2024 IEEE International Conference on Pervasive Computing and Communications (PerCom) | 979-8-3503-2603-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/PERCOM59722.2024.10494426

Furcifer: a Context Adaptive Middleware for Real-world Object Detection Exploiting Local, Edge, and Split Computing in the Cloud Continuum

Matteo Mendula*, Paolo Bellavista* Marco Levorato†, and Sharon Ladron de Guevara Contreras†
*Department of Computer Science and Engineering, University of Bologna, Italy
†Donald Bren School of Information and Computer Sciences, University of California at Irvine, United States
{matteo.mendula,paolo.bellavista}@unibo.it
{sladrond, levorato}@uci.edu

***Abstract*—Modern real-time applications widely embed compute intense neural algorithms at their core. Current solutions to support such algorithms either deploy highly-optimized Deep Neural Networks at mobile devices or offload the execution of possibly larger higher-performance neural models to edge servers. While the former solution typically maps to higher energy consumption and lower performance, the latter necessitates the low-latency wireless transfer of high volumes of data. Time-varying variables describing the state of these systems, such as connection quality and system load, determine the optimality of the different computing configurations in terms of energy consumption, task performance, and latency. Herein, we propose Furcifer, a framework capable of dynamically adapting the cloud continuum computing configuration in response to the perceived state of the system. Our container-based approach incorporates low-complexity predictors that generalize well across operating environments. In addition, we develop a highly optimized split Deep Neural Network model, which achieves in-model supervised compression and enhances task offloading. Experimental results for object detection across diverse conditions, environments, and wireless technologies, show Furcifer's remarkable outcomes, including a 2x energy reduction, 30% higher mean Average Precision score than pure local computing, and a notable three-fold increase in frame per second rate compared to static offloading.**

***Index Terms*—Edge Computing, Machine Learning, Object Detection, Mobile agents, Image compression**

I. Introduction

The increasing adoption of Machine Learning (ML) solutions in a broad range of real-world application scenarios has highlighted the need for system architectures with high flexibility and adaptability. However, the usability of ML algorithms in practical settings is often hampered by computing and communication limitations not considered during their development and evaluation stages. Challenges include constrained computing capabilities and energy budget of mobile devices, as well as communication channel capacity. Local Computing (LC) and Edge Computing (EC) stand as the primary strategies for tackling the broad range of real-world heterogeneous tasks centered on the execution of complex data analysis and decision making algorithms. On the one hand, LC, that is, the execution of the ML algorithm onboard a mobile device, aims at the optimal interplay between software applications and hardware components to efficiently harness available onboard resources. This near-deterministic computing strategy comes at the price of reduced lifetime and limited computing capabilities. On the other hand, EC, where the ML tasks are offloaded to a compute-capable device positioned at the network edge, leverages high-performance communication and computing technologies to effectively support real-time applications. While EC promises higher computational capabilities and lower energy consumption to pervasive mobile systems, it requires a stable wireless link that, given the volatile nature of wireless channels, is not available consistently.

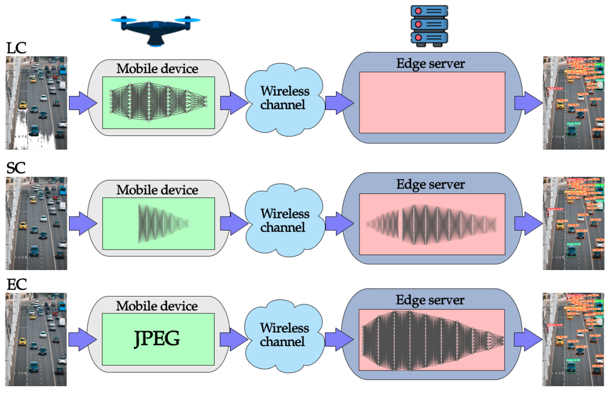

Fig. 1. Representation of computation distribution between Mobile Device (MD) and Edge Server (ES) for Local, Split, and Edge computing scenarios.

Recently, a third paradigm - Split Computing (SC), where sections of ML models optimized to facilitate offloading are allocated to the mobile device and edge server - emerged as a promising alternative to EC and LC (see Figure 1). Indeed, the most advanced SC frameworks, where specialized models embed neural encoder/decoder-like structures, result in minimal computing load to the mobile device, while considerably reducing network usage [1].

Despite the substantial progress made in all the computing paradigms pursued by academic and industrial efforts, none of aforementioned computing strategies can be deemed as universal solution in every scenario. We contend that the volatility and time-varying nature of a multitude of relevant parameters and state variables (e.g., channel quality, or network and server load) make online adaptation not just desirable to achieve optimal performance, but also necessary in practical deployments. However, adaptation of the computing configuration is explored only in academic and theory-driven investigations, which often overlook critical characteristics and issues emerging in real-world systems and deployments [2]–[4]. Moreover, despite its potential to become a critical computing configuration in many operational settings and system states, SC has yet to be evaluated beyond purely academic frameworks, making its ability to contend with EC and LC solutions unclear.

In response to these open technical challenges and deficiencies, we present *Furcifer*, an innovative middleware framework specifically designed to provide seamless adaptation of the computing modality in realistic application environments. Furcifer transparently monitors the state of the underlying system, evaluating at runtime the feasibility of EC, LC, and SC configurations in highly dynamic environments, and switch between them. The core of Furcifer is a new containerized approach that can effectively support the dynamic transition between EC, LC and SC. The proposed framework reduces context switch latency between different computing modalities – less than 2ms – by using storage as an additional and inexpensive resource.

In this article we evaluate Furcifer's applicability in Object Detection (OD); however, it's important to note that the framework is designed to be deployed effortlessly within the Cloud Continuum. Indeed, the methodology we introduced for constrained devices seamlessly extends to cloud solutions by simply plugging in containers suitable for cloud architectures without changing the underlying architecture. Our assessment involves an extensive measurement campaign encompassing more than 250 indoor and outdoor experiments, featuring wireless technologies, such as the IEEE 802.11n and 802.11ac Wi-Fi protocols. In our tests, Furcifer achieves an impressive **2x** reduction in energy consumption and an additional **30%** mean Average Precision score compared to LC, while also providing a remarkable **threefold** frame per second rate increase compared to EC. The main novel contributions of the paper are as follows:

- Furcifer, an innovative middleware that enables real-time monitoring of cloud continuum resources;
- A low-overhead container-enabled adaptation which allows the switch to a different model with minimal latency and negligible additional bandwidth usage;
- The first SC encoder-decoder model competitive against highly optimized state-of-the-art OD models used in practical applications;
- A dataset (which we pledge to release) consisting of more than 250 indoor and outdoor experiments, encompassing diverse scenarios with mobile devices operating under a broad range of channel conditions, wireless technologies and system load levels;
- A low complexity policy management module, trained on the dataset and experiments above, capable of optimizing the computing configurations, based on target power consumption, OD performance, and overall latency.

II. Related work

Furcifer builds on top of prior work in (a) Edge Computing context adaptation, (b) Image Compression and (c) Energy Consumption on Embedded Systems.

A. Real-World Context Adaptation in Edge Computing

Dynamic offloading of computational load to edge servers is essential to support resource-intensive applications on constrained mobile devices while meeting demanding QoS requirements [5]. Despite substantial advances in self-adaptive offloading strategies, most state-of-the-art solutions have mainly tackled this optimization problem from a theoretical perspective [6], [7]. Recent studies have investigated the benefits of context-aware adaptation for specific tasks such as 4k mobile Augmented Reality (AR) [8] and mobile video streaming [9] providing an in depth overview of the set of optimization operations required to effectively deploy self-adaptive policy managers in real-world field experiments. Progressive adaptation to different OD contexts using synthetic data [10] or uncertainty-aware domain adaptation networks [11] proved to be a promising direction when implementing an ML-based solution in real-world scenarios. Furthermore, employing strategies such as the random exploration of optimal scaling factors [12] can help alleviate the negative effects of source domain bias. However, these approaches often fail to consider critical system metrics such as energy consumption and network occupancy. In stark contrast with the current state of the art, Furcifer takes a comprehensive approach by tackling both system related and Computer Vision challenges, demonstrating the practical deployability of a low-complexity policy managers in constrained scenarios.

B. Image compression and Object Detection

Many neural models for computer vision, and OD in particular, are commonly trained and evaluated using state-of-the-art datasets such as COCO2017 [13] and Pascal VOC [14]. However, these evaluations often overlook the significant performance degradation caused by image compression [15], [16], which is inevitable in practical EC systems. In particular, widespread image compression techniques are designed for human perception rather than for image analysis. As a consequence, high performance requires the transfer of large volumes of data over capacity-constrained channels. To address this issue, SC (also known as supervised compression in some contexts) [17], [18] has recently emerged as a promising alternative to achieve state-of-the-art performance in computer vision tasks while effectively reducing bandwidth usage. The idea is to incorporate encoder/decoder-like structures within the ML models themselves, and use specialized training techniques to train task-oriented compressed representations [19], [20]. Knowledge Distillation [21], [22] is one of the tools used to maximize the effectiveness of SC frameworks.

Although the academic literature has demonstrated the potential of SC in popular datasets, an evaluation considering a complete computer vision pipeline is missing. Our work addresses this gap by illustrating how - often overlooked -

components such as preprocessing, acquisition, and timing characteristics significantly influence the overall performance of a computing configuration. Additionally, we emphasize the importance of proper model optimization, noting that SC is frequently compared to non-quantized models rarely used in practical deployments. Our work aims to evaluate the resilience of SC frameworks to quantization and to evaluate their performance compared to optimally designed models for embedded computers.

C. Energy Consumption on Embedded Systems

While much of the current ML-centered research is primarily focused on achieving the best task performance in the absence of resource restrictions, it is imperative to acknowledge energy consumption as a pivotal metric when considering mobile deployments. In recent years, initiatives such as The Low Power Image Recognition Challenge (LPIRC) [23] and a burgeoning energy-conscious perspective [24], [25] have emerged, underscoring a deliberate shift towards evaluating energy consumption. This evolving approach aligns with a sustainable trajectory aimed at achieving Green AI [26], standing in stark contrast to the opposite trend of Red AI.

In the domain of real-time computer vision, energy consumption is not solely determined by the number of Floating Point Operations (FLOPs) or Multiply-Accumulate (MAC) operations indicative of the model's complexity. Indeed, energy consumption is also proportional to the number of frame per seconds (FPS) processed by the system [27], [28]. Furthermore, an increase in image resolution results in a significantly expanded tensor space representation within the hardware accelerator. This expansion necessitates the activation of a larger portion of the hardware board to harness the advantages of parallelized convolutional operations [29]. Although in-depth studies address energy optimization from an embedded system perspective [30]–[32], current state of the art falls short of evaluating this aspect from a holistic cloud continuum perspective. Furcifer aims to strike a balance between resource efficiency and predictive precision, spanning from the edge to the cloud, and catering to the comprehensive energy optimization needs of modern mobile computing. By minimizing the energy consumption of mobile devices based on the desired mean Average Precision (mAP) score FPS rate, our solution represents a leap forward in realizing practical ubiquitous computer vision applications.

III. Problem Statement and Preliminaries

First, we discuss the application setting we target in this work, where a cluster of mobile devices (MD) and an edge server (ES) collaboratively perform OD on the streams of images generated by the MDs. In the following, we list the main computing strategies to achieve an optimal operating point in terms of energy consumption, OD performance, and frame rate.

- **Edge Computing:** The execution of the task on the edge server allows the use of high-performance models (e.g., large non-quantized models). However, the limited computing capabilities of ESs compared to cloud servers means that the server may struggle to serve a large number of task streams. Moreover, the need to transfer the input data to ES means that robust and high-capacity wireless channels are needed.
- **Local Computing:** In settings where the task complexity is low enough to match the capabilities of the mobile device, then local execution of the algorithm is a viable option. A trade-off is struck between task performance, power consumption and frame rate. Importantly, LC performance is not dependent on the state of the wireless channel connecting the MD to the ES, or the network and server load. In this context, quantization assumes a pivotal role in reducing execution time and energy consumption and enabling the use of better performing models whose use would otherwise be impractical on resource-constrained devices with limited computational capabilities. However, LC implies high energy usage, which leads to a reduced battery lifespan.
- **Split (collaborative) Computing - SC:** In SC, a subset of operations that would be executed on the ES is allocated to the mobile device. This subset often includes pre-processing operations, such as JPEG encoding and partial model inference altered to embed neural supervised encoding [33]. The objective is to decrease the amount of data to be transported over the wireless channel while minimizing the involvement of MD and possibly also decreasing the server load. This computing modality proves advantageous in settings where the communication channel's reliability is compromised, bandwidth demands exceed channel capacity or computing demands exceed server capacity. SC is specifically designed to address this scenario by mitigating both channel usage and computation burden on the ES.

Computing strategies comparison: The most popular metric used to evaluate OD is mean Average Precision (mAP), which combines precision and recall values based on Intersection over Union (IoU) scores across various levels of confidence thresholds. Typically, mAP scores are obtained by testing the algorithm on benchmark datasets such as COCO2017. However, when deploying an OD engine in a real-world setting, various factors such as camera resolution or scaling factor alterations come into play to determine the performance perceived by the application. Additional factors such as model quantization and image compression also play a significant role. With these in mind, we conduct an extensive evaluation of EC and LC, as well as Furcifer's SC engine. As MD, we select a Jetson Nano Dev Kit device, connected to a Jetson AGX Orin Dev Kit that acts as ES. At the ES, we deploy a modified version of Faster R-CNN [34] with Res50 backbone as a feature extractor testing various JPEG compression rates: 0%, 50%, and 70%. We also explore high-frame-rate alternatives for LC. Specifically, we select a quantized FP16 version of YOLOv5 [35] and SSD300, a customized adaptation of the Single Shot MultiBox Detector (SSD) [36] developed by NVIDIA.

In our exploration of SC, we develop a specialized encoder-decoder architecture trained using supervised compression and

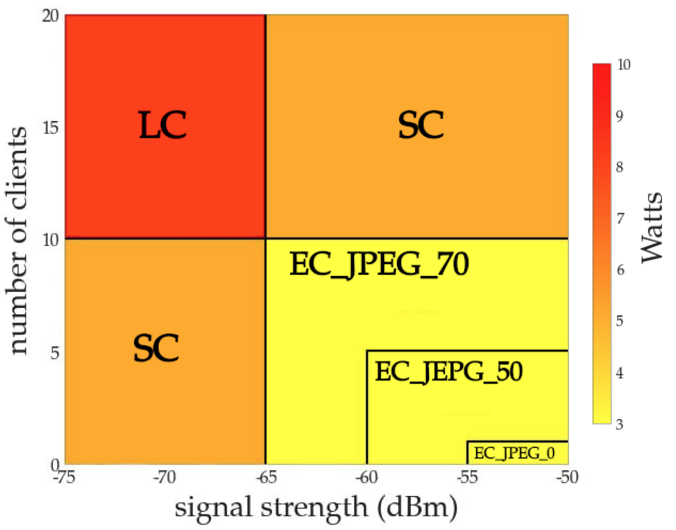

Fig. 2. Best performing computing modality and associated MD's power consumption as a function of signal strength and number of connected users.

OD Computing Configuration	mAP	FPS± std	FPS$_{min}$	FPS$_{max}$
EC_JPEG_0	37.051	3.29 ± 1.264	1.2	5.52
EC_JPEG_50	31.797	6.46 ± 3.007	1.89	10.01
EC_JPEG_70	29.476	6.66 ± 3.058	2.01	10.32
SC_FURCIFER	25.964	8.38 ± 1.992	4.96	11.72
LC_YOLOv5(FP_{16})	23.403	13.46 ± 0.261	11.89	13.89
LC_SSD$_{300}$(FP_{16})	23.201	28.45 ± 0.635	27.83	33.12

TABLE I
MEAN AVERAGE PRECISION AND FPS STATISTICS DEPENDING ON OD COMPUTING CONFIGURATION

Faster R-CNN as a teacher model. Our design is based on the model proposed in [37]. However, we optimized the original model by quantizing the encoder to FP16 and running it with an optimized inference engine. In addition, we fine-tuned the student model in order to match the camera resolution with the feature extractor upscaling factor. Further details about the distillation process are in Section IV-E.

The data reported in Figure 2 and Table I provides a comparison of the frame per second (FPS) and mAP obtained by each computing modality and model, whereas the figure shows the computing modality achieving the best FPS rate and the associated power consumption (MD only) as a function of signal strength (MD to ES channel) and the number of connected users. The results show that the best mAP performance is obtained using EC without JPEG compression - that is, the largest model running without image compression. Conversely, the maximum frame rate is achieved by a quantized version of the original model deployed on the MD. The figure illustrates how the optimal configuration is a function of the state of the channel and how different options result in a different amount of power spent by the mobile device. It should be pointed that this refers to the power consumed by MD and it does not take into account the overall total energy consumed by the whole system. As demonstrated later, Furcifer outperforms EC in terms of speed by achieving up to twice the FPS rate, all while achieving a higher mAP score in low-channel quality scenarios compared to the LC models.

IV. FURCIFER: SYSTEM DESIGN AND IMPLEMENTATION

The quantitative indicators presented in the previous section emphasize how there is no absolute winner among EC, SC, and LC even when considering a specific task, computing platforms, and communication technology. Instead, the – time varying – state of the system, which is influenced by mobility and load dynamics, determines the best computing configuration. However, changing the computing modality in real-world deployments is technically non-trivial. Furcifer realizes an adaptation engine composed of highly effective containerized models whose activation is determined by a control module informed by comprehensive system monitoring. While every element within the system holds a crucial role to enable adaptation to context, the container-based Service-Oriented Architecture (SOA) nature of Furcifer enables the independent deployment of each component. Furcifer tackles this obstacle by encapsulating each computing strategy within a container. Such containerization bundles the code and its dependencies, ensuring the ML application runs swiftly and reliably across diverse computing environments and facilitating a seamless transition between running ML models based on specific context characteristics. We have implemented Furcifer approach to Object Detection (OD) models as an ML task that is representative of real-time applications. Within this section, we provide an in-depth discussion of the main features of each component while graphically representing the overall architecture and component unit in Figures 4 and 3.

A. Energon: a Transparent Energy Monitoring Module

We first describe the system monitoring module - a mandatory component as it informs Furcifer's decision-making process about the current state of the system. Energon is developed from scratch in order to be compliant with Prometeus [38], a well-established real time metric standard. Energon collects system metrics from connected MDs by unobtrusively extracting data from the registries of the targeted operating system with minimal overhead to the monitored device and without the need to implement modifications to the application's underlying logic.

Energon focuses primarily on energy consumption and resource utilization in MDs, while also providing insights into additional metrics, including network quality, packet transmission and drop rates, CPU usage for individual cores, storage utilization, GPU usage percentage, and temperature measurements from various regions of the board. Scraped metrics are made available through an HTTP endpoint that can be queried on demand by the orchestrator.

B. From the Cloud the Edge: On-Demand Image Pulling

Before Furcifer, the adoption of containerized models on mobile devices was limited by the lack of hardware acceleration support and excessive computational power required. Our middleware integrates GPU-enabled capabilities offered by the original Docker runtime in a lightweight version of the renowned container framework where unused modules were

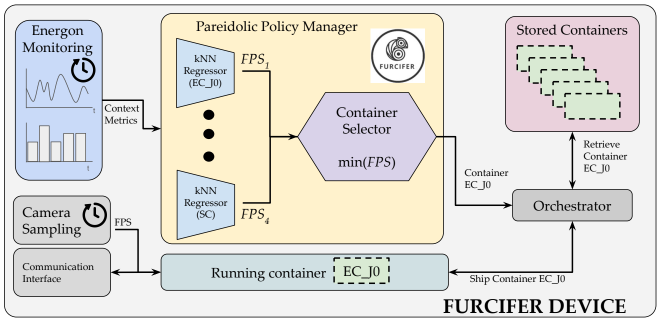

Fig. 3. Furcifer: Components on MD.

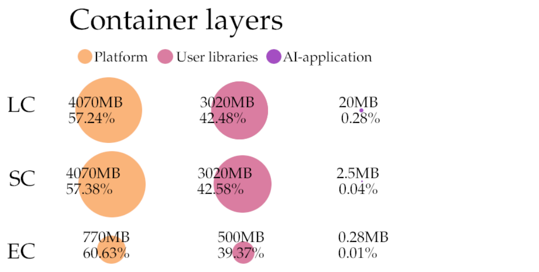

Fig. 5. Furcifer's GPU enabled containers footprint for LC, SC and EC OD computing configurations.

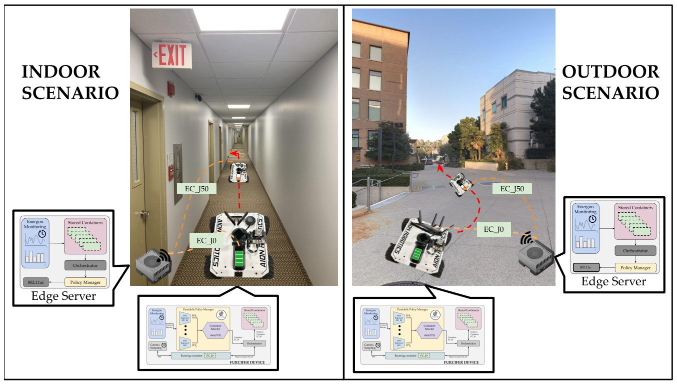

Fig. 4. Furcifer: Architecture and Settings.

removed. Furcifer provides a specialized container registry specifically designed for image compression and storage. This registry stores well-tailored images optimized for each compatible device, which are cached for future use based on the specific task the device is assigned. For each device type, a subset of images shares identical interfaces with the operating system hypervisor. However, these images differ at the application layer and user library level, adapting to the specific task to be executed and the corresponding dependencies that are necessary.

This shift of paradigm from the cloud to the edge empowers proactive mechanisms that enable seamless adjustments in response to evolving context requirements. We choose to apply containerization as a practical way to guarantee flexibility and fast reactiveness of the framework to future environment states. Our evaluation of the size of resulting container images reveals that less than 1% of the image comprises application-level files. This efficient design enables the download of only the last layer of the image, which has the same footprint size of the model itself. This approach minimizes network usage, since the model would anyway need to be transferred no matter whether a containerized approach is used or not.

Figure 5 shows the memory footprint of Furcifer container images in the LC, SC and EC configurations. Notably, the memory requirements for hardware acceleration dependencies at both the platform and user library levels are about four times more extensive for LC and SC compared to EC. Remarkably, the contribution of the application layer remains minimal compared to the other components, reinforcing the viability of switching between containers on demand. This is achieved by pulling only the necessary components, specifically the last layer of the container image, thereby reducing unnecessary network usage and resource consumption.

C. Communication Interface and Protocol

Deployed containerized models corresponds to a distinct and uniquely identified end point on the MD. Those microservices interact with the central orchestrator through a REST API, facilitating seamless communication and interaction while capitalizing on the advantages of minimal communication overhead. It continuously monitors potential new connections and, in the event of a connection loss, takes informed countermeasures to address the situation by switching to a local computing strategy. Furcifer's REST APIs operate on a request-response TCP-based model, enabling the framework to discern round-trip packet latencies. This functionality allows the introduction of well-defined rules for filtering out requests characterized by excessive communication delays. This mechanism ensures that the framework remains responsive and efficient, even in scenarios where the network conditions might fluctuate. Exchanged messages are defined as follows:

- **keep_alive**: This message is sent periodically using a polling mechanism to ascertain the presence of mobile devices within the same network.
- **start/stop_OD**: This message instructs the Mobile Device MD to initiate or terminate an Object Detection task. When initiating a task, the message also specifies the preferred computing strategy among LC, SC, and EC.
- **set_target_frame_rate**: This command sets the desired FPS rate for camera sampling based on dynamic requirements defined at the application level. Recognizing the direct correlation between higher FPS rates and increased power consumption, Furcifer intelligently conserves energy and network resources when higher frequency camera sampling is unnecessary, e.g., Vehicle-to-Vehicle (V2V) cameras in low-traffic environments [39]).
- **set_compression_rate**: If an EC configuration is used, the MD can opt to compress captured images before transmitting them to the ES for final detection. This message specifies the desired compression rate, controlling the balance between reduced compression for improved mAP score.

D. Orchestrator

An orchestrator constitutes a software mechanism designed to streamline the deployment, management, scaling, and networking of containers in a computing environment. The Furcifer's orchestrator is a customized open-source version of the Docker container's platform. Its design is tailored to enable the deployment of a singular container per offloading strategy across any MD connected with the ES. This strategic deployment facilitates the isolation of containerized machine learning model implementations.

Furthermore, the orchestrator's scaling and management protocols are controlled by the pareidolia policy. This policy evaluates the energy consumption metrics of the MD, as well as an array of context-sensitive metrics. This evaluation enables the orchestrator to dynamically adjust its operational strategy, seamlessly transitioning between two or more containers to optimize performance and resource utilization.

E. Furcifer's SC Engine

Furcifer introduces a new SC engine tailored for resource-constrained devices, marking a significant advancement in this real-world domain. Leveraging Faster R-CNN as the teacher model, we use a modified version of the knowledge distillation process adopted in SC2 Benchmark [37] to design a compact encoder optimized for constrained devices. This encoder serves a dual purpose: minimizing channel occupancy and effectively distributing computation load between mobile devices and the edge server. Differently from the original model described in [37] we optimized each tensor operation to exploit the parallel execution on the GPU. We transform the canonical matrix operations into cuDNN tensor operations [40] and run the model by building a TensorRT engine [40] for high performance inference optimization. In the fine-tuning process, we employ preprocessing image upscaling transformation of 400 pixels, which is smaller than that of the original model, with a minimum height of 800 pixels. This resolution adjustment is implemented to facilitate quicker inference on resource-constrained devices.

The optimized encoder heavily relies on quantization and channel compression to reduce execution time as much as possible. To enhance data compression, we strategically place a one-channel bottleneck in the initial layers of the feature extraction segment of the network. This choice leads to further data reduction, increasing the efficiency of the whole process. Additionally, we incorporate INT8 quantization at the end of the encoder. This quantization approach optimizes the representation of the data, contributing to both improved data compression and streamlined computation. The dynamic nature of the system is upheld by calculating the scaling factor and zero point on a per-image basis as they are processed. These values are then communicated to the decoder located at the ES, along with the resulting INT8 tensor from the encoder inference process.

The incorporation of INT8 quantization and dequantization for the inference tensor on the MD minimizes the influence of weight quantization within the encoder engine, which transitions from FP32 to FP16. We confirm the negligible impact on mAP score testing Furcifer's SC engine on the COCO2017 dataset, obtaining 25.966 and 25.964 as mAP scores for FP32 and FP16, respectively. Thus, the adoption of an FP16 quantized encoder on the mobile device delivers a nearly twofold increase in processing speed compared to its FP32 counterpart, while preserving about the same mAP score. This finding underscores the additional advantage of applying quantization to SC encoders, which, unlike their LC counterparts, are already quantizing the final encoding result to minimize channel occupancy. As a result, they are less susceptible to mAP score reduction due to quantization. We remark how this optimization makes SC a competitive option against optimized models for embedded devices, as demonstrated by our results.

F. Camera Sampling Module

The Camera Sampling Module (CSM) performs the capture of frames from accessible cameras, integrating essential drivers to ensure optimal performance with adaptability to various camera models. Additionally, this module offers to the user the ability to set precise directives for the desired camera sampling rate and image resolution [41]. Such dynamic adjustments align with distinct embedded OD models stored within the container registry located on the ES. This synergy between the Camera Sampling Module and the containerized models underscores Furcifer's ability to match detection demands with resource availability, enabling seamless contextual adaptation and optimized performance in a wide range of different use cases. Furthermore, this module assumes responsibility for image compression prior to transmission to the ES when EC is selected as computing configuration.

G. Pareidolia: Low-Complexity Similarity-Based Context Adaptation

Pareidolia, a concept rooted in human perception, reflects the inclination to perceive distinct, often meaningful shapes or images within random or ambiguous visual patterns. It manifests as a natural cognitive process, wherein the brain attempts to link novel ideas with existing concepts. Leveraging already solved tasks, Furcifer leverages "pareidolia" as a context adaptation approach. Each participating MD maintains a record of previously completed tasks. This historical context empowers the node to discern which computing strategy aligns best with the current system state by identifying analogous past scenarios. When a sufficiently similar context is detected, the ES intervention may not be required. Conversely, if an analogous context is not found, pertinent task details are shared with the ES to collaboratively determine the optimal model and computing configuration (EC, LC or SC) that best matches the current system state. The Pareidolic Policy Manager (PPM), as defined, facilitates low-complexity trend forecasting [42]–[44]. It has demonstrated its effectiveness across a diverse range of real-world scenarios, operating seamlessly without simplifications while reducing the minimal additional burden on already limited computational capabilities. As a result of the interaction between the increasing number of concurrent clients and the variability of network conditions, PPM forecasts the expected number of FPS that the mobile device will achieve when employing each considered computing configuration. This predictive functionality enables PPM to anticipate the impact of each strategy choice on FPS and tailor the decision accordingly. PPM forecasting capability comes from a set of predictors which based on the current metrics collected by Energon are capable of determine the resulting FPS rate the framework will achieve as a consequence of choosing a specific computing configuration.

V. Experimental Evaluation

In this section, we present and report the result of the experiments carried out using the Furcifer framework. These experiments report relevant performance metrics on a broad range of states and settings of the targeted deployment environment, including outdoor and indoor ones covered with IEEE 802.11n and 802.11ac connectivity. Through this extensive set of experiments, we aim to assess the ability of Furcifer to dynamically adapt the cloud continuum configuration against system state dynamics. The dataset we collect comprises over 250 distinct combinations of channel conditions (expressed as signal strength) and number of concurrent client connections. Although indoor and outdoor experiments exhibit similar trends, indoor scenarios tend to be characterized by a higher degree of unpredictability, which is primarily due to the presence of obstacles that complicate signal propagation. As a result, the overall channel quality is adversely affected, leading to more variable and less consistent performance outcomes.

A. IEEE 802.11n Experiments

First, we focus on the widely used Wi-Fi 801.11n standard. Our experimental setup features a Jetson Nano DevKit as the mobile device equipped with a 640x480px USB webcam. In indoor scenarios, we orchestrated the movement of the device along a designated path spanning approximately 20 meters. In outdoor scenarios, the path extends over a distance of 50 meters. This deliberate variation allowed us to replicate a spectrum of signal strengths and network dynamics, capturing the intricacies of both indoor environments and outdoor settings. For each experimental run, we meticulously examine the system scalability across different user scenarios. Specifically, we investigate the performance of the system, as the number of clients varies between 1 and 20.

Figures 6a and 6b depict the FPS and error percentage metrics for SC and EC with JPEG compression gain 0, 50 and 70% as a function of distance with a single connected user. Within the indoor experiment settings (Figure 6a), there is a striking similarity in the average FPS achieved by EC and SC, except in the scenario without image compression. Conversely, in the outdoor environment (Figure 6b), SC takes advantage of the improved channel conditions compared to the indoor setting and achieves up to 2 additional FPS. Importantly, it is worth noting that SC not only excels in FPS but also achieves a higher mAP score compared to LC, showcasing its suitability in terms of both task performance and frame processing speed.

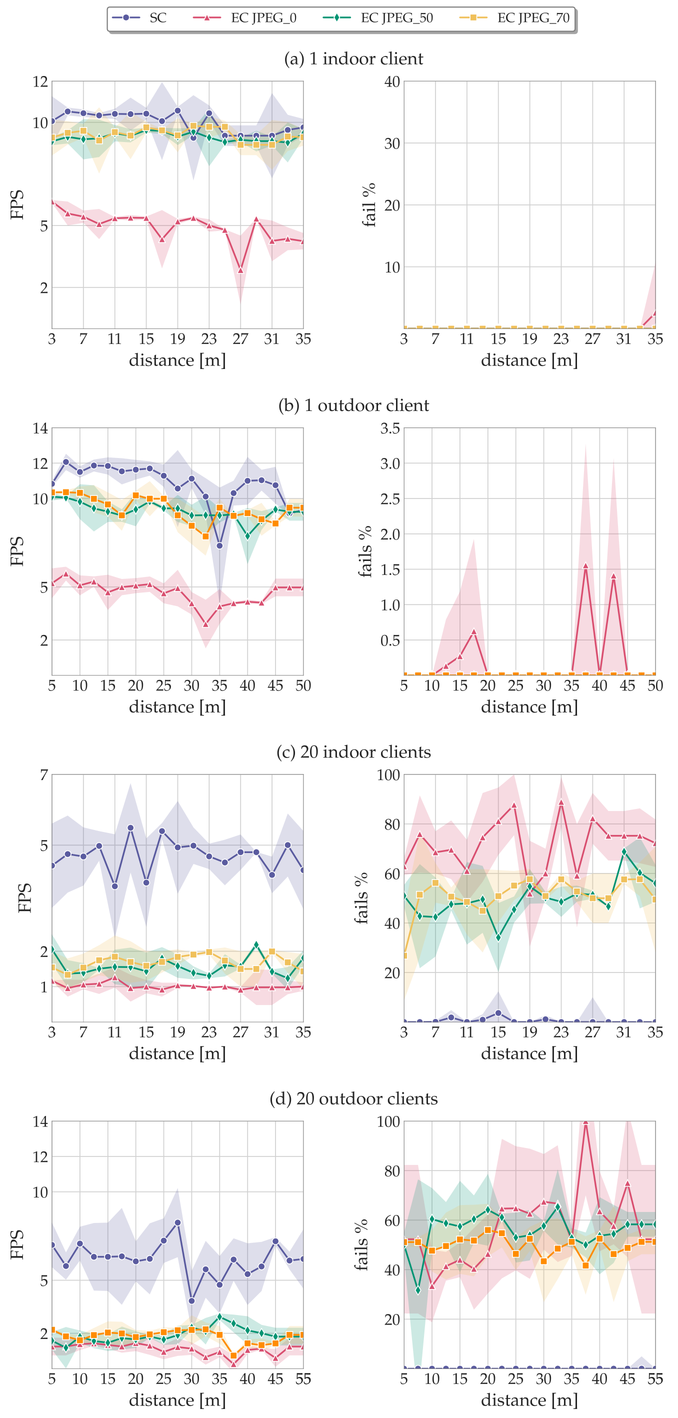

Fig. 6. IEEE 802.11n experiments: with one connected client (a) indoors and (b) outdoors, and twenty connected clients (c) indoors and (d) outdoors. Performance metrics - FPS, fail percentage - as a function of distance for SC and EC with 0%, 50%, and 70% JPEG compression gains. The lines in the plots correspond to average metrics, while the shading represents variance.

Figures 6c and 6d show performance metrics as the number of clients connected to ES varies. Several notable trends emerge from this analysis. First, we observe an interesting pattern regarding the impact of image compression techniques as the number of concurrent clients increases. The failure rates associated with JPEG are dramatically larger compared

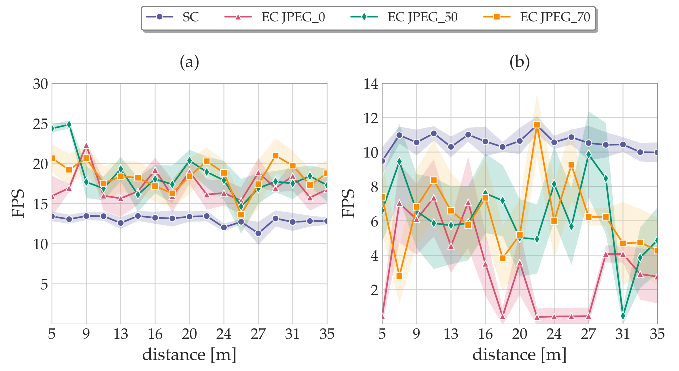

Fig. 7. IEEE 802.11ac FPS rate metric: ten connected client (a) and twenty connected client (b) - as a function of distance for SC and EC with JPEG compression gain 0, 50 and 70%. . The lines in the plots correspond to average metrics, while the shade is the variance.

to SC, reaching up to 100% failure rate in some measurements. This result underscores the perils of relying solely on image compression when dealing with a larger number of clients, highlighting the potential limitations of this approach in dynamic and demanding network/server conditions. On the contrary, SC achieves almost a steady 0% failure rate due to the small amount of network and server resources used by this configuration. The resulting improved resilience of Furcifer underlines the effectiveness of its context-aware approach, which enables SC to consistently outperform EC in the range of tested network conditions.

B. IEEE 802.11ac Experiments

The overhead in a wireless communication channel fluctuates based on the technology employed, thereby influencing the trade-off between computing and wireless communication inherent in the SC and EC conditions. We extend the evaluation of Furcifer performance to include IEEE 802.11ac. In these experiments, we replicated the same path for the MD, while concurrently running increasing parallel connections of up to thirty clients. We use a Wi-Fi network interface that supports the IEEE 802.11ac 5GHz protocol and establish a connection between the ES and MD over an 80 MHz band. Our Wi-Fi 5Ghz antennas also support MU-MIMO technology which additionally improves the overall communication performance.

Figure 7 shows FPS rate as a function of the number of concurrent clients, ranging from 10 to 30. All EC compression strategies benefit from improved connection capabilities, outperforming SC when the number of clients is smaller than 10. Instead, when the number of clients increases to 20 and 30, the additional load on ES penalizes EC over SC of about 40% in terms of the average FPS successfully processed over time. As shown in the analysis that follows, the superior performance compared to ES when the system is under pressure is due both to the decrease in channel usage and the effort of the server granted by SC. On the contrary, when the full capacity of the network and server are available, SC is penalized by the computing effort allocated to MD.

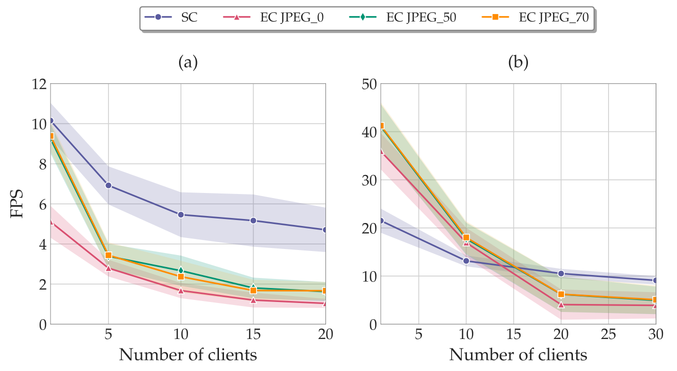

Fig. 8. Comparison between the 802.11n (a) and 802.11ac (b) Wi-Fi protocols, depicting the FPS_fails score with an increasing number of concurrent clients.

To better appreciate the impact of communications on the computing mode, we conduct a comparative analysis between the IEEE 802.11n and 802.11ac Wi-Fi protocols. The results shown in Figure 8 depict the FPS achieved as the number of connected clients increases. We note how the improved data rates offered by the IEEE 802.11ac means that EC is the winning solution up to a certain load level, whereas in IEEE 802.11n experiments, the superior compression granted by SC results in the latter being the winning solution. However, this comparison is made with the same signal strength variability for EC and SC, moving the MD along the same spatial trajectory.

C. PPM: Pareidolic Policy Management

In the previous sections, we demonstrated the need for the dynamic adaptation of the computing configuration. We now evaluate the ability of Furcifer - and specifically its policy management module - to provide adaptation capabilities without imposing a significant overhead. In terms of energy consumption and optimal task performance, computing configurations can be clearly ranked based on the MD perspective. In fact, EC does not impose any computing load to the MD and achieves the best mAP thanks to the use of larger models. SC allocates minimal computing effort to the MD and has the second best mAP. Finally, LC results in the largest energy intake and worst performance. Thus, the decision engine has to evaluate the ability of the individual computing configurations to achieve the desired FPS rate given the currently perceived system state, and then select them in the order dictated by energy and mAP. To support such decision process, we then build simple KNN regressors that take as input application context metrics sush as: the inference time on the ES, the average round trip time, the communication channel quality, and the current resource usage. This produces as output the predicted FPS rate for each considered computing configuration, allowing PPM to anticipate the resulting Quality of Service (QoS) of a particular action before executing it. We evaluate the regressors when training and applying them in specific environments (e.g., indoor or outdoor), as well as on their ability to generalize. The resulting loss metrics are reported in Table II. It can be observed that when these regressors are trained in the same context where they are applied,

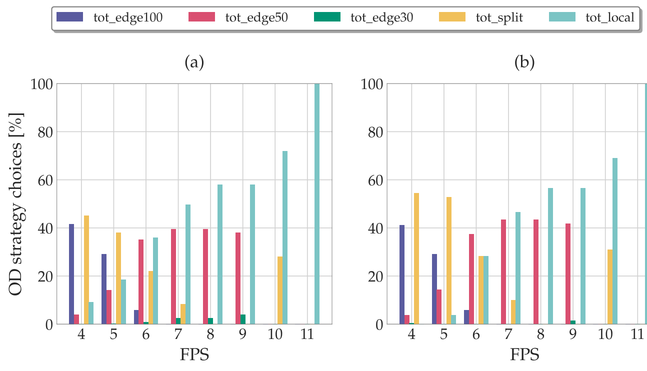

Fig. 9. Choices distribution of considered computing configurations depending on target FPS rate for the Ground truth oracle (a) and our Pareidolic Policy manager (b)

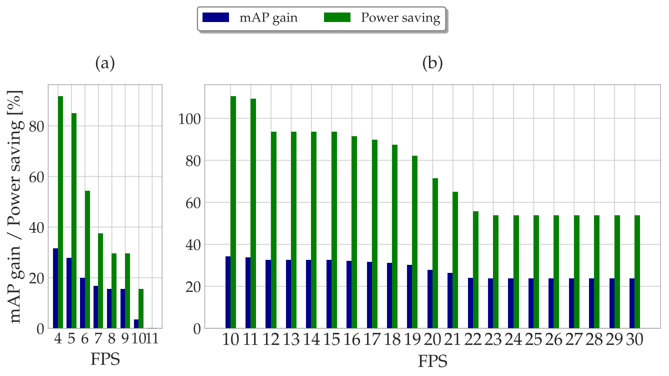

Fig. 10. mAP gain and Energy saving with IEEE (a) 802.11n and (b) 802.11ac Wi-Fi protocols

TABLE II
LOW COMPLEXITY FRAME RATE PREDICTORS

	indoor		outdoor		indoor $\rightarrow$ *outdoor*	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
EC_J0	0.27	7.30	0.37	10.69	0.58	20.37
EC_J50	0.50	7.86	0.59	9.25	0.71	15.37
EC_J70	0.47	7.48	0.53	7.68	0.81	15.98
SC	0.70	7.46	0.97	9.48	1.23	15.63

TABLE III
RMSE BETWEEN PERCENTAGE OF OD COMPUTING CONFIGURATIONS

target_FPS	PPM	DRL
4	7.848	87.210
5	2.287	104.979
6	21.875	123.547
7	32.552	101.824
8	36.892	75.419
9	7.465	78.375
10	24.045	69.928
11	58.593	71.310
$\overline{RMSE}$	23.944	89.074

the error is minimal (below 11% $MAPE$ – Mean Absolute Percentage Error) across all computing configurations. In the case of transfer learning, where models trained indoors are deployed outdoors, the maximum loss value $MAPE$ increases to 20%.

Adaptation Results: We compare Furcifer performance with a static LC solution, which represents the only viable option when the connection quality or system load cannot support the desired FPS rate. Figure 9 shows the distribution of decisions made by Furcifer policy manager across the spectrum of available cloud continuum strategies. It is important to note that our policy manager, despite its low complexity, demonstrates the ability to match the configuration that an oracle controller would implement. Table III reports the Root Mean Squared Error (RMSE) between the percentage of choices made by Furcifer PPM low complexity policy manager and a baseline DRL agent. The striking alignment between the decisions made by the Furcifer pareidolic policy manager and the ground truth highlights the feasibility of deploying a low complexity predictor deployed at MD, where more complex controllers may fail to train properly or adequately generalize.

In the IEEE 802.11n configuration, Furcifer reduces the energy intake by approximately 80% while achieving an average mAP score increase of over 20% in comparison to LC. The relevance of these outcomes is further amplified when using the IEEE 802.11ac protocol. In this scenario, energy savings exceed 100%, and mAP consistently maintains a level above 20% for all defined targets FPS. As Figure 10 shows, Furcifer can easily generalize, accurately predicting the frame per second even when trained on an indoor environment and then tested outdoor.

VI. CONCLUSIONS AND FUTURE WORK

This paper presented Furcifer, an innovative framework designed to provide seamless adaptation of the cloud continuum computing configuration in dynamic mobile settings. Our approach based on containers and simple predictors result in an extremely low-complexity and low-overhead solution, which we prove effective in a wide set of deployment environments, system states, and wireless settings. In addition to EC and LC, we have developed a highly optimized neural model performing supervised compression, by showing that it represents an extremely effective third computing configuration. In our tests, Furcifer achieves remarkable results, demonstrating a **2x** reduction in energy consumption and a **30%** additional mAP score gain compared to LC. Additionally, it delivers an impressive **three-fold** increase in the FPS rate compared to EC. These achievements underscore the considerable impact of Furcifer's dynamic adaptation engine in enhancing both energy efficiency and performance. In the future, we intend to explore other computationally demanding tasks, such as Semantic Segmentation and LiDAR-based 3D mapping, to practically show the applicability of the Furcifer framework to a wider spectrum of real-world use cases. Moreover, our road map includes the release of an open-source version of Furcifer, along with an in-depth exploration of the benefits of our policy-based adaptation on a diverse range of mobile devices.

ACKNOWLEDGEMENT

This work was partially supported by the National Science Foundation under grants CCF 2140154 and CNS 2134567.

References

[1] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," *ACM Comput. Surv.*, 2022.

[2] E. M. Grua, I. Malavolta, and P. Lago, "Self-adaptation in mobile apps: A systematic literature study," in *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2019.

[3] B. Jinsung and P. Sehyun, "Development of a self-adapting intelligent system for building energy saving and context-aware smart services," *IEEE Transactions on Consumer Electronics*,

[4] G. Orsini, D. Bade, and W. Lamersdorf, "CloudAware: Empowering context-aware self-adaptation for mobile applications," *Transactions on Emerging Telecommunications Technologies*, e3210, 2017.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, 2012.

[6] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Transactions on Vehicular Technology*, 2020.

[7] K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, S. Salloum, and M. S. Mahmud, "A review of optimization methods for computation offloading in edge computing networks," *Digital Communications and Networks*, 2023.

[8] J. Ren, L. Gao, X. Wang, *et al.*, "Adaptive computation offloading for mobile augmented reality," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, pp. 1–30, 2021.

[9] W. U. Rahman, C. S. Hong, and E.-N. Huh, "Edge computing assisted joint quality adaptation for mobile video streaming," *IEEE Access*, 2019.

[10] H.-K. Hsu, C.-H. Yao, Y.-H. Tsai, *et al.*, "Progressive domain adaptation for object detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[11] D. Guan, J. Huang, A. Xiao, S. Lu, and Y. Cao, "Uncertainty-aware unsupervised domain adaptation in object detection," *IEEE Transactions on Multimedia*, 2022.

[12] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi, "St3d: Self-training for unsupervised domain adaptation on 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2014.

[14] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*,

[15] L. Galteri, M. Bertini, L. Seidenari, and A. Del Bimbo, "Video compression for object detection algorithms," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018.

[16] H. Choi and I. V. Bajić, "Deep feature compression for collaborative object detection," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018.

[17] X. He, M. Ji, and H. Bao, "A unified active and semi-supervised learning framework for image compression," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[18] E. Nowara and D. McDuff, "Combating the impact of video compression on non-contact vital sign measurement using supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[19] D. Tellez, G. Litjens, J. van der Laak, and F. Ciompi, "Neural image compression for gigapixel histopathology image analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[20] D. Tellez, D. Höppener, C. Verhoef, *et al.*, "Extending unsupervised neural image compression with supervised multitask learning," in *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, ser. Proceedings of Machine Learning Research.

[21] Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh, "Distilled split deep neural networks for edge-assisted real-time systems," in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019.

[22] Y. Matsubara, D. Callegaro, S. Singh, M. Levorato, and F. Restuccia, "Bottlefit: Learning compressed representations in deep neural networks for effective and efficient split computing," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2022.

[23] K. Gauen, R. Rangan, A. Mohan, Y.-H. Lu, W. Liu, and A. C. Berg, "Low-power image recognition challenge," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2017.

[24] E. Cai, D.-C. Juan, D. Stamoulis, and D. Marculescu, "Neuralpower: Predict and deploy energy-efficient convolutional neural networks," in *Proceedings of the Ninth Asian Conference on Machine Learning*, M.-L. Zhang and Y.-K. Noh, Eds., 2017.

[25] J. Li, A. Louri, A. Karanth, and R. Bunescu, "Gcnax: A flexible and energy-efficient accelerator for graph convolutional neural networks," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021.

[26] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, 2020.

[27] N. Tijtgat, W. Van Ranst, T. Goedeme, B. Volckaert, and F. De Turck, "Embedded real-time object detection for a uav warning system," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017.

[28] A. Suleiman and V. Sze, "An energy-efficient hardware implementation of HOG-based object detection at 1080hd 60 fps with multi-scale support," *Journal of Signal Processing Systems*, 2015.

[29] M. Verucchi, G. Brilli, D. Sapienza, *et al.*, "A systematic assessment of embedded neural networks for object detection," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[30] J. Deng, Z. Shi, and C. Zhuo, "Energy-efficient real-time uav object detection on embedded platforms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[31] H. Mao, S. Yao, T. Tang, B. Li, J. Yao, and Y. Wang, "Towards real-time object detection on embedded systems," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[32] A. Omid-Zohoor, C. Young, D. Ta, and B. Murmann, "Toward always-on mobile object detection: Energy versus performance tradeoffs for embedded hog feature extraction," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[33] Y. Ji, H. Zhang, Z. Zhang, and M. Liu, "CNN-based encoder-decoder networks for salient object detection: A comprehensive review and recent advances," *Information Sciences*, 2021.

[34] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016.

[35] G. Jocher, A. Stoken, J. Borovec, *et al.*, *Ultralytics/yolov5: V7.0 - yolov5 sota realtime instance segmentation*, 2022.

[36] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, 2016.

[37] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, *Sc2 benchmark: Supervised compression for split computing*, 2023.

[38] N. Sukhija and E. Bautista, "Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus," in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019.

[39] I. Takai, T. Harada, M. Andoh, K. Yasutomi, K. Kagawa, and S. Kawahito, "Optical vehicle-to-vehicle communication system using led transmitter and camera receiver," *IEEE Photonics Journal*, 2014.

[40] S. Chetlur, C. Woolley, P. Vandermersch, *et al.*, *Cudnn: Efficient primitives for deep learning*, 2014.

[41] J.-Y. Wu, V. Subasharan, T. Tran, and A. Misra, "Mrim: Enabling mixed-resolution imaging for low-power pervasive vision tasks," in *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2022, pp. 44–53. DOI: 10.1109/PerCom53586.2022.9762398.

[42] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, 2016.

[43] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, 2012.

[44] H. Wang, W. Li, J. Sun, *et al.*, "Low-complexity and efficient dependent subtask offloading strategy in iot integrated with multi-access edge computing," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023. DOI: 10.1109/TNSM.2023.3295653.