Collaborative Inference in Resource-Constrained Edge Networks: Challenges and Opportunities

Nathan Ng
University of Massachusetts
kwanhong@cs.umass.edu

Abel Souza
University of California, Santa Cruz
absouza@ucsc.edu

Suhas Diggavi
University of California, Los Angeles
suhas@ee.ucla.edu

Niranjan Suri
Army Research Lab
niranjan.suri.civ@army.mil

Tarek Abdelzaher
University of Illinois
zaher@illinois.edu

Don Towsley
University of Massachusetts
towsley@cs.umass.edu

Prashant Shenoy
University of Massachusetts
shenoy@cs.umass.edu

***Abstract*—Many IoT applications have increasingly adopted machine learning (ML) techniques, such as classification and detection, to enhance automation and decision-making processes. With advances in hardware accelerators such as Nvidia's Jetson embedded GPUs, the computational capabilities of end devices, particularly for ML inference workloads, have significantly improved in recent years. These advances have opened opportunities for distributing computation across the edge network, enabling optimal resource utilization and reducing request latency. Previous research has demonstrated promising results in collaborative inference, where processing units in the edge network, such as end devices and edge servers, collaboratively execute an inference request to minimize latency.**

This paper explores approaches for implementing collaborative inference on a single model in resource-constrained edge networks, including on-device, device-edge, and edge-edge collaboration. We present preliminary results from proof-of-concept experiments to support each case. We discuss dynamic factors that can impact the performance of these inference execution strategies, such as network variability, thermal constraints, and workload fluctuations. Finally, we outline potential directions for future research.

***Index Terms*—Collaborative inference, edge computing**

I. Introduction

Machine learning (ML) techniques such as detection and classification have become essential in IoT sensor-to-decision pipelines. In edge networks, network components such as sensing nodes, edge servers, and the cloud possess varying processing capabilities. Traditionally, sensing nodes are resource-constrained compared to servers and often lack the processing power and energy needed for these ML techniques. To mitigate this performance disparity, sensing nodes can offload computation-intensive tasks to nearby edge servers equipped with more powerful hardware. This offloading strategy is illustrated in Figure 1.

With advances in hardware accelerators, new opportunities have emerged to further optimize computation placement within the edge network. Accelerators such as Intel's Movidius Vision Processing Unit (VPU) [1], Nvidia's Jetson Nano [2], TX2 [3], Orin Nano [4], Orin NX edge GPUs [5], and Google's Edge Tensor Processing Unit (TPU) [6] are designed to bridge the performance gap between resource-constrained sensing nodes and powerful servers. These lower-cost, energy-efficient accelerators are specifically designed to efficiently handle ML tasks such as computer vision and deep learning inference. As a result, it is now possible to equip sensing nodes with these accelerators and achieve performance comparable to that of general-purpose servers for specific applications. Since data transfers are often a source of high latency, processing data directly at the sensing nodes can potentially reduce the latency of IoT applications.

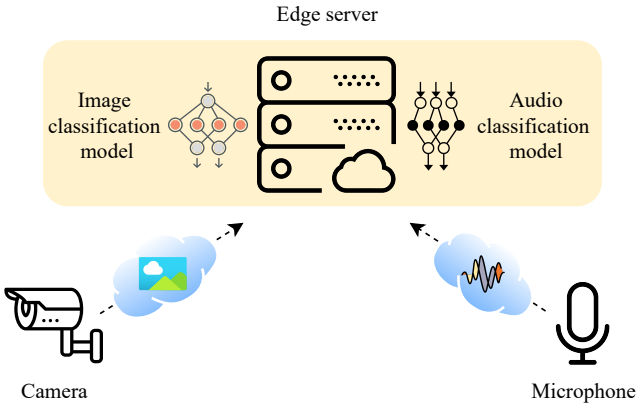

Fig. 1. Inference execution offloading in an edge network.

Specifically, recent research [7]–[16] has demonstrated the effectiveness of collaborative inference in edge networks, where multiple processing units or network components work together to execute inference requests. This collaboration can be implemented in two primary forms. First, a single network component can leverage its processing units to jointly perform inference, thereby optimizing the use of available resources. For instance, a sensing node equipped with an accelerator can distribute the computational load between its CPU and its accelerator to enhance performance. Second, multiple network components (e.g., a sensing node and an edge server) can collaboratively execute different portions of the same infer-

This research is supported by NSF grants 2211302, 2211888, 2213636, 2105494, 23091241, US Army contract W911NF-17-2-0196, and Adobe.

ence request, with the goal of minimizing network latency while leveraging the superior hardware available at the edge server. Each approach can be implemented using various combinations of hardware, presenting distinct trade-offs, with their applicability depending on the specific requirements of the task at hand.

Beyond static collaboration configurations, dynamic conditions in resource-constrained edge networks must also be carefully considered when designing collaborative inference applications. Specifically, external dynamics, such as attacks on sensing nodes, network disruptions, and DDoS attacks, are prevalent in the adversarial environments of IoBTs and may disrupt operations and degrade overall performance [17]. Additionally, internal dynamics, including thermal conditions and battery levels of sensing nodes, can also affect system performance. Therefore, collaborative inference applications must be designed to be robust and adaptive to address these challenges.

This paper explores collaborative inference strategies that execute inference on a single ML model and their various implementations. We categorize these strategies into two main classes: *intra-component collaboration*, which involves the collaboration of processing units within a single network component, and *inter-component collaboration*, which involves collaboration across multiple network components. For each class, we discuss potential implementation approaches and present preliminary results to demonstrate their benefits. Additionally, we examine the challenges of dynamics in resource-constrained networks that may impact system performance. In summary, we make the following contributions:

- We discuss different forms of single-model collaborative inference strategies and categorize them into two main classes (Section III).
- We present preliminary results to support the practicality of the strategies (Section IV).
- We discuss challenges associated with the dynamics in resource-constrained edge networks (Section V) and outline directions for future work (Section VI).

II. Background

This section presents background on edge-based IoT applications, commonly used hardware accelerators, and collaborative inference techniques.

A. Edge-based IoT Applications

In IoT applications, sensing nodes collect raw data that must be processed to extract meaningful insights. Machine learning (ML) inference is a key technique used in this process, which involves passing input data through a trained neural network to generate predictions. For example, a surveillance camera captures an image and requires image classification to identify objects within the frame. Since sensing nodes are often resource-constrained, they typically offload computationally intensive ML inference tasks to nearby edge servers, which may introduce high network latency. However, if a sensing node is equipped with a hardware accelerator, it can perform the computation locally and only transmit the results to the remote server, thereby reducing network latency and improving overall efficiency.

B. Hardware Accelerators

A wide range of hardware accelerators with small form factors have been developed to speed up machine learning inference workloads on the edge. Table I highlights the characteristics of several commonly deployed accelerators, spanning from low-end to high-end options. Specifically, the Intel Movidius Myriad X Vision Processing Unit (VPU) is designed to accelerate vision-based applications and CNN inference. Google Edge Tensor Processing Unit (TPU) optimizes ML inference with the Tensorflow-lite framework on the edge. Nvidia Jetson Nano, TX2, Orin Nano, and Orin NX are all tailored for AI workloads, delivering accelerated GPU performance for tasks such as deep learning and computer vision. These accelerators are engineered to operate within low power budgets, making them suitable for devices with constrained power availability, such as low-end Raspberry Pi-class nodes. All network components can deploy an accelerator to boost the processing capabilities. Not only can end devices (e.g., sensing nodes) leverage accelerators to achieve near-server-level performance in certain applications, but edge servers can also utilize these accelerators to optimize performance and manage increased workloads.

C. Collaborative Inference

Collaborative inference has recently gained significant attention in both academia [7]–[14], [16] and the industry [15]. It involves utilizing multiple processing units in the edge network to execute inference requests, and the collaboration can be realized in two forms: single-model collaboration and multi-model collaboration. In single-model collaboration, a model is partitioned into multiple segments, each processed by different network components. Conversely, multi-model collaboration utilizes inputs from various sensors to jointly complete an inference task, leveraging diverse data sources to enhance overall accuracy and performance. Additionally, collaboration can occur across different network topologies, such as a device interacting with multiple edge servers in a broadcast manner, peer-to-peer interactions between devices, or a client-server model between devices and edge servers. In this paper, we focus on single-model collaboration approaches that employ the client-server model. We focus on collaboration approaches that take the ML model as a black box and do not require modifying the neural network structure. Techniques such as integrating collaborative inference with model quantization [11], [13], [15] or employing multi-exit deep neural networks [10], [16] are beyond the scope of this paper.

III. Collaborative inference Approaches

Single-model collaborative inference strategies can be categorized into two main classes: *intra-component collaboration*, which involves the collaboration of processing units within

TABLE I
CHARACTERISTICS OF COMMON HARDWARE ACCELERATORS

Accelerator	Peak Power (W)	Memory	Cost	Target Workload
Intel Movidius Myriad X VPU	1 – 2.5	2.5 MB of SRAM	$59	Imaging, computer vision and CNN inference
Google Edge TPU	2	8 MB of SRAM	$75	ML Inference with TensorFlow Lite
Nvidia Jetson Nano	10	4 GB of LPDDR4	$99	AI and GPU workload
Nvidia Jetson TX2	15	8 GB of LPDDR4	$399	AI and GPU workload
Nvidia Jetson Orin Nano	15	8 GB of LPDDR5	$499	AI and GPU workload
Nvidia Jetson Orin NX	25	16 GB of LPDDR5	$899	AI and GPU workload

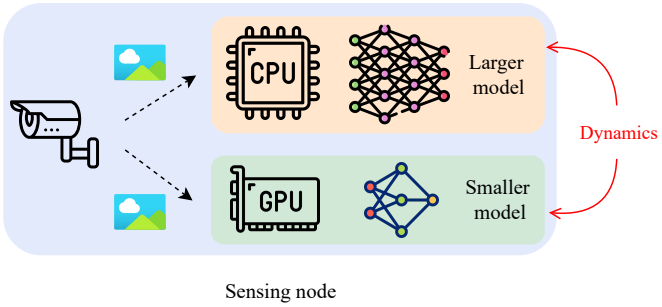

Fig. 2. Intra-component collaboration with GPU and CPU.

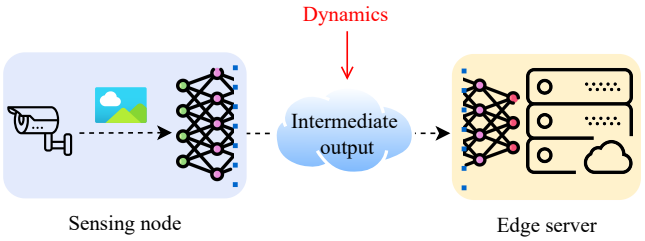

Fig. 3. Device-edge collaboration example.

the same network component, and *inter-component collaboration*, which spans across multiple network components. This section explores possible implementations for each class and discusses their associated benefits and tradeoffs.

A. Intra-component Collaboration

With the integration of hardware accelerators, sensing nodes can now have more than one processing unit. This creates opportunities for applications to utilize both processing units within the device to execute the same inference task, thereby maximizing resource utilization and reducing latency. For example, an NVIDIA Jetson TX2 can leverage both its GPU and CPU, while a Raspberry Pi 5 with an attached TPU can utilize both its TPU and CPU. In the following, we present an example of an implementation of intra-component collaboration.

Figure 2 illustrates an example of intra-component collaboration using both GPU and CPU on the sensing node, where two models that perform the same inference task but differ in computational requirements and precision are used. The quantized MobileNetV2 model, which runs on the GPU, is optimized for efficient execution, while the full MobileNetV2 model, running on the CPU, offers higher precision at the cost of increased computational demand. Given an input, the application performs inference on both models in parallel. If the classification confidence level from the quantized model exceeds a predefined threshold, the application returns the result and terminates the CPU processing. Conversely, if the confidence level is low, the GPU picks up the intermediate output from the CPU and continues processing the full model inference. This approach offers potential latency reduction by exploring the accuracy-latency tradeoffs. The application only relies on the more advanced but slower classifier for challenging cases, ensuring that accuracy is maintained while minimizing overall average latency.

The effectiveness of this approach largely depends on the models chosen, as highlighted in previous work [18], [19]. First, inference on the smaller model must be fast while still maintaining high accuracy. Second, the two models should behave differently, ideally not returning low-confidence results for the same set of inputs. This ensures that if the smaller model fails, the larger model still has a strong likelihood of producing a high-confidence result. Additionally, as will be discussed in Section V, dynamics such as changes in power levels can result in throttling, which affects the performance of each processor. Therefore, adaptive techniques are needed to select the appropriate models for each processor at runtime.

B. Inter-component Collaboration

Next, we discuss two possible implementations of inter-component collaboration: device-edge collaboration and edge-edge collaboration.

1) Device-edge collaboration: In device-edge collaboration, a sensing node first partially processes the inference request using its accelerator. Subsequently, the sensing node transmits the intermediate output to a nearby edge server over the network, and the edge server runs the remaining computation.

Figure 3 illustrates an example of a sensing node and an edge server collaboratively executing an image classification request. The sensing node captures an image with its camera and processes it through the first k layers of the neural network. Then, it transmits the intermediate output of layer k to the edge server, which completes the processing with the remaining layers. This approach reduces latency by leveraging the strengths of both on-device and edge processing. Instead of performing the entire inference on the sensing node, it takes advantage of the edge server's superior hardware for the final

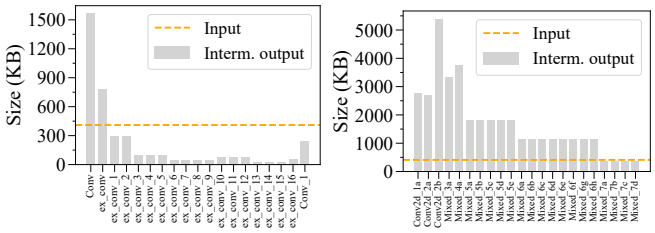

Fig. 4. Intermediate output size of each layer for the (a) MobileNetV2 and (b) InceptionV4 models.

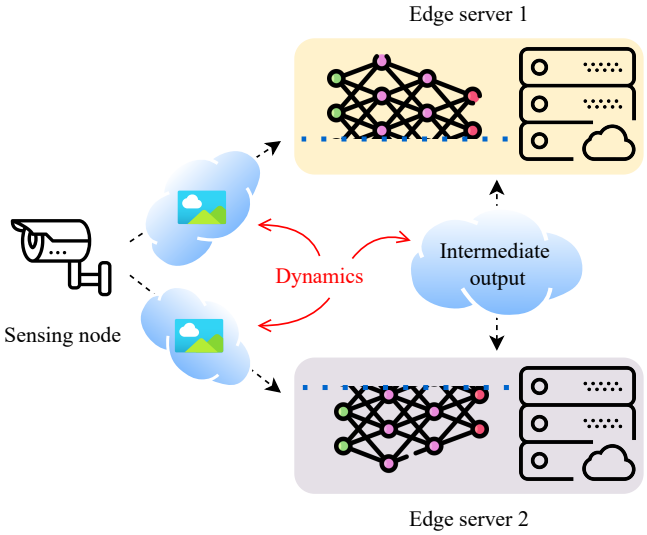

Fig. 5. Edge-edge collaboration executing an inference request.

computation. Additionally, it minimizes the payload size by transmitting only the intermediate output rather than the full input.

One critical consideration when using this approach is determining the amount of processing to be performed by the sensing node [8], [9]. Specifically, to reduce latency compared to the inference offloading approach in Figure 1, the intermediate output size must be smaller than the input size. This requirement means that the applicability of this approach is highly model-dependent. Figure 4 shows the intermediate output sizes of two commonly used image classification models– MobileNetV2 and InceptionV4. As illustrated in Figure 4a, the output size of the *ex_conv_6* layer of MobilenetV2 is 8× smaller than the input size, making it a viable candidate for splitting. In contrast, as shown in Figure 4b, all layers of InceptionV4, except for the last four, have output sizes larger than the input, making it less suitable for this strategy unless compression techniques [20]–[22] are used.

2) Edge-edge collaboration: The second possible inter-component collaboration implementation is edge-edge collaboration, where multiple edge servers work together to execute an inference request. Unlike device-edge collaboration where

TABLE II
COMPARING INTRA-COMPONENT COLLABORATION WITH BASELINES FOR CIFAR-10 ON JETSON ORIN NANO.

	Baselines		Collaborative inference		
	Res-8	Res-50	$\alpha = 0.7$	$\alpha = 0.8$	$\alpha = 0.9$
Accuracy (%)	78	86	80	82	84
Latency (ms)	32	91	38	43	47

inference is performed sequentially, edge-edge collaboration explores parallelism in the execution. Figure 5 presents an illustrative example of this collaboration approach, showing two edge servers collaboratively processing an inference request. Here, the neural network is horizontally partitioned into two or more segments, with each server hosting a portion of the network. Each edge server processes its assigned segment in parallel and communicates over the network to exchange intermediate outputs to complete the inference execution.

Previous work [12] has focused on minimizing network communication overhead during inference. However, new challenges arise with dynamic conditions and adversarial factors (discussed in Section V) in edge networks. Therefore, solutions must be adaptive to these changing conditions to maintain performance and reliability.

IV. PRELIMINARY RESULTS

In this section, we present preliminary experimental results to demonstrate the effectiveness of the discussed collaborative inference implementations.

Table II compares the performance of the intra-component collaboration approach discussed in Section III against two baseline models. We use a proof-of-concept collaborative inference setup, where an Orin Nano sensing node hosts the smaller, less accurate ResNet-8 model on the GPU and the larger, more accurate ResNet-50 model on the CPU. We execute the inference request on both processors simultaneously. If the confidence of the prediction result from ResNet-8 falls below the classification threshold , we load the intermediate results from the CPU and continue processing with the ResNet-50 model on the GPU; otherwise, we directly return the result. We vary from 0.7 to 0.9 and compare the accuracy and latency of the proposed approach with scenarios where only the Resnet-8 or Resnet-50 model is used on the GPU. We observe that, with = 0:9, the collaborative approach achieves a 48% reduction in latency compared to the ResNet-50-only approach while incurring only a 2% decrease in accuracy. Compared to the ResNet-8-only approach, the proposed method provides a 6% improvement in accuracy, with a trade-off of a 46% increase in latency at = 0:9. These results illustrate that the collaborative inference approach effectively balances latency and accuracy. The user-defined allows for fine-tuning this trade-off, showcasing the potential of this approach for optimizing performance in practical applications.

Figure 6 illustrates the performance of various inference execution strategies, including device-only execution, edge-only execution, and inter-component collaborative inference.

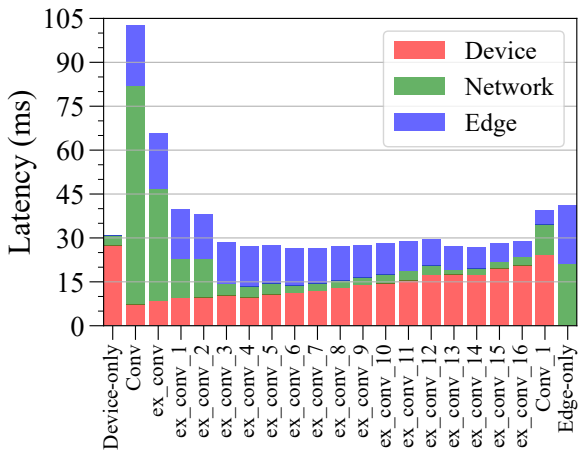

Fig. 6. Comparing the performance of different inference execution strategies.

We use a Jetson Orin Nano as the device and a server equipped with Nvidia's GTX 1080 GPU as the edge. The results indicate that splitting the inference at layer *ex_conv_6* yields the optimal performance, as it effectively minimizes network latency while leveraging the superior hardware capabilities of the edge server for the latter half of the inference processing. We note, however, that the performance of different strategies may vary with dynamic conditions, such as network fluctuations and edge server load, which we will discuss in the next section.

V. Challenges from Dynamics

In addition to static configurations, dynamic conditions within edge networks can significantly impact the efficiency of collaboration strategies. This section investigates the challenges posed by these dynamic conditions.

A. Network Variability

Network conditions in IoT environments are highly variable and can be further exacerbated by the presence of adversaries [17]. Such variability may lead to intermittent connectivity, network congestion, and even network partitions, all of which affect the performance of collaborative inference strategies. For example, decreased network bandwidth can increase communication overhead in edge-edge collaborative approaches. In mission-critical scenarios, network partitioning can result in disconnections, potentially compromising the application's ability to function as intended. Therefore, to ensure continued functionality and optimal performance, applications must be resilient to these dynamic network changes.

B. Node Failures

Node failures are common in IoT settings due to the numerous network components and environmental uncertainties. In adversarial environments, attacks on battlefield intelligent devices and servers can also lead to failures, potentially causing severe consequences. For instance, critical information may be lost if an edge server fails during a collaborative inference request. To ensure robust performance, collaborative inference applications must be resilient to node failures.

C. Thermal Conditions and Energy Constraints

Environmental or operational heat can significantly impact sensing node performance. Managing thermal conditions becomes even more critical in adversarial environments, where sensing nodes might be subjected to harsh conditions or intentional interference. A major source of heat within the device is prolonged inference execution. For instance, in an IoBT setting, a sensing node might continuously perform multiple inference tasks, such as audio and seismic classification for object detection. This can generate substantial heat, potentially leading to overheating and thermal throttling, which forces the system to reduce the clock speed of the CPU and accelerators to prevent hardware damage, leading to increased latency [23]. Moreover, sensing nodes, such as field-deployed sensors, are often required to operate over long mission lifetimes. Many of these nodes rely on batteries, and as the power level decreases, nodes may enter power-saving modes that underclock the processors, negatively affecting inference performance. To maintain efficiency and performance under varying thermal and power conditions, adaptive techniques are necessary to dynamically forward requests or adjust execution strategies.

D. Mission Dynamics

Mission goals may dynamically change in real time in response to evolving situations. For example, an initial mission might focus on detecting new subjects within video frames in a battlefield context. Once a subject is identified, the mission could shift to classifying the detected subject into specific categories. This change requires the collaborative inference application to reconfigure its resource allocations or reoptimize the collaborative inference strategy. In the previous example, the application should prioritize classification requests over detection requests from other sensors to align with new mission objectives. Applications must provide such flexibility to ensure that mission-critical tasks continue to function effectively even as goals and conditions evolve.

E. Environmental Variability

Environmental variability can impact the performance of collaborative inference systems. Factors such as adversarial interference, changing weather conditions, and fluctuations in ambient noise levels can affect the accuracy and reliability of sensors, potentially diminishing their effectiveness. To ensure that the system maintains optimal performance and accuracy despite varying environmental conditions, the collaborative inference application must dynamically adapt by reconfiguring its data fusion topology and analytics workflow to leverage the most reliable sensors available.

F. Workload Fluctuations

Workload fluctuations on edge servers can result in increased queuing delays and prolonged latency. For example,