Intra-DP: A High Performance Collaborative Inference System for Mobile Edge Computing

Zekai Sun, Xiuxian Guan, Zheng Lin, Zihan Fang, Xiangming Cai, *Member, IEEE*, Zhe Chen, *Member, IEEE*, Fangming Liu, *Senior Member, IEEE,* Heming Cui, *Member, IEEE*, Jie Xiong, *Senior Member, IEEE*, Wei Ni, *Fellow, IEEE*, and Chau Yuen, *Fellow, IEEE*

***Abstract*—Deploying deep neural networks (DNNs) on resource-constrained mobile devices presents significant challenges, particularly in achieving real-time performance while simultaneously coping with limited computational resources and battery life. While Mobile Edge Computing (MEC) offers collaborative inference with GPU servers as a promising solution, existing approaches primarily rely on layer-wise model partitioning and undergo significant transmission bottlenecks caused by the sequential execution of DNN operations. To address this challenge, we present Intra-DP, a high-performance collaborative inference system optimized for DNN inference on MEC. Intra-DP employs a novel parallel computing technique based on local operators (i.e., operators whose minimum unit input is not the entire input tensor, such as the convolution kernel). By decomposing their computations (operations) into several independent sub-operations and overlapping the computation and transmission of different sub-operations through parallel execution, Intra-DP mitigates transmission bottlenecks in MEC, achieving fast and energy-efficient inference. The evaluation demonstrates that Intra-DP reduces per-inference latency by up to 50% and energy consumption by up to 75% compared to state-of-the-art baselines, without sacrificing accuracy.**

***Index Terms*—Computation offloading, model inference, mobile edge computing, distributed system and network**

I. Introduction

Deep Neural Networks (DNNs) have empowered a wide range of mobile applications, from intelligent wearable sensors [1] and autonomous vehicles [2] to industrial IoT systems [3]. These applications are built upon advances in object detection [4], robotic control [5], and environmental perception [6], all of which demand fast inference for swift responses.

Z. Sun, X. Guan, and H. Cui are with the Department of Computer Science, University of Hong Kong, Pok Fu Lam, Hong Kong SAR, China. Z. Sun and H. Cui are also with Shanghai AI Laboratory, Shanghai, China. (e-mail: zksun@cs.hku.hk; xxguan@cs.hku.hk; heming@cs.hku.hk).

Z. Lin, Z. Fang, and Z. Chen are with the Institute of Space Internet, Fudan University, Shanghai 200438, China, and also with the School of Computer Science, Fudan University, Shanghai 200438, China (e-mail: zlin20@fudan.edu.cn; zhfang19@fudan.edu.cn; zhechen@fudan.edu.cn).

X. Cai is with the School of Physics and Information Engineering, Fuzhou University, Fuzhou 350108, China (e-mail: xiangming.cai@fzu.edu.cn).

F. Liu is with the Peng Cheng Laboratory, Shenzhen, China, and Huazhong University of Science and Technology, Wuhan, China (e-mail: fmliu@hust.edu.cn).

J. Xiong is with the School of Computing and Data Science, Nanyang Technological University, Singapore 639798 (e-mail: jie.xiong@ntu.edu.sg).

W. Ni is with Data61, CSIRO, Marsfield, NSW 2122, Australia, and the School of Computing Science and Engineering, and the University of New South Wales, Kennington, NSW 2052, Australia (e-mail: wei.ni@ieee.org).

C. Yuen is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: chau.yuen@ntu.edu.sg).

(Corresponding author: Heming Cui)

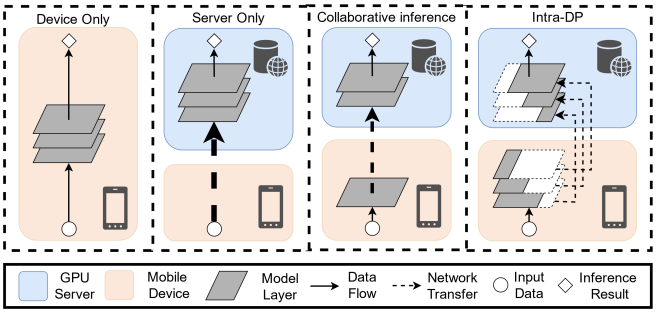


Fig. 1: Comparison between conventional Mobile Edge Computing approaches and Intra-DP. Inference results computed on the GPU server must be transmitted back to the mobile device for application utilization.

Deploying DNN models on real-world mobile devices (e.g., smartphones and IoT devices) presents two major challenges: the need for fast inference and the limitation of computing resources, including computational power and battery life.

Recent studies [7]–[16] have proposed mobile edge computing (MEC) as a promising solution for fast and energy-efficient inference by leveraging GPU servers at the edge of the radio access network. Conventional MEC approaches (illustrated in Fig. 1) are categorized into device-only, server-only, and collaborative inference [9], [16]. Device-only inference is constrained by limited on-device computing resources (see Sec. II-A), while server-only inference suffers from bandwidth limitations (see Sec. II-B). Collaborative inference strikes a better balance between computation and communication through layer-wise model partitioning (layer partitioning), which distributes DNN layers across mobile devices and GPU servers. Since some intermediate layers produce significantly smaller activations than raw input data [12], this approach reduces transmission overhead, further optimizing resource utilization (see Sec. II-D).

Despite its advantages, the sequential nature of layer partitioning introduces significant transmission delays, making it a major bottleneck for collaborative inference in MEC. Existing methods enforce strictly sequential execution, consisting of early-layer computation on mobile devices, intermediate result transmission, and final inference on GPU servers. Limited network bandwidth (due to MEC network limitations and practical instability; see Sec. II-B) prolongs transmission (about half of inference time in our experiments), causing significant idle periods on mobile devices. This not only increases overall inference time but also results in significant energy waste (around 40%). Although pipeline execution [17]

attempts to alleviate transmission overheads by overlapping computation and communication across multiple requests, its primary benefit lies in improving throughput, as opposed to reducing the inference time of individual requests, which is a crucial requirement for real-time mobile applications.

To address these limitations, a parallel execution strategy that overlaps computation and communication within a single inference request offers a promising solution. Unlike traditional sequential execution, this approach allows computation and data transmission to proceed concurrently, significantly reducing idle time on mobile devices. This speeds up inference and improves energy efficiency, as idle-period energy consumption is primarily driven by core components such as the CPU, GPU, and memory modules [18] (95% of total energy in our experiment), while wireless network interface cards contribute only 1.5%.

Implementing this parallel computing technique poses three key challenges. ① Traditional parallel computing methods (data, tensor, and pipeline parallelism) work well in data centers but struggle in MEC due to batch size limits, high synchronization costs, and transmission delays (see Sec. II-C), necessitating a finer-grained scheduling unit. ② Fine-grained parallelism is vulnerable to consistency issues, where the failure or error of any single unit (e.g., receiving incorrect input) can compromise the entire inference result. ③ A new scheduling paradigm is essential to balance computation and transmission within such fine-grained parallelism, yet its optimization is significantly complicated by two primary factors: an expanded search space from $O(n)$ (layers) to $O(n^2)$ (finer-grained scheduling units within layers), and the distinctive transmission and computing constraints inherent to MEC.

To address these challenges, we propose Intra-DP (Intra-Data Parallel), a high-performance collaborative inference system optimized for MEC. ① Intra-DP defines scheduling units based on local operators, where the minimum input is a subset of the full tensor (e.g., individual elements for ReLU [19] and tensor blocks for convolution [8]) and their computations (operations) can be decomposed into independent sub-operations (local operations). This intrinsic property allows local operations in subsequent layers to commence execution once their specific inputs are available, eliminating the need to await completion of all local operations in the current layer. Such a capability facilitates finer-grained parallel intra-data scheduling, thereby achieving computation-communication overlap even within a single inference request. ② To ensure result correctness and completeness, we introduce Local Operation Parallelism (LOP), a novel parallel computing technique operating at the granularity of local operations. It guarantees that each operation receives the correct input and propagates the correct output during parallel execution by managing data dependencies, wherein the output of one operation serves as the designated input for another. ③ To achieve fast and energy-efficient inference, we propose the Local Operation Scheduling Strategy (LOSS) for optimally distributing local operations between mobile devices and GPU servers. This strategy formulates the distribution task as a constrained optimization problem that leverages LOP, incorporates inherent MEC constraints, and is solved efficiently using differential evolution [20].

In summary, this paper makes the following contributions:

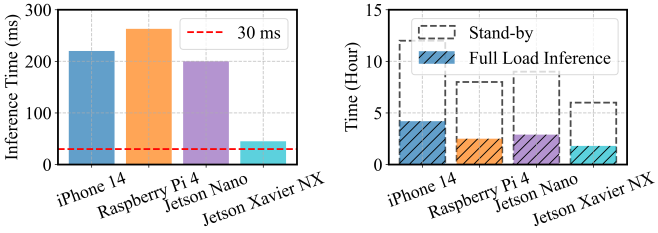
- We propose Intra-DP, a novel collaborative inference system that enables fine-grained parallel scheduling based on local operators, specifically optimized for MEC. The code is released on https://github.com/hku-systems/intraDP.
- We develop a comprehensive parallel computing technique (LOP) that guarantees inference correctness based on local operations while enabling efficient computation-transmission overlapping.
- We design an innovative scheduling algorithm (LOSS) that achieves fast and energy-efficient inference based on a constrained optimization formulation for distributing local operations in MEC.
- We implement an adaptive control mechanism to dynamically handle real-time network bandwidth fluctuations in MEC.
- We empirically evaluate Intra-DP through extensive experiments, demonstrating its superiority over state-of-the-art baselines in MEC in terms of inference latency and energy efficiency.

The rest of the paper is organized as follows. Sec. II introduces the fundamental concepts and technical background of Intra-DP. Sec. III presents the overview of Intra-DP, with detailed design in Sec. IV. Sec. V introduces the system implementation, followed by performance evaluation in Sec. VI. Related works and technical limitations are discussed in Sec. VII. Finally, conclusions are presented in Sec. VIII.

II. Background and Motivation

A. Device-only Inference

Device-only inference, which deploys DNN models directly on mobile devices, is severely constrained by the limited computational power of processors and battery capacity. As shown in Fig. 2a, the inference latency on these devices exceeds the 30 ms threshold required for smooth video fluency [21] (indicated by the red dotted line). Fig. 2b demonstrates that frequent model inference significantly reduces device standby time to 20–40% of their normal duration, severely affecting user experience and overall device practicality.



(a) Inference Time (b) Battery Life

Fig. 2: The performance of VGG-16 under device-only inference across different mobile devices [22]–[26].

B. Server-only Inference

Server-only inference, which deploys DNN models directly on a GPU server, is susceptible to long tail delays and requires substantial bandwidth for data transmission. As shown in Fig. 3, under the same testbed settings as Sec. V (a robot

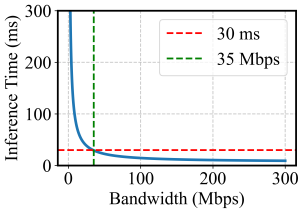


Fig. 3: The inference time of VGG-16 under server-only inference across different network bandwidth.

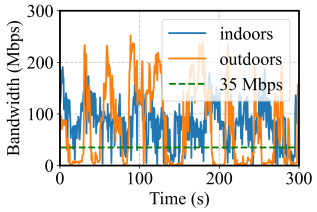
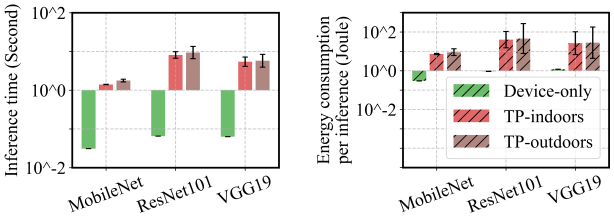


Fig. 4: The wireless transmission instability of TCP between our robot and the base station in MEC networks.



(a) Inference Time (b) Energy Consumption

Fig. 5: The performance of TP for different models.

equipped with a Jetson Xavier NX [24] and a GPU server with an NVIDIA GeForce GTX 3080), inference time increases sharply as bandwidth decreases. However, in real-world scenarios, mobile devices primarily rely on wireless networks, which offer high mobility but limited bandwidth compared to data center networks equipped with high-speed technologies (e.g., 40–500 Gbps for InfiniBand [27] and PCIe [28]).

The inherent bandwidth capacity of wireless networks in MEC is subject to restrictions stemming from both theoretical limits and practical implementation factors. While Wi-Fi 6 can achieve a peak bandwidth of 1.2 Gbps per stream [29], mobile devices lack the necessary hardware to fully leverage this capacity [30]. The actual available bandwidth varies significantly due to factors such as device mobility [31], signal obstruction [32], and channel contention [33].

To examine wireless instability in MEC scenarios, we conducted a robot surveillance experiment where four-wheeled robots navigated through a lab (indoors) and a campus garden (outdoors) at speeds of 5–40 cm/s. Using iperf [34], we measured real-time wireless bandwidth capacity between the robot and a base station over TCP [35] at 0.1-second intervals for five minutes. As shown in Fig. 4, the average bandwidth was 93 Mbps indoors and 73 Mbps outdoors, with outdoor measurements exhibiting higher fluctuations and occasional near-zero drops due to obstacles and reduced signal reflections, confirming that server-only inference is difficult to sustain under realistic wireless network conditions.

C. Related Parallel Computing Techniques

Parallel computing techniques have been extensively explored and proven effective in modern data centers [17], [36], [37]. However, they are unsuitable for MEC due to fundamental differences in computational constraints and latency requirements. The three primary parallelism strategies in data centers are data parallelism (DP), which replicates the model across devices and processes mini-batches in parallel; tensor parallelism (TP), which partitions computations within a layer across multiple devices; and pipeline parallelism (PP), which distributes different layers of DNN model across devices (layer partitioning) and executes them sequentially (pipeline execution) to reduce idle time.

DP's efficiency is fundamentally constrained by batch size requirements [36]. In data centers, large batch sizes (e.g., 16 images) are split into mini-batches (e.g., 2 images each), maintaining computational efficiency. However, MEC requires real-time inference with minimal latency, typically processing a single input at a time (e.g., 1 image). Further splitting such small inputs into sub-mini-batches (e.g., 1/4 of an image) is impractical, making DP inefficient.

TP enables parallel computation within a single layer across devices but incurs substantial communication overhead due to the required all-reduce synchronization [37]. We evaluated DINA [8], a state-of-the-art TP method, on our testbed (see Sec.V). As shown in Fig. 5, the excessive all-reduce overhead results in an inference time 45.2 to 143.9 times, and an energy consumption per inference 28.5 to 62.7 times, that of device-only inference. While there have been efforts to reduce the communication overhead of TP in data centers, such as distributing each layer along both the spatial and temporal dimensions [38], these methods remain ineffective in MEC due to the persistent all-reduce requirement. In contrast, Intra-DP eliminates this communication for local operators, further reducing overhead.

PP employs pipeline execution to enhance throughput by overlapping computation and communication across multiple requests. However, this approach does not reduce the completion time of a single inference [17]. [39] attempts to optimize transmission efficiency through an early-exit policy combined with layer partitioning. While this reduces the number of transmissions at the cost of accuracy, it introduces unpredictable and incorrect inference results, and requires careful expert tuning of parameters. Additionally, transmission delays persist in [39], an issue that can be effectively addressed by the combination of Intra-DP without sacrificing accuracy.

In conclusion, while conventional parallel computing techniques have proven effective in data centers, they are unsuitable for MEC. DP is limited by batch size constraints, TP suffers from excessive synchronization overhead, and PP's pipeline execution focuses solely on throughput, offering no provision for inference latency reduction. Consequently, existing collaborative inference in MEC relies on layer partitioning, which enforces sequential execution and introduces transmission delays, making them a major bottleneck for low-latency inference.

D. Existing Collaborative Inference

Existing collaborative inference methods [9]–[16] primarily focus on layer partitioning to balance inference time and energy consumption, as shown in Fig. 6. "Layer 0" represents server-only inference, where all model layers run on a GPU server, and transmission time varies with network bandwidth, even under the same partitioning strategy. These methods can be categorized by their optimization objectives: accelerating

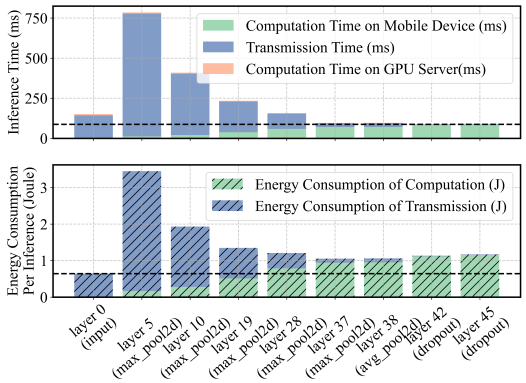


Fig. 6: The performance of different layer partitioning strategies for VGG-19 under 35 Mbps in our experiments. The X-axis represents various partitioning points, where 'layer i' denotes that all layers up to and including the i_{th} layer are executed on the robot, while the remaining layers are offloaded to the GPU server.

DNN inference [9]–[14] or minimizing energy consumption under deadline constraints [15], [16]. The widespread adoption of layer partitioning in mobile applications has drawn significant research attention [9]–[14], as optimal strategies depend on multiple factors, including model architecture, hardware capabilities (e.g., GPU servers and mobile devices), network conditions, and application-specific trade-offs between inference speed and energy efficiency. However, all existing methods [9]–[16] are inherently constrained by transmission delays due to the sequential nature of layer partitioning, a limitation that can be effectively addressed by Intra-DP.

While some methods attempt to mitigate transmission delays in collaborative inference through finer-grained scheduling units, such approaches lose effectiveness in MEC scenarios. [40], [41] propose input-splitting methods, distributing raw input across devices based on their computational power. However, their design for IoT device groups is ill-suited for MEC; the substantial computational disparity between mobile devices and GPU servers therein leads to near-complete reliance on server-side execution, effectively reverting to server-only inference. [42] employed tile-based scheduling for concurrent computation and communication, but the resulting input fragmentation from small tiles leads to substantial overhead of frequent invocation. To mitigate this, they utilize an asynchronous execution approach, which is CPU-restricted and consequently precludes effective GPU acceleration in MEC. Separately, their selection of coarser tile granularity, coupled with a simplistic, unscheduled greedy strategy, impairs overall parallel execution efficiency. These distinct limitations underscore the necessity for Intra-DP.

III. System Overview

In this section, we introduce Intra-DP, a high-performance collaborative inference system optimized for MEC. In this setting, resource-constrained mobile devices (e.g., robots, drones, and smartphones) collaborate with GPU servers through wireless networks (e.g., Wi-Fi 6/5G) to achieve real-time and energy-efficient inference on local inputs.

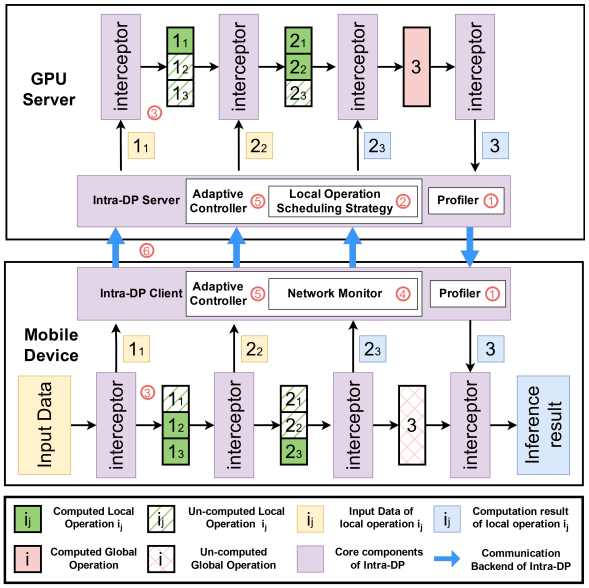


Fig. 7: Overview and inference workflow of Intra-DP. Local operation i_j represents the j-th local operation within the i-th operator of the model.

A. Key Insight

Existing methods are limited to layer partitioning, which enforces serial execution and limits parallelism. Each DNN layer comprises one or more operators (e.g., convolution [8], Softmax [43]), which must be executed sequentially to ensure computational correctness. As a result, current approaches can only partition these serial execution sequences at the layer level, restricting opportunities for parallel execution.

Our key insight is that, although operators are the smallest computation units at the system level, their computations (operations) can still be decomposed into several independent sub-operations based on their computational properties. Specifically, local operators, whose computations do not require access to the entire input tensor, enable finer-grained parallel execution. Existing methods treat an operator's computation based on its entire input as a single, indivisible operation. In contrast, Intra-DP leverages the inherent divisibility within local operators by treating each individual computation, based on its minimal input unit within the local operator, as an independent local operation. Since these local operations are independent, those local operations in subsequent layers can be computed as soon as their required inputs are ready, without waiting for all operations in the current layer to finish. This independence enables a new parallelization opportunity within serial execution sequences, breaking the strict sequential execution constraint.

B. Overall Workflow

As illustrated in Fig. 7, Intra-DP consists of three stages: Profiling, Scheduling Optimization, and Runtime.

The first stage, Profiling (①), is an offline process that collects essential runtime traces for Scheduling Optimization. It runs only once and captures key execution characteristics, including: i) operator types and input/output sizes; ii) execution times of operations on the mobile device and GPU server;

iii) data dependencies between operations (i.e., the output of one operation serves as the input for another).

Using this profiling data, the system enters the Scheduling Optimization stage, where it applies Local Operation Scheduling Strategy (see Sec. IV-C, ②) to achieve low-latency, energy-efficient inference. This strategy determines the optimal distribution of local operations, balancing computational workload and transmission synchronization. To mitigate the impact of network fluctuations, Intra-DP precomputes optimal scheduling plans for varying network bandwidth conditions in 1 MB intervals within the typical MEC bandwidth range (0˜50 MB), eliminating runtime optimization overhead.

During Runtime, Intra-DP utilizes Operator Interceptors (see Sec. IV-A, ③) to manage input/output tensor partitioning and reconstruction for each operator, enabling partial tensor processing based on local operations. To ensure reliability in MEC, the system continuously monitors network conditions utilizing established tools [44] (④) and Adaptive Controllers (see Sec. IV-D, ⑤) dynamically adjust scheduling strategies based on real-time bandwidth fluctuations, ensuring stable performance. Then, Local Operation Parallelism (see Sec. IV-B, ⑥) enables parallel execution between the client and server while preserving data dependencies for correct inference. To support seamless transitions between different schedules without delays, Intra-DP maintains a model replica on the GPU server (Fig. 7), allowing instant adaptation to network variations.

Despite these optimizations, Intra-DP introduces negligible overhead beyond the original inference process. The only additional cost stems from tensor splitting and combining. However, the impact of tensor splitting is effectively masked because it executes concurrently with data transmission and computation, allowing its associated latency to be absorbed by these parallel tasks. While tensor combining introduces potential waiting time due to data dependencies, LOSS minimizes this impact by formulating the waiting overhead as a nonlinear optimization problem and solves it efficiently. This careful design ensures that Intra-DP maintains high execution efficiency with minimal extra cost.

IV. Detailed Design

In this section, we detail the design of Intra-DP, a collaborative inference system enabling fine-grained parallel scheduling based on local operators, specifically optimized for MEC. We first identify and categorize local versus global operators (see Sec. IV-A), which establishes the foundation for our parallel execution strategy. Building upon this, Sec. IV-B introduces LOP, a parallel computing technique that guarantees inference correctness with these local operations while achieving efficient computation-transmission overlap. To effectively orchestrate these parallel local operations, Sec. IV-C presents LOSS, an innovative scheduling algorithm that achieves fast and energy-efficient inference by formulating the distribution of local operations as a constrained optimization problem. Finally, to ensure robust performance under MEC's inherent network fluctuations, Sec. IV-D describes an adaptive control mechanism designed for dynamic real-time network bandwidth management.

A. Local Operators and Global Operators

As the performance gains of Intra-DP rely on the parallel execution of local operations, it is crucial to determine the minimal input unit required for each operator based on its computational characteristics. To achieve this, Intra-DP employs Operator Interceptors (Fig. 7 ③) to handle input/output tensor partitioning and reconstruction, enabling partial tensor processing, and classify operators into local and global categories: local operators process only a subset of the input tensor, whereas global operators require the entire tensor. We identify three common types of local operators in models commonly used on mobile devices (mobile models):

- Element-wise local operators operate on individual tensor elements and are widely used in activation functions (e.g., ReLU [19], Sigmoid [45], and SiLU [46]). However, activation functions like Softmax [43] require access to the entire input tensor and are therefore global operators.
- Block-wise local operators process tensor blocks at corresponding positions, commonly found in convolution-related layers (e.g., convolution [8] and max pooling [47]). The block size is determined by the operator's parameters (e.g., kernel size, padding, and dilation) [40].
- Row-wise local operators operate on individual rows of the input tensor and are widely used in matrix operations (e.g., addition and multiplication). This operational paradigm involves partitioning the input matrix by rows, enabling each device to process an independent subset while critically operating with a shared (or replicated, i.e., non-partitioned) layer parameter matrix.

$$\begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} \times \begin{pmatrix} b_1 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{pmatrix} \tag{1}$$

The capacity of these operators to eliminate all-reduce communication is founded on a core matrix calculation principle: an operation on an input row a_1 yields an output row $(c_{11} \cdots c_{1n})$ that is itself directly processable by subsequent matrix operators in a similar row-wise manner. This self-contained, per-row processing characteristic inherently obviates the need for inter-device synchronization via all-reduce. TP, in contrast, partitions the layer's parameter matrix and replicates the full input matrix across devices, a strategy that necessitates all-reduce operations to aggregate partial results. Row-wise local operators circumvent this costly communication by enabling independent row processing on each device coupled with a shared parameter matrix. Moreover, LOP treats operators with layer parameter matrices containing only one row as global operators.

Model	Local / Global	Model	Local / Global
DenseNet [48]	428 / 3	RegNet [49]	231 / 3
ResNet [50]	341 / 3	VGGNet [51]	73 / 5
ResNeXt [52]	342 / 3	ConvNeXt [53]	340 / 6

TABLE I: Number of local/global operators in mobile models.

Models with many local operators, prevalent in mobile models (e.g., CNNs for computer vision [4] and point cloud

processing [5]), can benefit from Intra-DP through parallel execution. Table I quantifies their distribution across several mobile models based on the above definition.

While local operators dominate CNN-based models, transformer-based models rely on self-attention mechanisms, which use Softmax [43], a typical global operator, to compute attention weights over entire input sequences. Since each transformer layer contains at least one global operator, the performance gains of Intra-DP over existing collaborative inference methods are limited, as its advantages primarily come from parallelizing local operations, which is constrained by the presence of global operators. However, Intra-DP still outperforms existing methods by enabling parallel execution of local operations between consecutive global operators and seamlessly falling back to conventional layer partitioning when handling models without local operators.

B. Local Operation Parallelism

In this section, we demonstrate how Intra-DP ensures inference correctness based on local operators and how LOP achieves low latency and energy efficiency through the parallel execution of local operations (Fig. 7 ⑥).

LOP guarantees inference correctness by comprehensively managing dependencies between local and global operators. It tracks data dependencies to ensure that the output of one operation is correctly used as the input for subsequent operations, considering a dependency established if any part of the output serves as input. For local operators, it preserves computational correctness by enforcing the appropriate execution sequence and ensuring that each operation receives the correct input (either a raw input data or the output from a preceding operation) based on the DNN model structure and operator computation logic. For global operators, synchronization barriers are introduced to guarantee that all required input tensors are fully assembled before execution. By maintaining proper data flow throughout the inference process, this approach guarantees correctness across both local and global operators.

Fig. 8 compares the workflow of LOP with conventional TP and PP (layer partitioning). To reduce transmission overhead, LOP employs two key optimizations. First, it overlaps computation and transmission for different local operations to maximize execution efficiency (e.g., transmitting the input for '$LO1_1$' while computing '$LO1_2$', '$LO1_3$', and '$LO2_3$' on the robot). Second, it exploits data locality by prioritizing execution on devices that already contain the required input (e.g., '$LO2_1$' directly using the output of '$LO1_1$' on the GPU server). Unlike TP, which suffers from high communication costs due to frequent all-reduce synchronization, LOP limits synchronization to global operators only, eliminating unnecessary data transfers. Although LOP incurs finer-grained communication and a higher volume than PP, its early overlapping strategy effectively reduces overall transmission time, outperforming PP's sequential execution for individual inference requests. Through the effective overlapping of computation and communication, Intra-DP curtails idle time on mobile devices, thereby accelerating inference and bolstering energy efficiency. This improvement is significant because energy consumption in idle states is chiefly governed by core components (e.g., CPU, GPU, and memory modules), as opposed to wireless network interface cards.

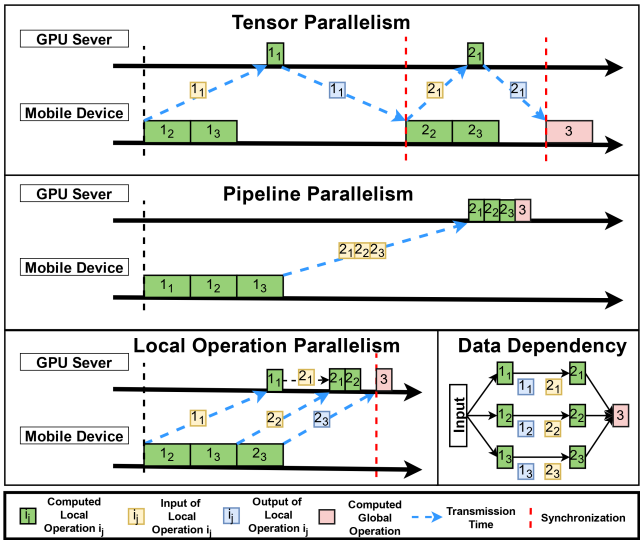


Fig. 8: Workflow of TP, PP and LOP. In the three cases above, each local operator executes three operations with identical computation times on both the mobile device and the GPU server, along with the corresponding transmission time, while the lower right corner illustrates the data dependencies between operations.

C. Local Operation Scheduling Strategy

In this section, we detail how Intra-DP optimizes computation and transmission scheduling for local operations, achieving fast and energy-efficient inference (Fig. 7 ②).

To ensure efficient inference across varying network conditions, LOSS operates under the assumption of stable bandwidth during each individual inference request. This assumption is justified because the typical model inference duration (tens to hundreds of milliseconds) is significantly shorter than the timescale of common wireless bandwidth fluctuations (on the order of seconds [54]). Leveraging this stability, LOSS formulates the local operation scheduling problem for a given network bandwidth as a constrained optimization problem, which is then efficiently solved using differential evolution [20]. To streamline heterogeneous operation integration, it unifies global and local operations by treating global operators as special cases that aggregate all outputs from preceding local operations. The key performance boost stems from computation-communication overlapping, where LOSS leverages data locality by prioritizing execution on devices that already contain the required input.

When multiple operations, particularly block-wise ones, located on both mobile devices and GPU servers concurrently depend on the same input, this shared input must be transferred irrespective of its original location. To further optimize such scenarios, LOSS exploits data locality by introducing local operation replication, which allows select operations to be executed simultaneously on both mobile devices and GPU servers. The effectiveness of local operation replication stems from the common scenario where the shared input is the output of a preceding operation; replication ensures this output, now the required shared input, can be directly obtained locally on each device without extra transmission. This strategy in-

curs minimal computational redundancy but yields significant reductions in data transmission costs, proving particularly beneficial for distributed operations that exhibit shared input dependencies.

1) DNN Execution Model

A DNN is represented as a directed acyclic graph $G = (V, E)$, where vertices V denote layers (operators) and edges E indicate data dependencies. Two virtual vertices, $input$ and $output$, represent the model's initial input and final inference result. The execution spans two instances: one on a resource-constrained mobile device (M) and the other on a GPU server (R).

2) Variables and Functions

We treat each calculation of an operator based on its minimum input unit as an operation, and define $OP(v)$ as the set of operations for vertex v, where global operators always have $|OP(v)| = 1$. Operation allocations to M and R are represented by $x_M(v)$, $x_R(v) \subseteq OP(v)$, with overlaps ($x_M(v) \cap x_R(v) \neq \emptyset$) occurring when local operations are replicated, especially for block-wise operators.

Execution starts at times $s_M(v), s_R(v) \geq 0$, and computation durations on each device are estimated as $C_M(v)$ and $C_R(v)$. If no operation is performed on a device, its computation time is zero. Data transmission times include $T_{\text{MR}}(u, v)$ for sending data from M to R and $T_{\text{RM}}(u, v)$ for the reverse direction, both determined by network bandwidth and dependent on the volume of input/output data associated with the respective operations.

3) Objective Function

The primary objective of LOSS is to minimize end-to-end inference latency while ensuring that the final results are available on the mobile device. This is achieved by minimizing T, defined as:

$$\min T = s_M(\text{output}), \tag{2}$$

where $s_M(output)$ represents the time at which the mobile device receives the final inference result, thereby reducing the idle time on mobile device. While Intra-DP can be adapted to prioritize minimizing energy consumption under specified deadline constraints, similar to approaches in [15], [16], by modifying this objective function, such exploration is not pursued here. Optimizing such application-specific trade-offs between inference speed and energy efficiency falls beyond the scope of this paper.

4) Constraints

Data Partitioning. To ensure correctness, all operations must be executed: $x_M(v) \cup x_R(v) = OP(v), \forall v \in V$.

Location. Input data is initially available on M, ensuring $x_M(input) = input$ and $x_R(input) = \emptyset$. Similarly, the final results must reside on M, so $x_M(output) = output$ and $x_R(output) = \emptyset$.

Data Dependency. Operations start only after receiving all required inputs. For each v with parent connections $e = (u, v)$, execution on M follows:

$$s_M(v) = \begin{cases} s_M(u) + C_M(u), \\ \quad \text{if } x_M(v) - child(x_M(u)) = \emptyset, \\ \max\big(s_R(u) + C_R(u) + T_{\text{RM}}(u, v), \\ \quad s_M(u) + C_M(u)\big), \\ \quad \text{if } x_M(v) - child(x_M(u)) \neq \emptyset. \end{cases} \tag{3}$$

Here, child$(x(u))$ denotes the set of operations consuming outputs of $x(u)$ according to their data dependencies, and the transmission volume in $T_{\text{MR}}(u, v)$ is given by $x_M(v) -$ child$(x_M(u))$. When $x_M(v) -$ child$(x_M(u)) = \emptyset$, all required inputs for $x_M(v)$ are produced by operations on M; otherwise, $x_M(v)$ also depends on outputs from operations executed on R. Similar constraints apply to R.

Computational Efficiency. To mitigate excessive kernel launch overhead, stemming from input fragmentation when LOSS processes minimal units, and maximize GPU server resource utilization for faster overall inference, Intra-DP employs operation batching, instead of asynchronous execution [42]. This technique consolidates multiple fine-grained operations into a single system-level kernel launch, thereby preventing redundant invocations. By managing the partitioning and reconstruction of input/output tensors for each operator, this approach preserves fine-grained control over local operations while minimizing redundant executions. To formalize this, for all operators $v \in V$: $x_M(v) = \{i \mid i \in [\min(x_M(v)), \max(x_M(v))]\}$, $x_R(v) = \{i \mid i \in [\min(x_R(v)), \max(x_R(v))]\}$.

Transmission Efficiency. Substantial transmission costs in Mobile Edge Computing (MEC) necessitate the minimization of redundant data transfers. Leveraging the insight from layer partitioning methods [12], whereby certain intermediate DNN layers yield significantly smaller outputs than raw input data, we impose the following constraint to curtail this overhead. This also substantially reduces the search space, significantly accelerating the solving process of LOSS. For layers u whose outputs exceed the raw input size ($u \in \Pi$), with all $e = (u, v) \in E$: $x_M(v) -$ child$(x_M(u)) = \emptyset$, and $x_R(v) -$ child$(x_R(u)) = \emptyset$.

5) Solution Algorithm

As the search space increases from $O(n)$ (layers) to $O(n^2)$ (operations), LOSS employs differential evolution [20], leveraging its robust global search capability and parallelism to efficiently solve the nonlinear, non-convex scheduling problem. For a given hardware configuration, bandwidth emerges as the primary determinant for partitioning local operations. This insight allows LOSS to significantly constrain the search space, thereby circumventing the substantial exploration overhead characteristic of methods like reinforcement learning [55]. Consequently, optimized scheduling plans only require recalculation when the underlying hardware changes (e.g., an upgraded mobile accelerator or GPU server). While generating the scheduling plans for a model takes a few minutes, this computation is performed offline within the Scheduling

Algorithm 1 Intra-DP client at runtime stage

Input: Data input for inference `input`; DNN model `model`
Output: The inference result `ret`
Parameter: `Input(i)`,`Output(i)`: input and output of layer i; x_M^b, x_R^b: schedule plan under the b bandwidth.

1: $b \leftarrow$ TESTBANDWIDTH()
2: `Input(0)` $\leftarrow$ `input`
3: **for all** layer u in model **do**
4: **if** $Input(u) \neq \emptyset$ and $x_M^b(u) \neq \emptyset$ **then**
5: `output(u)` $\leftarrow$ COMPUTE(`Input(u)`, $x_M^b(u)$)
6: **end if**
7: **for all** edge $e = (u, v) \in E$ **do**
8: **if** $(x_M^b(v) - \texttt{child}(x_M^b(u)) \neq \emptyset$ **then**
9: `Input(v)` $\leftarrow$ COMBINE(`Output(u)`, `Receive()`)
10: **end if**
11: **if** $x_R^b(v) - \texttt{child}(x_R^b(u)) \neq \emptyset$ **then**
12: SEND(`Output(u)`, $x_R^b(v) - \texttt{child}(x_R^b(u))$)
13: **end if**
14: **end for**
15: **end for**
16: **return** $ret \leftarrow$ `Output(t)`

Optimization stage, imposing no impact on runtime inference performance.

D. Adaptive Control Mechanism

The adaptive control mechanism of Intra-DP operates on both the mobile device client and the GPU server (Fig. 7, ⑤). A backend network monitor on the mobile device ascertains TCP bandwidth at the transport layer; its energy consumption is negligible compared to that of inference computation. This mechanism enforces a stateless execution pattern: the mobile device and the GPU server synchronize current bandwidth and select an identical scheduling plan prior to each inference. The core assumption of LOSS is stable bandwidth throughout an individual inference request, which obviates the need to switch scheduling plans for ongoing inference. Leveraging model replicas on the GPU server alongside pre-computed scheduling plans, the adaptive control within Intra-DP facilitates seamless, delay-free transitions between different schedules, enabling instant adaptation to network variations. Consequently, Intra-DP maintains robust performance even as mobile devices transition between networks, since its scheduling decisions depend solely on the transport-layer bandwidth between the mobile device and the GPU server.

The detailed algorithms constituting this adaptive control mechanism are presented in Alg. 1 and Alg. 2. During runtime, Intra-DP estimates real-time bandwidth (line 1 in both algorithms) and selects a suitable scheduling plan. It then computes the designated local operations (line 5 in both algorithms), producing outputs that represent the computation results of these local operations or an empty set if no local computation is performed. Leveraging operation batching from LOSS's computational efficiency constraints, Intra-DP merges all operations within the same operator ($x_M^b(u)$) into a single kernel launch. This launch executes once the required inputs are ready, prioritizing inference correctness over strict adherence to the scheduled execution time ($s_M(u)$). The associated blocking overhead, primarily from tensor combining, is optimized in LOSS.

Algorithm 2 Intra-DP server at runtime stage

1: $b \leftarrow$ RECEIVE()
2: `Input(0)` $\leftarrow \emptyset$
3: **for all** layer u in model **do**
4: **if** $Input(u) \neq \emptyset$ and $x_R^b(u) \neq \emptyset$ **then**
5: `output(u)` $\leftarrow$ COMPUTE(`Input(u)`, $x_R^b(u)$)
6: **end if**
7: **for all** edge $e = (u, v) \in E$ **do**
8: **if** $x_M^b(v) - \texttt{child}(x_M^b(u)) \neq \emptyset$ **then**
9: SEND(`Output(u)`, $x_M^b(v) - \texttt{child}(x_M^b(u))$)
10: **end if**
11: **if** $(x_R^b(v) - \texttt{child}(x_R^b(u)) \neq \emptyset$ **then**
12: `Input(v)` $\leftarrow$ COMBINE(`Output(u)`, `Receive()`)
13: **end if**
14: **end for**
15: **end for**

If the scheduling plan requires data transmission, the server sends the computed outputs to the client (line 9 in Alg.2), which integrates them with its results for subsequent operators (line 9 in Alg.1). Similarly, if data needs to be sent from client to server, the client transmits the outputs (line 12 in Alg.1), and the server integrates them accordingly (line 12 in Alg.2). Otherwise, the outputs of the current operator directly serve as inputs for subsequent operators. Once all operations are completed, the final inference result remains on the client, ready for upper-layer applications (line 16 in Alg.1).

V. IMPLEMENTATION

In this section, we first elaborate on the implementation of Intra-DP, and we then introduce the experiment setup.

A. System Implementation

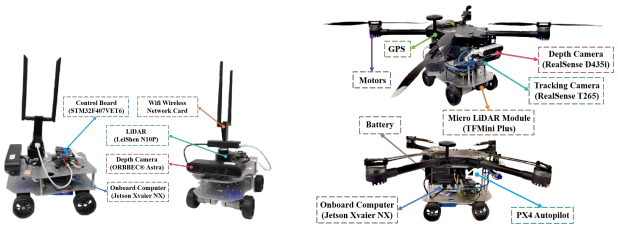
1) Software

We implemented Intra-DP using Python and PyTorch [56], integrating it seamlessly into existing deep learning workflows. The implementation hooks into the model's forward method, where the first forward pass profiles the model using PyTorch's default profiler and execution schedule. Based on this profiling data, Intra-DP then intercepts and parallelizes all subsequent forward calls according to the optimized execution plan. With a lightweight integration requiring only three lines of code, Intra-DP ensures ease of deployment in existing applications.

2) Hardware Testbed

We evaluated Intra-DP on two customized robotic platforms: a four-wheeled ground robot (Fig. 9a) and an air-ground hybrid robot (Fig. 9b). Each robot is equipped with a Jetson Xavier NX (8GB) [24], enabling on-board model inference with local computation. The system runs Ubuntu 20.04 with ROS Noetic and uses a dual-band USB network adapter (MediaTek MT76x2U) for wireless communication. Detailed hardware and sensor configurations are shown in Fig. 9. The GPU server is a PC equipped with an Intel i7-7700K CPU and an NVIDIA GeForce RTX 3080 GPU. It connects to the robots via Wi-Fi 6 on an 80 MHz channel at 5 GHz.

Tab. II summarizes the robots' on-board energy consumption (excluding motor power) in different states: inference (full GPU utilization, including CPU/GPU power), communication



(a) Four-wheeled robot (b) Air-ground robot

Fig. 9: The detailed composition of the robot platforms.

	inference	communication	standby
Energy (Watt)	13.35	4.25	4.04

TABLE II: Power draw (Watt) of our robot in different states.

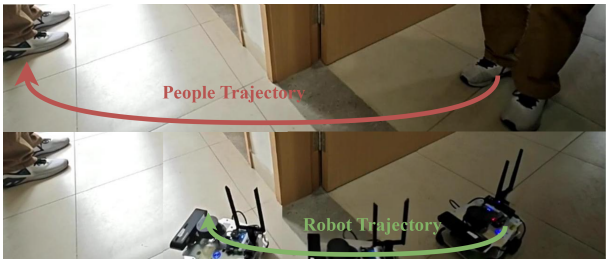


Fig. 10: Kapao [4], a real-time people-tracking application on our four-wheeled robot with a CNN-based human keypoint detection model.

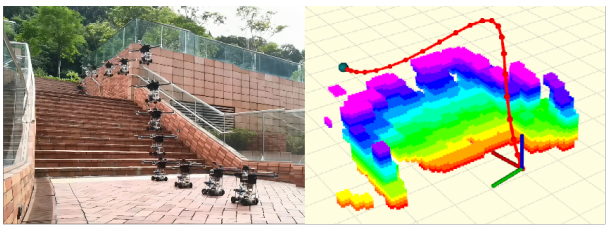


Fig. 11: AGRNav [5], an autonomous navigation application on our air-ground robot with a CNN-based 3D semantic scene completion model.

(communication with the GPU server, including wireless network card energy consumption), standby (no tasks to execute). Each Jetson Xavier NX is powered by a 21.6 Wh battery, sustaining up to 1.6 hours of continuous model inference. We continuously log the robot's instantaneous on-board power draw (in Watts) at 1-second intervals, utilizing the back-end power consumption and performance monitoring methodology from [24]. Subsequently, the energy consumption per inference (in Joules) is precisely calculated by integrating this power draw profile over the exact duration of each inference, determined by its start and end timestamps.

B. Experiment Setup

1) Task

We evaluated two typical real-world robotic applications on our testbed: Kapao [4] (Fig 10) and AGRNav [5] (Fig 11). We also evaluated several models common to mobile devices with their implementation from Torchvision [57] on a larger scale to further corroborate our observations and findings: VGGNet [51], ConvNeXt [53], ResNet [50], DenseNet [48].

2) Emulation Environments

We evaluated two real-world environments: indoors (robots move in our laboratory with desks and separators interfering with wireless signals) and outdoors (robots move in our campus garden with trees and bushes interfering with wireless signals, resulting in lower bandwidth). The corresponding bandwidths between the robot and the GPU server in indoors and outdoors scenarios are shown in Fig. 4.

3) Baselines

To comprehensively evaluate the performance of Intra-DP, we conducted comparative experiments against several baseline approaches:

- **Device-only inference ("Device-only")**: A conventional setup where the entire model is deployed and executed on the mobile device.
- **Server-only inference ("Server-only")**: A cloud-based approach in which the full model runs on a GPU server. While both Device-only and Server-only represent extreme cases of layer partitioning, we evaluate them separately to highlight the benefits of other baselines in terms of inference latency and energy efficiency.
- **DSCCS** [9]: A state-of-the-art layer partitioning method designed for inference acceleration. To ensure a fair comparison and eliminate the influence of parameter variations, we enhance DSCCS with Intra-DP's adaptive control mechanism, allowing it to dynamically adjust to wireless network fluctuations.
- **SPSO-GA** [16]: An advanced layer partitioning method that optimizes energy consumption while satisfying timing constraints. We configure SPSO-GA with a 1 Hz deadline constraint, corresponding to the minimum frequency required for effective robotic motion control. As with DSCCS, we integrate Intra-DP's adaptive control mechanism into SPSO-GA to allow real-time adaptation to network variations.

For consistency and accurate performance comparison, all systems are implemented without model compression, transmitting raw model activations to maintain accuracy. The Profiling and Scheduling Optimization stages of Intra-DP and all baselines run offline in advance.

VI. Performance Evaluation

In this section, we evaluate the performance of Intra-DP from three research aspects: i) comparisons with baseline systems to demonstrate the superiority of Intra-DP in inference time and energy consumption for real-world mobile applications; ii) investigating the LOSS of Intra-DP through micro-event analysis; iii) sensitivity studies to analyze system performance under varying network bandwidth variations, model structures, and equipment computational powers.

A. Superiority of Intra-DP

1) Inference Time

Fig. 12 demonstrates the superior performance of Intra-DP to four baseline methods across various tasks and environments, achieving significant speedup in inference time

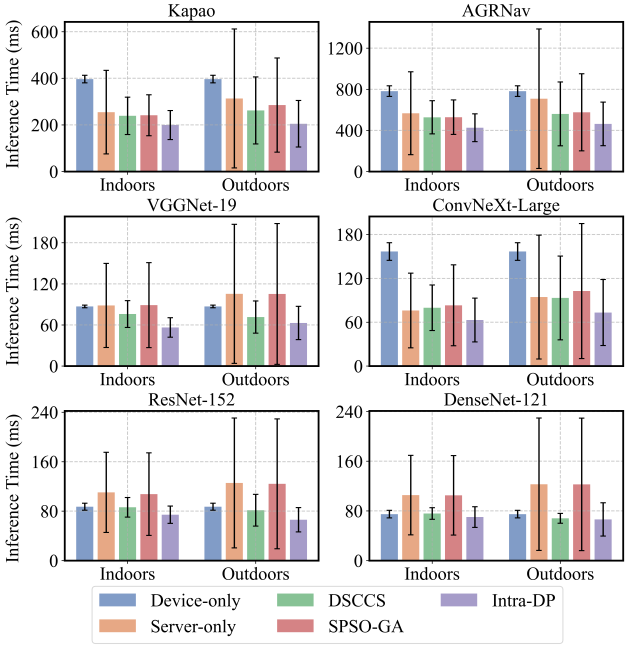


Fig. 12: Inference time for different models across various environments and systems.

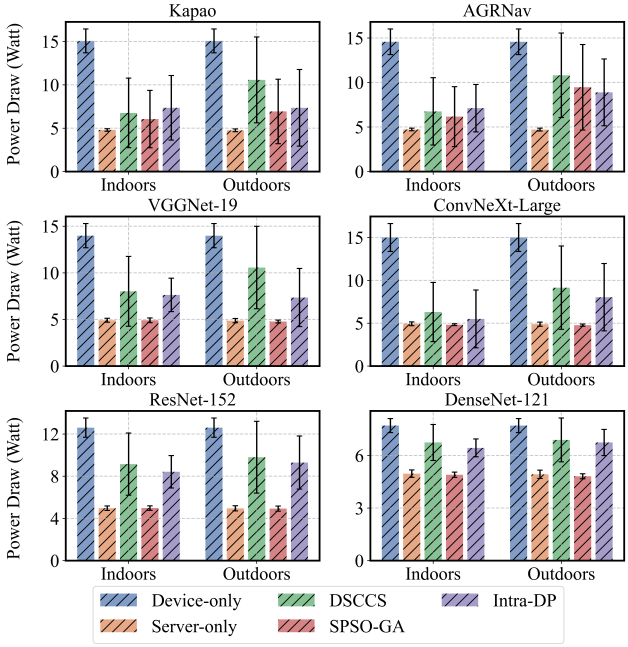


Fig. 13: Power draw for different models across various environments and systems.

(up to 50% for indoors and up to 48% for outdoors). Intra-DP outperforms the closest baseline DSCCS by 8% to 26% indoors and 8% to 22% outdoors. These improvements primarily result from LOP, which enables parallel execution and overlaps computation with transmission across local operations within the same inference request, and LOSS, which optimizes scheduling under varying network conditions. Notably, all offloading-based methods, including server-only, two layer partitioning methods, and Intra-DP, exhibit greater standard deviations and longer inference times outdoors due to severe bandwidth fluctuations, as shown in Fig. 4. Although the performance gain is less pronounced on DenseNet, this is analyzed in our sensitivity study in Sec. VI-C2. Additionally, a micro-event analysis in Sec. VI-B further explains the efficiency of Intra-DP in inference.

2) Energy Consumption

Fig. 13 presents the power draw across different methods and scenarios. The device-only approach consumes the most energy due to the intensive computational load on robots, while the server-only approach achieves the lowest energy consumption by offloading all computations to the GPU server. Among the partial-offloading methods, SPSO-GA demonstrates better energy efficiency than DSCCS due to its energy-oriented optimization, whereas Intra-DP exhibits slightly higher energy consumption due to occasional re-computation of local operations. The power draw between Intra-DP and DSCCS remains nearly the same, indicating that the computation-communication overlap in Intra-DP does not introduce significant additional energy costs. Tab. II further supports this conclusion, showing that robots continue consuming 95% of their active power even when idle, mainly due to static energy consumption in components such as the CPU, GPU, and memory [18]. In contrast, the wireless network card consumes only 0.21 W during transmission compared to 13.35 W during computation.

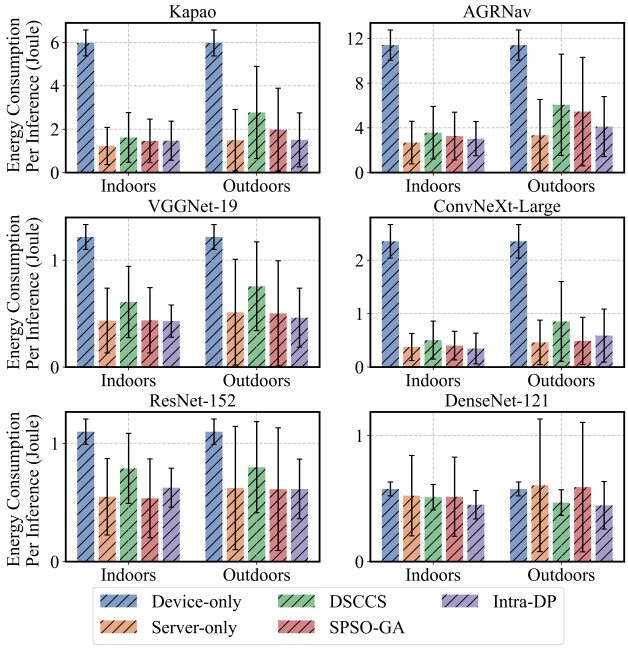


Fig. 14: Energy consumption per inference request for different models across various environments and systems.

Fig. 14 presents the energy consumption per inference request across different methods and scenarios. Although Intra-DP exhibits relatively higher power draw in Fig. 13, it achieves significantly lower energy consumption per inference request across all models and environments, reducing energy usage by up to 75% indoors and up to 72% outdoors due to its fast inference time. Notably, compared to the most energy-efficient server-only approach, Intra-DP increases energy consumption per inference request by at most 20% while reducing inference

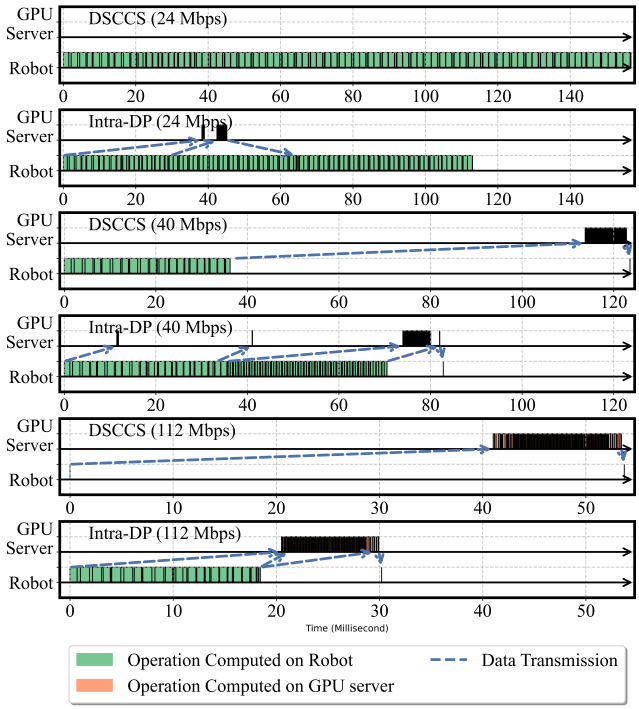


Fig. 15: Snapshots of schedule plan of Intra-DP and baseline during runtime under various network bandwidth.

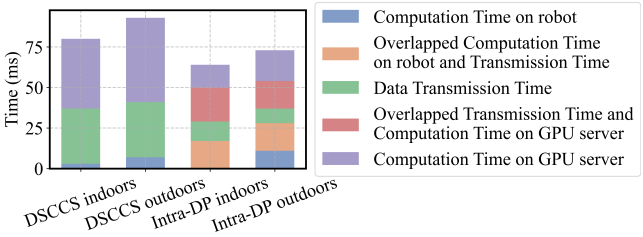


Fig. 16: Breakdown of each phase of the inference process.

time by up to 42% and greatly reduced long tail delays.

B. Micro-Event Analysis

Next, we present a detailed investigation of Intra-DP's superior inference speed through micro-event analysis, focusing on the effectiveness of LOSS compared to baseline methods.

1) Detailed Schedule Plan

To quantify the performance advantages of Intra-DP, we conduct a comparative analysis with DSCCS by recording real-time scheduling plans for ConvNeXt under varying bandwidth conditions, as shown in Fig. 15. The primary performance gain of Intra-DP stems from the parallel execution of local operations, enabling concurrent computation on the robot, data transmission, and computation on the GPU server. With the integration of Intra-DP's adaptive control mechanism, DSCCS can dynamically select the optimal layer partitioning plan based on real-time network bandwidth. Under low-bandwidth conditions, DSCCS shifts more computation to the device, trading increased energy consumption for reduced tail latency. While DSCCS degrades to device-only, Intra-DP maintains its distributed execution capability by leveraging an expanded parallel scheduling space enabled by LOP, requiring less bandwidth for GPU server acceleration. Conversely, in

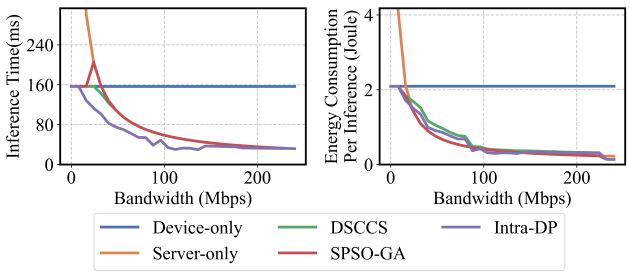


Fig. 17: Performance comparison of Intra-DP and baselines under different network bandwidth conditions.

high-bandwidth scenarios, DSCCS transitions to server-only, whereas Intra-DP leverages LOSS to optimize network and GPU server utilization for faster inference.

2) Breakdown

We also provide a detailed breakdown of each phase of the inference process throughout the ConvNeXt experiment, as illustrated in Fig. 16. In DSCCS, the transmission time accounts for up to 44.2% of the total inference time, underscoring the significant transmission bottlenecks encountered by existing layer partitioning methods, even when employing state-of-the-art strategies. Unlike layer partitioning methods, which execute each phase sequentially, Intra-DP enables the transmission phase, the robot computation phase, and the GPU server computation phase to run in parallel, resulting in faster inference times. While Intra-DP incurs additional communication overhead from its fine-grained transmission of local operators, it effectively shortens overall completion time. This is achieved by overlapping computation with communication and initiating data transfer early (Fig. 15), leading to superior performance over traditional layer partitioning methods.

C. Sensitivity Studies

1) Network Bandwidth

To systematically evaluate Intra-DP's performance under varying bandwidth conditions in MEC, we leveraged the Linux Traffic Control (tc) tool [58] over wired Ethernet to create controlled test scenarios, enabling comprehensive comparisons with baseline methods. As shown in Fig. 17, Intra-DP maintains superior performance across a wider range of network conditions, thanks to the expanded scheduling space enabled by LOP. When the bandwidth is extremely low (near 0 Mbps), Intra-DP also degrades to device-only; however, its threshold bandwidth for degradation is significantly lower than that of DSCCS. Similarly, under extremely high bandwidth conditions, Intra-DP degrades to server-only, but its degradation threshold remains much higher than DSCCS. This highlights Intra-DP's extensive parallel optimization space, enabling more efficient execution across diverse network conditions. Notably, the bandwidth thresholds at which layer partitioning methods and Intra-DP transition between device-only and server-only depend on model architecture and device computational power. Furthermore, the performance gain of Intra-DP in Fig. 12 and Fig. 14 is smaller than in Fig. 17 due to inaccurate bandwidth estimation during runtime.

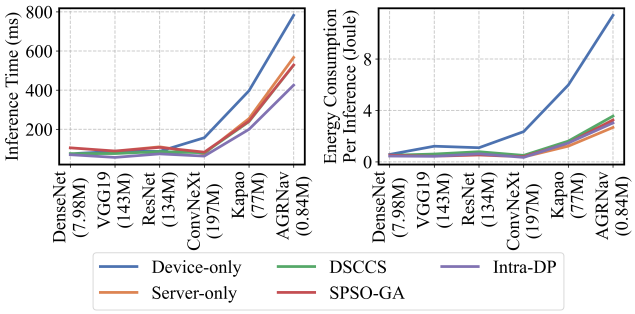


Fig. 18: Performance comparison of Intra-DP and baselines with varying model structures.

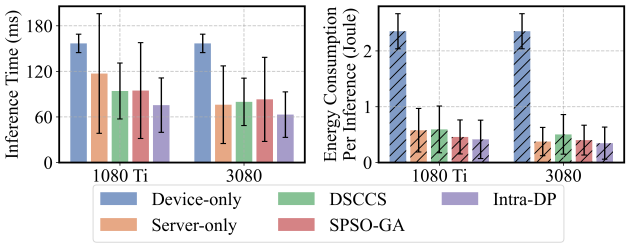


Fig. 19: Performance comparison of Intra-DP and baselines on GPU servers with varying computational power.

2) Model structure

As shown in Fig. 18, offloading methods (except device-only) achieve greater performance gains for models with higher computational demands but perform worse than device-only on DenseNet. This indicates that offloading is more beneficial for computationally intensive models, while models with lower computational demands may suffer from communication overhead, especially in bandwidth-constrained MEC systems. While Intra-DP consistently outperforms other baselines, its performance gains over them are relatively smaller for models with fewer parameters. This observation can be attributed to the fact that Intra-DP's performance improvements are primarily achieved through the parallel execution of local operations, making its effectiveness dependent on the model's size and structure. When a model has fewer parameters, the number of local operations available for parallel execution is reduced, thereby limiting Intra-DP's optimization potential for enhancing performance.

3) Computational Power

To assess Intra-DP's performance under varying computational power disparities between the robot and a GPU server, we conducted evaluations using an NVIDIA GTX 1080 Ti. As illustrated in Fig. 19, the performance advantage of offloading methods over device-only inference grows with increasing disparity. Notably, for any given computational gap, Intra-DP demonstrated the most significant improvement compared to other offloading approaches. This superior performance stems from its optimization of GPU server utilization via parallel execution of local operations. It is also pertinent to note that our experimental robot (Fig. 2a) possesses considerably higher computational power than typical mobile devices (e.g., smartphones), suggesting Intra-DP's benefits could be even more pronounced on more resource-constrained platforms.

VII. Related Work and Discussion

Limited bandwidth. Due to hardware limitations, we evaluated Intra-DP's performance only on commonly available Wi-Fi networks in robotic environments, rather than a broader set of wireless technologies (e.g., 5G). While different wireless networks vary in throughput and range due to distinct transmission protocols, they all suffer from weak connections and bandwidth fluctuations in practice [31]–[33]. Under these conditions, Intra-DP proves beneficial. When GPU servers are deployed in commercial cloud environments, network congestion and routing inefficiencies further restrict available bandwidth [59], making Intra-DP even more advantageous. Furthermore, as models scale and GPU computational power increases, Intra-DP's advantage continues to grow, further highlighting its relevance.

Inference Request Scheduling. It schedules multiple DNN inference requests to optimize overall latency and energy consumption, leveraging existing collaborative inference methods for individual executions and employing various decision algorithms to determine optimal execution locations and timing. These decision algorithms are crucial for adapting to the unique transmission and computation demands inherent in diverse MEC scenarios. For instance, such approaches are applied in industrial IoT networks with replay selection [3], space information networks with reconfigurable intelligent surfaces [60], distributed multi-node networks with integrated localization and computing of radio frequency signals [61], and heterogeneous vehicular networks with Vehicle-to-Everything communication technologies [62]. Intra-DP enhances this domain by introducing a higher-performance collaborative inference method tailored for these individual executions within MEC. Consequently, existing and future scheduling strategies can seamlessly integrate Intra-DP to capitalize on its improved efficiency for individual inference tasks, thereby boosting their overall effectiveness.

Model Compression. Quantization and model distillation are two most common model compression techniques for mobile devices. Quantization [63] reduces the precision of model weights and activations to lower computation costs, while model distillation [64] trains a smaller model to mimic a larger one with fewer resources. Unlike these methods, which trade accuracy for efficiency, collaborative inference accelerates computation by intelligently distributing workloads without compromising accuracy. Building on the significant performance gains already achieved by Intra-DP, our future work will explore how sacrificing some accuracy (e.g., quantization and early-exit policies [39]) can further enhance its efficiency.

Future Work. First, although transformer-based models are not yet commonly deployed directly on mobile devices, their increasing adoption underscores the need for continuous enhancement of Intra-DP. Addressing this, future work will focus on expanding support for a more diverse range of local operator types and on developing lightweight synchronization techniques for global operators. For instance, for operators like Softmax [43], synchronizing only the intermediate sum results instead of the entire input tensor can significantly

reduce synchronization overhead; the feasibility of exploiting such computational characteristics has been validated in FlashAttention [65]. Second, it is important to acknowledge that the system performance of Intra-DP is directly influenced by the solution quality achieved by LOSS. As achieving global optimality in non-convex optimization problems remains a formidable challenge, future research will concentrate on refining the underlying solving algorithms to enhance solution quality, thereby improving scheduling efficiency and further boosting the overall system performance of Intra-DP.

VIII. Conclusion

In this paper, we have proposed Intra-DP, a high-performance collaborative inference system optimized for MEC networks. By leveraging LOP for fine-grained parallel execution and LOSS for optimized scheduling, Intra-DP dramatically reduces transmission overhead in MEC environments through computation-communication overlap across local operations. We envision that the fast and energy-efficient inference of Intra-DP will enable diverse tasks on real-world mobile devices, driving broader adoption of advanced machine learning in mobile applications.

References

[1] F. Luo, S. Khan, Y. Huang, and K. Wu, "Binarized Neural Network for Edge Intelligence of Sensor-Based Human Activity Recognition," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1356–1368, Mar. 2023.

[2] A. N. Saridena and A. Choromanska, "DNN Patching: Progressive Fixing and Augmenting the Functionalities of DNNs for Autonomous Vehicles," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3257–3264, Apr. 2022.

[3] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A Novel Framework of Three-Hierarchical Offloading Optimization for MEC in Industrial IoT Networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5424–5434, 2019.

[4] W. McNally, K. Vats, A. Wong, and J. McPhee, "Rethinking Keypoint Representations: Modeling Keypoints and Poses as Objects for Multi-Person Human Pose Estimation," in *Proc. ECCV*, 2022, pp. 37–54.

[5] J. Wang, Z. Sun, X. Guan, T. Shen, Z. Zhang, T. Duan, D. Huang, S. Zhao, and H. Cui, "AGRNav: Efficient and Energy-Saving Autonomous Navigation for Air-Ground Robots in Occlusion-Prone Environments," in *Proc. ICRA*, May 2024.

[6] A.-Q. Cao and R. de Charette, "MonoScene: Monocular 3D Semantic Scene Completion," in *Proc. CVPR*, Jun. 2022, pp. 3991–4001.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Dec. 2017.

[8] T. Mohammed, C. Joe-Wong, R. Babbar, and M. Di Francesco, "Distributed Inference Acceleration with Adaptive DNN Partitioning and Offloading," in *Proc. INFOCOM*, Jul. 2020, pp. 854–863.

[9] H. Liang, Q. Sang, C. Hu, D. Cheng, X. Zhou, D. Wang, W. Bao, and Y. Wang, "DNN Surgery: Accelerating DNN Inference on the Edge Through Layer Partitioning," *IEEE Trans. Cloud Comput.*, May 2023.

[10] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang, "Auto-Split: A General Framework of Collaborative Edge-Cloud AI," in *Proc. KDD*, Aug. 2021, pp. 2543–2553.

[11] J. Huang, C. Samplawski, D. Ganesan, B. Marlin, and H. Kwon, "CLIO: Enabling Automatic Compilation of Deep Learning Pipelines Across IoT and Cloud," in *Proc. MobiCom*, Sep. 2020, pp. 1–12.

[12] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge," in *Proc. INFOCOM*, Apr. 2019, pp. 1423–1431.

[13] B. Yang, X. Cao, C. Yuen, and L. Qian, "Offloading Optimization in Edge Computing for Deep Learning Enabled Target Tracking by Internet-of-UAVs," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9878–9893, Jun. 2021.

[14] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge," in *Proc. ASPLOS*, Apr. 2017, pp. 615–629.

[15] H. Wu, W. J. Knottenbelt, and K. Wolter, "An Efficient Application Partitioning Algorithm in Mobile Environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.

[16] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-Efficient Offloading for DNN-Based Smart IoT Systems in Cloud-Edge Environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2021.

[17] L. Gao, J. Liu, H. Xu, S. Xu, Q. Ma, and L. Huang, "Accelerating End-Cloud Collaborative Inference via Near Bubble-Free Pipeline Optimization," *arXiv preprint arXiv:2501.12388*, Jan. 2024.

[18] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage Current: Moore's Law Meets Static Power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003.

[19] J. He, L. Li, J. Xu, and C. Zheng, "ReLU Deep Neural Networks and Linear Finite Elements," *arXiv preprint arXiv:1807.03973*, 2018.

[20] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2008.

[21] A. Ghosh, S. Iyengar, S. Lee, A. Rathore, and V. N. Padmanabhan, "REACT: Streaming Video Analytics on the Edge with Asynchronous Cloud Support," in *Proc. IoTDI*, 2023, pp. 222–235.

[22] "MLPerf Mobile Benchmarks," https://mlcommons.org/working-groups/benchmarks/mobile/, 2023.

[23] RaspberryPi, "Raspberry Pi," https://www.raspberrypi.com/documentation/computers/configuration.html\#power-consumption, 2022.

[24] NVIDIA, "Jetson Xavier NX Series: The World's Smallest AI Supercomputer," https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/, 2024.

[25] AnandTech, "AnandTech," https://www.anandtech.com/show/17587/iphone-14-battery-life-test/4, 2023.

[26] Z. Ning, M. Vandersteegen, K. Van Beeck, T. Goedemé, and P. Vandewalle, "Power Consumption Benchmark for Embedded AI Inference," in *Proc. Int. Conf. Applied Comput. WWW/Internet*. IADIS Press, Mar. 2024, pp. 3–10.

[27] NVIDIA, "InfiniBand Networking Solutions," https://www.nvidia.com/en-us/networking/products/infiniband/, 2024.

[28] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, and K. J. Barker, "Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 94–110, Jan. 2019.

[29] R. Liu and N. Choi, "A First Look at Wi-Fi 6 in Action: Throughput, Latency, Energy Efficiency, and Security," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 7, no. 1, Mar. 2023, pp. 1–25.

[30] X. Yang, H. Lin, Z. Li, F. Qian, X. Li, Z. He, X. Wu, X. Wang, Y. Liu, Z. Liao *et al.*, "Mobile Access Bandwidth in Practice: Measurement, Analysis, and Implications," in *Proc. SIGCOMM*, Aug. 2022, pp. 114–128.

[31] A. Masiukiewicz, "Throughput Comparison between The New HEW 802.11 ax Standard and 802.11 n/ac Standards in Selected Distance Windows," *Int. J. Electron. Telecommun.*, vol. 65, no. 1, pp. 79–84, 2019.

[32] M. Ding, P. Wang, D. López-Pérez, G. Mao, and Z. Lin, "Performance Impact of LoS and NLoS Transmissions in Dense Cellular Networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 2365–2380, Mar. 2015.

[33] Y. Ren, C.-W. Tung, J.-C. Chen, and F. Y. Li, "Proportional and Preemption-Enabled Traffic Offloading for IP Flow Mobility: Algorithms and Performance Evaluation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 095–12 108, Dec. 2018.

[34] "iPerf - Download iPerf3 and Original iPerf Pre-Compiled Binaries," https://iperf.fr/iperf-download.php, 2022.

[35] Y. Tian, K. Xu, and N. Ansari, "TCP in Wireless Environments: Problems and Solutions," *IEEE Commun. Mag.*, vol. 43, no. 3, pp. S27–S32, Mar. 2005.

[36] D. Narayanan, M. Shoeybi, J. Casper, P. LeGresley, M. Patwary, V. Korthikanti, D. Vainbrand, P. Kashinkunti, J. Bernauer, B. Catanzaro *et al.*, "Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM," in *Proc. SC*, Nov. 2021, pp. 1–15.

[37] Y. Zhuang, H. Zhao, L. Zheng, Z. Li, E. Xing, Q. Ho, J. Gonzalez, I. Stoica, and H. Zhang, "On Optimizing the Communication of Model Parallelism," in *Proc. MLSys*, vol. 5, May 2023.

[38] H. Wang, L. Wang, H. Xu, Y. Wang, Y. Li, and Y. Han, "PrimePar: Efficient Spatial-Temporal Tensor Partitioning for Large Transformer Model Training," in *Proc. ASPLOS*, Apr. 2024, pp. 801–817.

[39] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "SPINN: Synergistic Progressive Inference of Neural Networks Over Device and Cloud," in *Proc. MobiCom*. Association for Computing Machinery, Sep. 2020.

[40] S. Zhang, S. Zhang, Z. Qian, J. Wu, Y. Jin, and S. Lu, "DeepSlicing: Collaborative and Adaptive CNN Inference with Low Latency," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 9, pp. 2175–2187, Sep. 2021.

[41] L. Zeng, X. Chen, Z. Zhou, L. Yang, and J. Zhang, "CoEdge: Cooperative DNN Inference With Adaptive Workload Partitioning Over Heterogeneous Edge Devices," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 595–608, Apr. 2021.

[42] K. Bin, J. Park, C. Park, S. Kim, and K. Lee, "CoActo: CoActive Neural Network Inference Offloading with Fine-grained and Concurrent Execution," in *Proc. MobiSys*, Jun. 2024, pp. 412–424.

[43] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-Margin Softmax Loss for Convolutional Neural Networks," *arXiv preprint arXiv:1612.02295*, Dec. 2016.

[44] J. Yao, S. S. Kanhere, and M. Hassan, "An Empirical Study of Bandwidth Predictability in Mobile Computing," in *Proc. WiNTECH*, Sep. 2008, pp. 11–18.

[45] J. Ramapuram, F. Danieli, E. Dhekane, F. Weers, D. Busbridge, P. Ablin, T. Likhomanenko, J. Digani, Z. Gu, A. Shidani *et al.*, "Theory, Analysis, and Best Practices for Sigmoid Self-Attention," *arXiv preprint arXiv:2409.04431*, 2024.

[46] T. Nishiyama, A. Kumagai, K. Kamiya, and K. Takahashi, "SILU: Strategy Involving Large-Scale Unlabeled Logs for Improving Malware Detector," in *Proc. ISCC*, Jul. 2020, pp. 1–7.

[47] L. Sun, Z. Chen, Q. M. J. Wu, H. Zhao, W. He, and X. Yan, "AMPNet: Average-and Max-Pool Networks for Salient Object Detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 11, pp. 4321–4333, Nov. 2021.

[48] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *arXiv preprint arXiv:1608.06993*, Aug. 2018.

[49] J. Xu, Y. Pan, X. Pan, S. Hoi, Z. Yi, and Z. Xu, "RegNet: Self-Regulated Network for Image Classification," *IEEE Trans. Neural Netw. Learn. Syst.*, Aug. 2022.

[50] S. Targ, D. Almeida, and K. Lyman, "ResNet in ResNet: Generalizing Residual Architectures," *arXiv preprint arXiv:1603.08029*, Mar. 2016.

[51] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, Apr. 2015.

[52] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," *arXiv preprint arXiv:1611.05431*, Jan. 2017.

[53] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, "ConvNeXt V2: Co-Designing and Scaling ConvNets with Masked Autoencoders," in *Proc. CVPR*, Jun. 2023, pp. 16 133–16 142.

[54] R. Poorzare and A. C. Augé, "How Sufficient Is TCP When Deployed in 5G mmWave Networks over the Urban Deployment?" *IEEE Access*, vol. 9, pp. 36 342–36 355, Mar. 2021.

[55] J. Ji, K. Zhu, and L. Cai, "Trajectory and Communication Design for Cache-Enabled UAVs in Cellular Networks: A Deep Reinforcement Learning Approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6190–6204, Oct. 2023.

[56] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[57] S. Marcel and Y. Rodriguez, "Torchvision the Machine-Vision Package of Torch," in *Proc. ACM Multimedia*. Association for Computing Machinery, Oct. 2010, pp. 1485–1488.

[58] J. D. Beshay, A. Francini, and R. Prakash, "On the Fidelity of Single-Machine Network Emulation in Linux," in *Proc. MASCOTS*, 2015, pp. 19–22.

[59] M. Noormohammadpour and C. S. Raghavendra, "Datacenter Traffic Control: Understanding Techniques and Tradeoffs," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 2, pp. 1492–1525, Jun. 2017.

[60] X. Cao, B. Yang, C. Huang, C. Yuen, Y. Zhang, D. Niyato, and Z. Han, "Converged Reconfigurable Intelligent Surface and Mobile Edge Computing for Space Information Networks," *IEEE Netw.*, vol. 35, no. 4, pp. 42–48, 2021.

[61] Q. Qi, X. Chen, and C. Yuen, "Joint Offloading Selection and Resource Allocation for Integrated Localization and Computing in Edge-Intelligent Networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 8, pp. 11 427–11 440, Aug. 2024.

[62] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent Task Offloading for Heterogeneous V2X Communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[63] C. Gong, Y. Chen, Y. Lu, T. Li, C. Hao, and D. Chen, "VecQ: Minimal Loss DNN Model Compression with Vectorized Weight Quantization," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 696–710, May 2020.

[64] L. Wang and K.-J. Yoon, "Knowledge Distillation and Student-Teacher Learning for Visual Intelligence: A Review and New Outlooks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 3048–3068, Jun. 2021.

[65] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness," in *Proc. NeurIPS*, Dec. 2022, pp. 16 344–16 359.

Zekai Sun is currently working toward a PhD degree in the Systems and Networks Laboratory at the University of Hong Kong under the guidance of Prof.Heming Cui. He received Bachelor's Degree at University of Science and Technology of China (USTC) in 2020. His primary research areas include distributed systems, edge computing and systems for machine learning.

Xiuxian Guan is currently a HKU-SusTech Joint PhD student (2021Fall–Now) in Department of Computer Science, HKU and Department of Electrical & Electronic Engineering, SusTech. He received Bachelor's Degree in Department of Computer Science and Technology in University of Science and Technology of China (USTC) in 2020. His research interest includes distributed systems, edge computing and systems for machine learning.

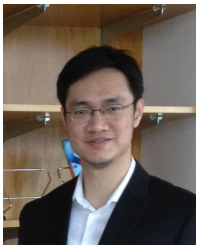
Zheng Lin received the B.Eng. degree in electronic information engineering from Fuzhou University in 2020, and the M.Eng. degree in electrical and computer engineering from Fudan University in 2023. He is currently pursuing his Ph.D. degree with the Department of Electrical and Electronic Engineering, the University of Hong Kong, Hong Kong, China. He was awarded Fudan Outstanding Graduate and Shanghai Outstanding Graduate in 2023. He serves as a Young Professional in the IEEE Vehicular Technology Society Ad Hoc Committee and as a Program Committee member for several top-tier international conferences, such as ICLR, ICML, NeurIPS, ACM MM, and AAAI. His research interests include wireless networking, edge intelligence, and distributed machine learning.

Zihan Fang received the B.Eng. degree in Electronic Science and Technology from Southeast University in 2019, and the M.Eng. degree in Electrical and Computer Engineering from Fudan University in 2023. She is currently pursuing her PhD. degree with the Department of Computer Science, the City University of Hong Kong. Her research interests include vehicle networks and distributed machine learning.

Xiangming Cai received the Ph.D. degree in communication and information systems from Xiamen University, Xiamen, China, in December 2021. From March 2022 to March 2023, he was a Research Fellow with the Engineering Product Development Pillar, Singapore University of Technology and Design, Singapore. From April 2023 to April 2024, he was a Post-Doctoral Fellow with the Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong. Since May 2024, he has been with the School of Physics and Information Engineering, Fuzhou University, China. His research interests include wireless communications and underwater acoustic communications. He was a recipient of the Best Paper Award of the 2021 IEEE Asia–Pacific Conference on Communications. He was also a recipient of the Exemplary Reviewer of IEEE TRANSACTIONS ON COMMUNICATIONS.

Zhe Chen is an assistant professor within the School of Computer Science at Fudan University, and the Co-Founder of AIWiSe Ltd. Inc. He obtained his Ph.D. degree in Computer Science from Fudan University, China, with a 2019 ACM SIGCOMM China Doctoral Dissertation Award. Before joining Fudan University, he worked as a research fellow in NTU for several years, and his research achievements, along with his efforts in launching products based on them, have thus earned him 2021 ACM SIGMOBILE China Rising Star Award recently.

Fangming Liu received the B.Eng. degree from the Tsinghua University, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a Full Professor at the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, data center and green computing, SDN/NFV/5G, and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, the First Class Prize of Natural Science of the Ministry of Education in China, as well as the Second Class Prize of National Natural Science Award in China.



Heming Cui is an Associate Professor in Computer Science of HKU. His research interests include operating systems, programming languages, distributed systems, and cloud computing, with a particular focus on building software infrastructures and tools to improve reliability and security of real-world software. He is a member of IEEE.

Jie Xiong is an Associate Professor in the College of Computing and Data Science at Nanyang Technological University. He was previously a Principal Researcher at Microsoft Research Asia and an Associate Professor at the University of Massachusetts Amherst. He earned his Ph.D. from University College London, M.S. from Duke University and Bachelor from Nanyang Technological University. His Ph.D. was supported by the Google European Doctoral Fellowship, and he was the Runner-Up for the 2016 British Computer Society Distinguished Dissertation Award. Jie's research interests include wireless sensing, mobile computing and smart health. His work has received numerous awards, including the SIGMOBILE 2024 Test-of-Time Award, MobiCom 2024 Best Paper Award, SenSys 2022 Best Paper Award, SECON 2022 Best Paper Award, UbiComp (IMWUT) 2021 Distinguished Paper Award, CoNEXT 2014 Best Paper Award, and multiple Best Paper Runner-Ups at MobiCom.

Wei Ni (Fellow, IEEE) received the B.E. and Ph.D. degrees in Electronic Engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively. He is a Senior Principal Research Scientist at CSIRO, Sydney, Australia. He is also a Conjoint Professor at the University of New South Wales. He was a Postdoctoral Research Fellow at Shanghai Jiaotong University from 2005 – 2008; Deputy Project Manager at the Bell Labs, Alcatel/Alcatel-Lucent from 2005 to 2008; and Senior Researcher at Devices R&D, Nokia from 2008 to 2009. He has coauthored three authored books, eleven book chapters, more than 300 journal papers, 100 conference papers, 27 patents, and ten standard proposals accepted by IEEE. His research interests include machine learning, online learning, stochastic optimization, and their applications to system efficiency and integrity.

Dr. Ni has been an Editor for IEEE Transactions on Wireless Communications since 2018, and an Editor for IEEE Transactions on Vehicular Technology since 2022, an Editor for IEEE Transactions on Information Forensics and Security and IEEE Communication Surveys and Tutorials since 2024, and an Editor for IEEE Transactions on Network Science and Engineering since 2025. He served first as the Secretary, then the Vice-Chair and Chair of the IEEE VTS NSW Chapter from 2015 to 2022, Track Chair for VTCSpring 2017, Track Co-chair for IEEE VTC-Spring 2016, Publication Chair for BodyNet 2015, and Student Travel Grant Chair for WPMC 2014.

Chau Yuen (Fellow, IEEE) received the B.Eng. and Ph.D. degrees from Nanyang Technological Univer sity, Singapore, in 2000 and 2004, respectively.

He was a Post-Doctoral Fellow with Lucent Tech nologies Bell Labs, Murray Hill, in 2005. From 2006 to 2010, he was with the Institute for Infocomm Research, Singapore. From 2010 to 2023, he was with the Engineering Product Development Pillar, ingapore University of Technology and Design. Since 2023, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University. He is currently the Provost's Chair of Wireless Communications and the Assistant Dean with the Graduate College. He has four U.S. patents and has published over 400 research articles at international journals.

Dr. Yuen received the IEEE Communications Society Leonard G. Abra ham Prize (2024), the IEEE Communications Society Best Tutorial Paper Award (2024), the IEEE Communications Society Fred W. Ellersick Prize (2023), the IEEE Marconi Prize Paper Award in Wireless Communications (2021), the IEEE APB Outstanding Paper Award (2023), and the EURASIP Best Paper Award for EURASIP Journal on Wireless Communications and Networking (2021). He serves as the Editor-in-Chief for Nature Computer Science (Springer) and an Editor for IEEE Transactions on Vehicular Technology, IEEE Systems Journal, and IEEE Transactions on Network Science and Engineering, where he received the IEEE TNSE Excellent Editor Award and was recognized as the Top Associate Editor for TVT from 2009 to 2015. He has also served as a Guest Editor for several special issues, including IEEE Journal on Selected Areas in Communications, IEEE Wireless Communications Magazine, IEEE Communications Magazine, IEEE Vehicular Technology Magazine, IEEE Transactions on Cognitive Communications and Networking, and Applied Energy (Elsevier). He is a Distinguished Lecturer of the IEEE Vehicular Technology Society, listed among the Top 2% Scientists by Stanford University, and a Highly Cited Researcher.