effectiveness of UAV systems. On the other hand, drones have some limitations in terms of their computational capacity and energy constraints. The inherent complexity of DNN-based image processing imposes significant computational overhead, posing substantial demands on the system resources. Furthermore, due to the rapidly increasing number of IoT sensing devices, the corresponding sensor data produced is growing exponentially [5]. This growth not only has negative effects on the latency of analytics required by IoT applications but also raises concerns regarding the freshness of data, a vital factor in ensuring the relevance of information processed by these systems.

In the face of these challenges, edge computing has emerged as a promising paradigm, poised to bridge the gap between data generation and the need to process them in a timely manner. As such, processing the sensor data by decentralizing computational tasks and moving the processing closer to data sources is essential [6]. This approach not only aligns with the high computation and low-latency demands intrinsic to deep learning applications but also brings other benefits, including enhanced privacy, improved bandwidth efficiency, and scalability of the system. Moreover, within real-time systems, the significant volume of data transmitted by sensors to the edge accentuates the necessity for a robust accelerator unit to improve the accuracy and satisfy the freshness of data. These accelerators must combine high performance with low energy consumption to effectively manage the data.

In this paper, we explore the potential for emerging hardware accelerator designs that can be integrated into edge servers for improved processing. In our proposed system architectures, the UAVs/autonomous vehicles act as sensors, capturing and transmitting the data to edge nodes equipped with diverse processing units—Central Processing Units (CPUs), Processing-in-Memory (PIM) technologies, and Field-Programmable Gate Array (FPGA). Each of these processing units offers unique advantages in terms of speed and energy efficiency, making them suitable for different operational contexts within the UAV-edge system.

In summary, the key contributions of our paper are:

- We design and demonstrate an edge server architecture augmented with PIM and FPGA as accelerator units to efficiently perform the computations and lower energy consumption correspondingly. Our system model considers AoI as a constraint in order to check the data freshness for real-time applications.
- We implement and evaluate our accelerator designs on a combination of real system Xilinx FPGA, and a simulation tool named MNSIM 2.0 to show the augmented acceleration capabilities of our system architecture. Ad-

ditionally, we present a custom hardware design of a DNN using a hardware description language, optimized for performance and energy.
- We conduct experiments and analysis for our proposed design to show the effectiveness of our system by reducing the computation time and power consumption using hardware accelerators. Our results demonstrate better scalability—in the case of PIM, the edge server's ability to interface with UAVs has increased by at least $3\times$ for VGG-8-based inference and up to approximately $92\times$ for LeNet-5—enhancing the server's capacity to handle a higher number of drones, each of which can send their data back to the server.

II. Background and Related Works

A. Age of Information

The concept of AoI plays a pivotal role within the domain of real-time systems, particularly those reliant on timely data transmission and processing. AoI is defined as a metric that evaluates the freshness of information, providing a measure of the time elapsed since the generation of the latest received update packet at a destination node. This metric is particularly crucial in scenarios where the relevance and utility of information are directly tied to its timeliness [7].

Prior research [8] in this domain has predominantly concentrated on examining the effects of data transmission and queuing on AoI. However, in applications such as autonomous driving systems and UAVs, the scenario is markedly different. Here, updates, which may frequently comprise images, necessitate not only transmission to a controller but also thorough processing before the extraction of valuable information. This necessity introduces significant delays, attributable to the limited computational capabilities of local processors, thus affecting the AoI [9].

The AoI has been applied to different network models as a performance metric for various communication systems that timeliness of data is critical, e.g., trust-aware resource allocation schemes [10], scheduling in networks [11], and UAV-assisted communication systems [12], [13].

B. Accelerators

In the age of artificial intelligence (AI), Convolutional Neural Networks (CNNs) have shown remarkable effectiveness in many fields, such as object detection, and image classification [14]. In addition to the high computing accuracy, the amount of data and model size increase dramatically as the CNN models become more and more complex. In traditional von-Neumann architectures (e.g., CPU and GPU), the CNN computations cause massive data movements between memory and computing units, which consume more than 80% of the overall system energy and execution time, thereby exacerbating this problem [15]. In order to tackle the "memory wall", researchers have proposed novel PIM architectures that perform computations inside memory [16], [17].

FPGAs usually consist of a large array of programmable logic blocks interconnected by an array of programmable data paths to implement custom hardware that is ideally matched to specific computational problems. FPGA chips find utility across a diverse range of applications, encompassing data-centric operations like image processing [18], as well as computation-centric tasks within parallel computing architectures [19]. This widespread adoption is attributable to the chips' exceptional programmability, which enables them to execute calculations rapidly and with minimal energy expenditure.

III. Design Overview

Most modern computing systems are predominantly optimized for computing performance and suffer from memory bottlenecks. These design choices go directly against at least three issues that cause performance, scalability, and energy bottlenecks: (1) data accesses in memory-intensive applications can violate latency constraints, (2) energy consumption is a key constraint in embedded computing platforms, such as UAVs and autonomous vehicles, (3) data movement, especially off-chip to on-chip, is very expensive and bandwidth limitations can present challenges.

In real-time system domains, particularly those that utilize drones as sensors for image processing applications, the limited computational capabilities of drones necessitate the offloading of tasks to a more powerful edge server. Additionally, a significant volume of data is transmitted from drones to the server, aimed at enhancing accuracy. Within such systems, there usually exist stringent deadlines for data processing and decision-making. Consequently, it is imperative to expedite calculations and data processing to avoid reliance on outdated information, a decidedly undesirable outcome. As a result, having the data on the monitor as fresh as possible is crucial.

In addressing the challenges in edge computing systems, especially those associated with real-time image processing applications using drones, our approach begins with a comprehensive evaluation of trade-offs within the edge server environment. These trade-offs encompass latency, power consumption, and bandwidth, all while upholding data freshness as an essential system constraint while calculating the AoI. In response to these needs, we propose and study the system designs that augment the edge server's capabilities. This involves the integration of PIM and FPGA alongside traditional CPU. We augment with one accelerator at a time to identify the most effective configuration for executing image processing tasks with high precision, minimal latency, and reduced power consumption, thereby ensuring the utmost data freshness.

By systematically evaluating the impact of incorporating each processing unit individually, we seek to understand the best hardware accelerator designs to alleviate the performance bottlenecks, scalability, and energy efficiency, presenting a tailored solution to the challenges prevalent in processing environments with drones embedded in them.

Processing In-Memory: PIM architectures are particularly beneficial for data-intensive applications because they can perform computations directly within or very near the memory. This significantly reduces the need for data movement

between the processor and memory. For image processing tasks, where data throughput and efficiency are critical, PIM can offer lower latency with higher bandwidth. Moreover, by minimizing the data movement, PIM also helps in reducing overall energy consumption, which is crucial for edge devices where power availability might be limited. By considering the critical importance of data freshness for real-time UAV-based applications in edge computing, it is necessary to perform computations for each task as fast as possible to avoid missing deadlines. The design of a server augmented with PIM is shown in Figure 1.

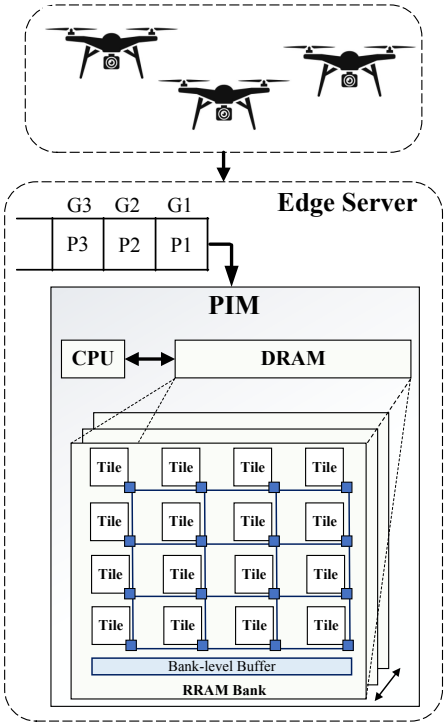

Fig. 1. Edge Server Augmented with PIM: The illustration shows the edge server's task flow, with 'G1' to 'G3' as task generation time and 'P1' to 'P3' as the corresponding tasks queued for PIM processing.

The edge server augmented with PIM is designed with multiple levels of Resistive Random-Access Memory (RRAM) banks, where each bank houses a network of Non-volatile Memory (NVM) tiles that resemble a Network-on-Chip (NOC) structure for optimized control and data flow. Individual tiles are tasked with processing separate layers of CNN computations, enabling efficient data handling, and when needed, larger layers are divided and processed across multiple tiles in parallel. A data forwarding unit (blue boxes) associated with each tile manages the integration of data from multiple sources, performing necessary computations like merging. The tiles, configurable for different CNN layers, are equipped with processing elements (PEs) to fulfill the computational needs of each layer, facilitating a flexible, streamlined processing environment within the PIM framework.

Utilizing PIM architectures in edge computing effectively addresses the challenge of ensuring data freshness in real-time applications. PIM's ability to perform computations close to memory significantly reduces latency and increases bandwidth, crucial for tasks like image processing. This architecture minimizes unnecessary data movement, speeding up processing times to meet critical deadlines while also conserving energy. The reduction in energy consumption is particularly valuable for edge devices with limited power resources.

To evaluate the integration of PIM with edge servers and study how it enhances the computational performance, we used MNSIM 2.0 [20], which is a behavior-level modeling tool for memristor-based neuromorphic computing systems. It can simulate the hardware performance and neural network computing accuracy of different PIM architectures.

Field Programmable Gate Arrays: FPGAs are applicable to accelerate many different workloads at the edge. Since the FPGA designs are highly optimized for specific applications, their performance and power can be well suited to the constraints of edge computing. In fact, FPGAs are more useful for compute-intensive applications in which there is a lot of computation that can be run using FPGA resources simultaneously. On the other hand, in the case of data-intensive applications such as performing inference with CNN, employing FPGAs for large models becomes impractical due to the complexity of implementation and the overhead associated with reading data from off-chip memory. Although, in simple DNNs like LeNet-5 [21], where all parameters can be stored within on-chip memory, including Block RAMs (BRAMs) and Look-Up Tables (LUTs), utilizing FPGAs becomes viable. This approach enables rapid computation with reduced energy consumption by eliminating the necessity for accessing off-chip memory.

Taking these factors into account, by efficiently executing image processing computations with minimal energy consumption, we can also fulfill the requirement of data freshness. This ensures that tasks received by the edge server are completed before the target's deadline, thereby maintaining the real-time characteristics of the system. The design of the augmented server with FPGA is in Figure 2.

To facilitate the implementation of the LeNet-5 model for image processing on FPGA, we utilized the Xilinx FPGA board. This involved coding the model in the hardware description language (VHDL), allowing for precise control over the hardware architecture. Through the synthesis and implementation processes, we were able to tailor the FPGA configuration specifically for the LeNet-5 architecture, enabling optimized performance for image processing tasks. FPGA provided the necessary framework to execute these operations, offering a platform where the computational efficiency and flexibility of FPGA could be harnessed effectively for deep learning applications. This approach underscores its capability to enhance processing speed and efficiency in real-time image analysis.

We note that it is possible to leverage the complementary strengths of PIMs and FPGAs, where the system can dynamically allocate processing tasks in a manner that optimizes both time and energy, all while satisfying the stringent requirements

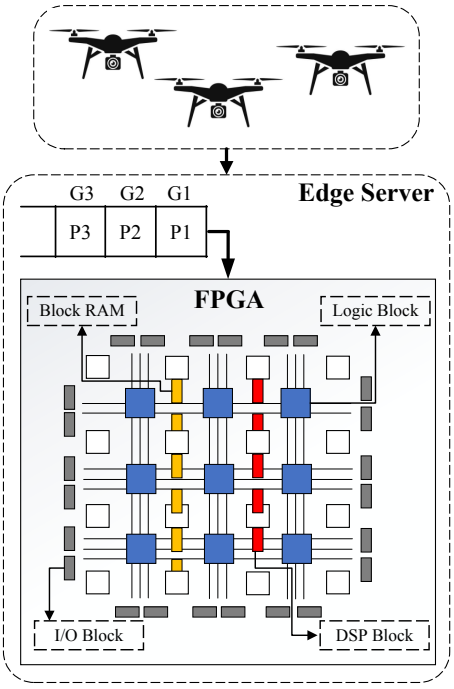

Fig. 2. Edge Server Augmented with FPGA: The illustration shows the edge server's task flow, with 'G1' to 'G3' as task generation time and 'P1' to 'P3' as the corresponding tasks queued for FPGA processing.

imposed by AoI. This adaptive approach not only enhances the operational efficiency of UAVs in real-time applications but also heralds a new era of *sustainability* in DNN-powered systems. Thus, the integration of edge computing into DNN-based UAV systems represents a forward-looking strategy to navigate the complex interplay between computational demands, energy constraints, and the imperatives of data freshness, paving the way for more agile, efficient, and responsive aerial intelligence platforms.

Age of Information: Data freshness is often identified as a key constraint for real-time systems and is measured by the AoI. We defined AoI as the elapsed time from the generation of status information in the task queue to its processing completion at the edge server. Equation (1) shows the formulation of AoI. Figure 3 shows the AoI graph of the augmented edge server.

$$\Delta(t) = t - u(t) \tag{1}$$

In this equation, t is the current time and $u(t)$ represents raw (pre-processor) sensor input data generation time.

As shown in Fig. 3, initially, the AoI starts at a baseline level, denoted as Δ_0 or G_1, symbolizing the freshness of the first task's information. The AoI continues to increase until time t_1, when the first task completes its computation, signified by a sharp decrease or update in the AoI as fresh information becomes available. The subsequent rise in AoI after t_1 marks the aging of the second task's information, starting from its generation time G_2, and the process repeats for subsequent tasks.

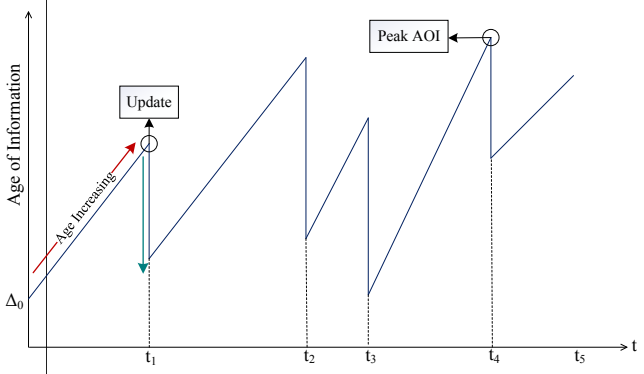

Fig. 3. AoI for Augmented Edge Server

Critical to our real-time image processing system is the concept of peak AoI, which we set as the maximum acceptable timeframe for processing tasks within the system to maintain data freshness. This peak AoI effectively acts as a deadline, ensuring that all tasks are processed within this timeframe to preserve the relevance and utility of the data for decision-making and operational actions. Should the processing extend beyond this peak AoI, the data risks becoming outdated, thus diminishing its value and adversely affecting the system's real-time performance and accuracy.

We established the peak AoI based on the data transmission bit rate to the server's queue. In scenarios that involve processing high-resolution data streams to enhance system accuracy, the bit rate is set at 30 frames per second [22], [23]. This rate crucially determines the speed at which data is generated and subsequently needs processing to ensure data freshness. 30 frames per second has a duration of 33.33 ms, indicating that, in this scenario, the peak AoI is 33.33 ms. By integrating an augmented edge server equipped with PIM and FPGA, we are well-positioned to effectively meet this constraint.

Moreover, when we take into account the data rate for inputs to the queue (arrival rate) and the outputs from the queue to PIM or FPGA for computation (service rate), the augmented edge server's low latency in performing computations enables the maintenance of the arrival rate while processing more tasks simultaneously. This approach opens up opportunities to enhance the system's scalability by increasing the number of drones sending data to the server, thereby expanding the system's capacity to handle larger volumes of data efficiently.

IV. Experiments and Results

A. Experimental Setup

We study various accelerators to enhance the edge server, including PIM and FPGA, and analyze their impact on the system's performance. Our experiments are designed to evaluate the efficacy of these accelerators in reducing latency and power consumption while maintaining the AoI within acceptable limits for real-time processing. In the following, we describe the details of the DNN, the dataset, and our evaluation testbeds.

- **Convolutional Neural Network:** We employed a range of CNNs, from the relatively simple LeNet-5 to more complex neural networks such as AlexNet [24] and VGG-8, for image processing tasks. Initially, the models were trained to obtain the pre-trained models and the necessary parameters for inference. Subsequently, we applied post-training quantization to these models and parameters, preparing them for deployment on PIM systems and FPGA.
- **Dataset:** The CIFAR-10 dataset [25] has been used for evaluating the trade-offs associated with augmenting the edge server with PIM and FPGA technologies. This dataset, widely recognized in machine learning and computer vision research, comprises 60,000 color images with a resolution of 32x32 pixels, distributed across 10 distinct classes, each class containing 6,000 images.
- **Experimentation Platform:** Our computational experiments on the CPU were conducted using an Intel® Core™ i7-1065G7 Processor with a 2.6GHz clock speed and 16 GB of RAM. For assessing the latency and power consumption of inference on PIM, we utilized the MNSIM 2.0 simulator [20]. Additionally, XCVU440-FLGA2892-2-i, which belongs to the Virtex® UltraScale™ was used to implement the LeNet-5 model using VHDL, which facilitated the exploration of various acceleration options on the edge server.

B. Experimental Results and Analysis

In this section, we present the experimental results demonstrating our system's effectiveness in reducing computation time and power consumption through the utilization of hardware accelerators.

Initially, we compare the latency and power consumption results of an edge server equipped with a CPU to those of an edge server augmented with PIM, using LeNet-5, AlexNet, and VGG-8 as CNN models. Subsequently, we evaluate an edge server augmented with FPGA for latency and power consumption using LeNet-5. This allows for a comparison between the edge server equipped with a CPU, its augmented version with PIM, and the performance of the FPGA-augmented edge server.

Table I presents a comparative analysis of the latency and efficiency of different hardware configurations within an edge computing server. The results are quantified in terms of latency, measured in milliseconds, and the speed-up factor, highlighting the efficiency with which PIM outperforms CPU-based computations.

This acceleration is substantiated by the MNSIM 2.0 simulation tool's features. MNSIM's hierarchical PIM modeling structure significantly reduces latency by optimizing the direct execution of neural network algorithms within memory, thereby mitigating data movement delays. Furthermore, the tool's dedicated support for algorithm-level optimizations, such as quantization for PIM, helps to take advantage of the unique efficiencies of PIM architectures. Additionally, MNSIM's capability to emulate both analog and digital PIM architectures through its unified memory array model provides a versatile evaluation platform that underlines the full spectrum of PIM technology benefits. These elements collectively drive the pronounced performance enhancements observed with PIM, showcasing the transformative potential of PIM technologies in elevating computational speed in edge computing servers.

TABLE I
CPU-PIM LATENCY

Model	Latency (ms)		Speed-up
	CPU	PIM	(CPU/PIM)
LeNet-5	43.48	0.47	92
AlexNet	54.53	2.40	23
VGG-8	60.84	20.31	3

Our results in Table I contrast the performance metrics of an edge server using a conventional CPU against the same server augmented with PIM technology. We used a suite of CNN models for this assessment—specifically LeNet-5, AlexNet, and VGG-8.

Table II provides a detailed comparison of power consumption between CPU and PIM across various CNN models utilized within an edge computing framework.

TABLE II
CPU-PIM POWER CONSUMPTION

Model	Power (W)	
	CPU	PIM
LeNet-5	11.57	0.36
AlexNet	15.84	8.12
VGG-8	35.62	24.20

As a second step, we implemented the LeNet-5 at the Register Transfer Level (RTL) using the VHDL language. This step was undertaken to scrutinize the capabilities of FPGA technology concerning both execution time and power consumption. The pertinent findings regarding latency and power metrics for the LeNet-5 are shown in Table III.

TABLE III
COMPARISON OF LENET MODEL LATENCY AND POWER

Model	Latency (ms)	Power (W)
LeNet-CPU	43.48	11.57
LeNet-FPGA	0.02	5.22

As shown in the table, the latency and power consumption metrics illustrate a marked improvement on the FPGA platform. The decision to implement LeNet-5, rather than more complex CNN models, stems from two considerations. First, the LeNet-5 model can be fully accommodated within the on-chip memory resources of the FPGA, which is not typically feasible for larger models. Second, the complexity of implementing DNNs into a hardware description language for FPGA realization, such as VHDL, increases substantially with the model size. Consequently, the FPGA implementation of LeNet-5 demonstrates the potential of FPGAs for efficient

DNN inference, leading to faster and more energy-efficient edge computing.

Table IV shows the FPGA resource utilization after the implementation of the LeNet-5.

TABLE IV
FPGA RESOURCE UTILIZATION

Resources	LUT	FF	DSP	BRAM	I/O
Available	2532960	5065920	2880	2520	2892
Utilized	311115	310179	50	0	96
Utilization (%)	12.28	6.12	1.73	0	3.31

Our findings indicate that with the original CPU-only setup, our edge server could interface with a finite number of drones—roughly one every 33.33 ms. With our accelerator-based enhancements reducing the processing latency significantly, our results show that server's computational capacity could be enhanced. Quantitatively, we observe that the system can now accommodate at least $1.6\times$ more drones with VGG-8 and up to $71\times$ more with LeNet-5, indicative of improved scalability as determined by our experiments. This remarkable increase in drone support—without extending the temporal constraints—suggests an inherent elevation in system accuracy. This signifies the benefits of our approach to real-time edge computing, demonstrating the profound impact of PIM and FPGA integration on system performance.

V. CONCLUSION

In conclusion, our studies successfully demonstrate the potential of PIM and FPGA technologies to significantly augment edge servers, catering to the demands of latency-sensitive and power-aware real-time applications. The integration of these accelerators into our edge computing framework has shown a notable reduction in computation time—latency improvements of up to $92\times$ for PIM. By using AoI as a metric for data freshness, we have assured the timely processing of UAV-sourced data, which in turn has significantly enhanced system scalability. Compared to a CPU-only system, the edge server, equipped with PIM, is now capable of supporting $1.6\times$ to $71\times$ more UAVs for VGG-8 and LeNet-5 models respectively. This enhancement not only paves the way for more dynamic and responsive UAV operations but also opens new avenues for better sustainability when performing latency-sensitive DNN inference tasks in edge computing.

ACKNOWLEDGMENT

This work was supported in part by NSF grant CNS-2219180.

REFERENCES

[1] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" Proc of IEEE INFOCOM, pp. 2731–2735, 2012.

[2] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," IEEE Wireless Communications, vol. 26, no. 1, pp. 87–93, 2019.

[3] S. M. Hamylton et al., "Evaluating techniques for mapping island vegetation from unmanned aerial vehicle (UAV) images: pixel classification, visual interpretation and machine learning approaches," Int. J. Appl. Earth Obs. Geoinf., 2020.

[4] H. Shafaghi, M. Kiani, A. Amirany, K. Jafari, and M. H. Moaiyeri, "A fast and light fingerprint-matching model based on deep learning approaches," Journal of Signal Processing Systems, vol. 95, no. 4, pp. 551-558, 2023.

[5] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in Internet of Things," Computer Networks, vol. 129, pp. 459–471, 2017.

[6] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah, "Edge computing for the internet of things," IEEE Network, vol. 32, no. 1, pp. 6–7, 2018.

[7] A. Kosta, N. Pappas, V. Angelakis, Age of information: A new concept, metric, and tool, Found. Trends Netw. 12 (3) (2017) 162–259.

[8] R. Talak, S. Karaman, and E. Modiano, "Minimizing age-of-information in multi-hop wireless networks," 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 486–493, 2017.

[9] R. Li, Q. Ma, J. Gong, Z. Zhou, X. Chen, Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time IoT applications, IEEE Internet Things J. 8 (19) (2021) 14471–14484.

[10] N. Torkzaban and J. S. Baras. Trust-aware service function chain embedding: A pathbased approach. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pages 31–36, 2020.

[11] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides. Asymptotically optimal scheduling policy for minimizing the age of information. In 2020 IEEE International Symposium on Information Theory (ISIT), pages 1747–1752, 2020.

[12] G. Ahani, D. Yuan, and Y. Zhao. Age-optimal uav scheduling for data collection with battery recharging. IEEE Communications Letters, pages 1–1, 2020.

[13] Roy D Yates. Status updates through networks of parallel servers. In 2018 IEEE International Symposium on Information Theory (ISIT), pages 2281–2285. IEEE, 2018.

[14] S. Safa aldin, N. B. Aldin, and M. Aykac, "Enhanced image classification using edge CNN (E-CNN)," The Visual Computer, vol. 40, no. 1, pp. 319-332, Jan. 2024.

[15] M. M. S. Aly et al., "The N3XT approach to energy-efficient abundant-data computing," Proc. IEEE, vol. 107, no. 1, pp. 19–48, Jan. 2019.

[16] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, 2020.

[17] J. Gómez-Luna, I. El Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: Experimental analysis and characterization of a real processing-in-memory system," IEEE Access, vol. 10, pp. 52565-52608, May 10, 2022.

[18] M. Pistellato, F. Bergamasco, G. Bigaglia, A. Gasparetto, A. Albarelli, M. Boschetti, and R. Passerone, "Quantization-Aware NN Layers with High-throughput FPGA Implementation for Edge AI," Sensors, vol. 23, no. 10, art. 4667, May 11, 2023.

[19] A. Suvizi, A. Farghadan, and M. S. Zamani, "A parallel computing architecture based on cellular automata for hydraulic analysis of water distribution networks," Journal of Parallel and Distributed Computing, vol. 178, pp. 11-28, Aug. 1, 2023.

[20] Z. Zhu, H. Sun, T. Xie, Y. Zhu, G. Dai, L. Xia, D. Niu, X. Chen, X.S. Hu, Y. Cao, and Y. Xie, "Mnsim 2.0: A behavior-level modeling tool for processing-in-memory architectures," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Mar. 2, 2023. doi: 10.1109/TCAD.2023.3251696.

[21] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

[22] Z. El Hadhri, C. Valderrama, and P. da Cunha Possa, "Image and video processing on FPGAs: An exploration framework for real-time applications," in IFAC Proceedings Volumes, vol. 43, no. 24, pp. 54-59, Jan. 1, 2010.

[23] M. Mahmoodpour, A. Amirany, M. H. Moaiyeri, and K. Jafari, "A learning based contrast specific no reference image quality assessment algorithm," in 2022 International Conference on Machine Vision and Image Processing (MVIP), 2022, pp. 1-4.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Adv. Neural Inf. Process. Syst., vol. 25, pp. 1–9, 2012.

[25] A. Krizhevsky, "The CIFAR-10 dataset," *Canadian Institute For Advanced Research*, 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html.