

DIPaaS: Distributed Inference Planning as a Service for the Heterogeneous AI-Native Edge

Washington R. D. Silva, Luciano S. Fraga, Andrew Williams, Omer Rana, Luiz F. Bittencourt, Kleber V. Cardoso, Fábio L. Verdi

Abstract—Realizing the vision of 6G and AI-native networks demands the deployment of compute-intensive Deep Learning (DL) workloads at the edge to guarantee millisecond-scale latency. Executing such workloads at the edge, which is highly heterogeneous, comprising traditional ASICs and AI-capable programmable network devices, necessitates distributing the inference processing to circumvent strict computing resource constraints. This architectural diversity creates a vast design space that is economically infeasible to explore through physical benchmarking, leaving network operators to struggle with accurate capacity planning and the risk of over-provisioning. In this paper, we propose the Distributed Inference Planning as a Service (DIPaaS) paradigm. By translating models into Directed Acyclic Graphs (DAGs) and mapping them to hardware profiles, this service creates a highly modular search space for exploring diverse what-if deployment scenarios, allowing infrastructure providers to vary AI models, device specifications, application constraints, and optimization strategies. We evaluated DIPaaS using a Mobile Augmented Reality (MAR) use case, exploring how different deployment scenarios and partitioning strategies affect workload scheduling. Our evaluation reveals that while constructive heuristics fail, exact and list-based methods successfully find viable deployment plans. Ultimately, the service’s modular architecture provides a broader strategic benefit: identifying feasible schedules that satisfy rigorous latency targets without relying on specialized accelerators, directly addressing the economic challenges of edge infrastructure dimensioning.

Index Terms—Model Partitioning, Distributed Inference, Resource Allocation, Heterogeneous Devices, Object Detection.

I. INTRODUCTION

THE roadmap to 6G is being defined by the convergence of connectivity and intelligence. As the network transforms into an Artificial Intelligence (AI)-Native platform, it must support data-intensive and ultra-low-latency applications powered by complex Deep Neural Networks (DNNs). These workloads typically require a millisecond-scale response time, which entails the migration of AI inference from the core to the edge of the network [1]. However, the computing infrastructure on the edge is undergoing a paradigm shift. Advances in programmable network devices, such as SmartNICs, Data Processing Units (DPUs), and Field Programmable

Gate Arrays (FPGAs), are enabling this hardware class to move beyond packet forwarding to execute application logic directly in the data plane, including Machine Learning (ML) workloads [2]. This revolution within the network fabric and the increasing innovation of Application-Specific Integrated Circuits (ASICs) [3] converge to a new hybrid landscape: an edge server is no longer just a CPU-GPU host, but a heterogeneous cluster where computing capabilities are also integrated into the network fabric [4].

While the feasibility of running ML inference on these network devices is well-proven, they remain compute-constrained compared to dedicated accelerators [2]. Consequently, utilizing this heterogeneous environment to execute compute-intensive DNNs necessitates a distributed inference strategy. Rather than running only a monolithic model on a single device, the DNN may be partitioned into discrete components and distributed across the available infrastructure. Thus, each device may compute a fragment compatible with its architectural and computing constraints, performing the inference computation collaboratively [2].

However, realizing distributed inference across a hybrid infrastructure introduces a highly complex design space exploration problem. The myriad of hardware and model architectures available raises an intractable number of deployment configurations, shifting the fundamental challenge faced by infrastructure providers from “How do we route traffic?” to “How do we provision computing?”. To effectively harness this paradigm, providers must evaluate diverse what-if scenarios by assessing different model partitioning strategies (e.g., optimizing for minimum latency versus energy efficiency), hardware combinations, and network topologies. Since physically assembling and benchmarking every potential configuration requires prohibitive upfront investments, analytical performance modeling serves as a fundamental enabler for distributed inference planning. Yet, current literature predominantly focuses on performance modeling for large-scale, homogeneous cloud systems [3], [5], [6].

Although distributed edge inference has received considerable attention in recent years, surveys spanning from earlier [7] to contemporary [8] reviews show that existing literature has not evolved to encompass edge infrastructures that integrate programmable network devices into the inference path, leaving distributed inference planning for this multi-architectural landscape as an unaddressed research gap [9]. Hence, to systematically navigate this complex design space, we introduce the *Distributed Inference Planning as a Service (DIPaaS)* paradigm. Rather than relying on costly and exhaustive physical benchmarking, DIPaaS provides a cru-

W. R. D. Silva and F. L. Verdi are with the Department of Computer Science, Federal University of São Carlos (UFSCar), Sorocaba, SP, 18052-780, Brazil. Email: washingtonrds@estudante.ufscar.br.

L. S. Fraga and K. V. Cardoso are with the Institute of Informatics, Universidade Federal de Goiás (UFG), Goiânia, GO, 74690-900, Brazil.

L. F. Bittencourt is with the Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, 13083-970, Brazil.

A. Williams is with Ericsson Research, Ericsson, Kista, Stockholm, 164 83, Sweden.

O. Rana is with the School of Computer Science and Informatics, Cardiff University, Cardiff, CF24 4AG, United Kingdom.

cial abstraction layer that fundamentally decouples the AI deployment logic from the underlying heterogeneous compute fabric. This architectural shift allows infrastructure providers to explore the aforementioned what-if scenarios, analytically verifying deployment plans by simultaneously partitioning the model and projecting its performance before a single device is acquired or deployed.

To achieve this, DIPaaS translates the target DNN into a Directed Acyclic Graph (DAG) and estimates the computational demand (in Floating Point Operations) of each node. Simultaneously, it profiles hardware specifications to characterize processing throughput, memory limits, bandwidth constraints, and energy consumption. By combining these attributes with optimization methods, the framework replaces physical testing with an analytical model that balances multiple trade-offs (energy, latency, and load distribution) to determine a viable workload partitioning scheme that satisfies the application's latency requirements. Consequently, this modeling framework enables infrastructure providers to decompose AI models, analytically project performance for both available and prospective hardware, and verify whether Service Level Agreements (SLAs) can be met without costly over-investment and over-dimensioning.

Hence, the primary contribution of this work is a unified deployment planning methodology for distributed inference across the emerging hybrid edge, where programmable network devices coexist alongside standard CPUs and GPUs. By leveraging the DAG intermediate representation (IR), DIPaaS enables device interoperability, generalization across AI architectures, and analytical estimation without physical benchmarking or trained performance predictors, allowing comprehensive deployment analysis prior to hardware procurement.

The remainder of this article is organized as follows. Section II details the challenges of distributed inference for the AI-Native edge. Section III introduces the model partitioning methodology underlying the deployment planning engine. Section IV presents a comprehensive case study demonstrating DIPaaS's utility in dimensioning resources for Mobile Augmented Reality (MAR) applications. Finally, Sections V and VI discuss future research directions and conclude the article, respectively.

II. NAVIGATING THE AI-NATIVE LANDSCAPE

While previous generations of cellular networks focused on broadband connectivity, the transition toward 6G is defined by a fundamental architectural shift: the AI-Native network. This refinement process is already underway. While 3GPP Releases 16 and 17 enhanced the 5G baseline, Release 18 marked the pivotal shift to 5G-Advanced by introducing native ML support for network optimization. Building on these advancements, 6G aims to complete the transition by evolving into a fully AI-Native architecture where network functions incorporate AI capabilities by design. This inherent intelligence does not merely optimize performance; it unlocks a new class of applications that previous generations could not fully support, e.g., Integrated Sensing and Communications (ISAC) and Space-Air-Ground-Sea integration [9], [10].

Realizing this AI-Native vision requires the infrastructure to satisfy two distinct but equally critical demands: supporting time-sensitive user applications and executing the intrinsic intelligence of the network itself. This combined pressure necessitates a fundamental migration of compute from the core to the edge to satisfy stringent latency constraints. The transition to this future landscape is already being charted, with 3GPP Release 18 establishing the technical framework for AI/ML in the wireless air interface to target performance gains in critical use cases like Beam Management (BM) and Channel State Information (CSI) feedback [11]. In these scenarios, the inference engine cannot be decoupled from the data source; decisions must be executed directly at the Radio Unit (RU) or distributed across the adjacent compute continuum to maintain link stability [10].

Although distributing these compute-intensive workloads mitigates individual hardware limitations, it introduces profound operational complexity. The fundamental challenge lies in systematically determining the scheduling order, data routing, and optimal placement of every inference subtask across a heterogeneous compute fabric [9]. Consequently, fulfilling these operational demands necessitates a shift from reactive provisioning to analytical pre-deployment planning. Such an approach is essential to guarantee both end-to-end latency and energy efficiency before committing physical resources [11].

III. DISTRIBUTED INFERENCE PLANNING AS A SERVICE

To realize the aforementioned pre-deployment planning, we introduce the DIPaaS architecture. This paradigm inherently decouples the deployment logic from specific hardware availability. As depicted in Fig. 1, the service achieves this abstraction by constructing a multidimensional search space, illustrated as a cube. In this representation, six key parameters act as modular inputs that effectively fill the segments of this structure:

- **AI Models:** Accepts pre-trained DNNs (e.g., YOLO) exported from standard frameworks (e.g., PyTorch, TensorFlow), requiring no manual adaptation for compatibility prior to deployment planning.
- **Device Specifications:** Integrates standard hardware specifications (derived from datasheets, e.g., theoretical throughput, memory capacity, and thermal limits), eliminating the need to physically benchmark devices to measure inference performance.
- **Network Topologies:** Specifies the intended communication fabric (e.g., available bandwidth and baseline transmission latencies between edge servers), eliminating the need to deploy and configure network infrastructure to evaluate data transfer bottlenecks.
- **Application Constraints:** Defines the strict SLA required for the target task (e.g., a maximum allowable end-to-end latency of 15 ms), functioning as a formal constraint within the partitioning model to guarantee that the distributed inference satisfies the application requirements.
- **Resource Contention:** Applies specific background utilization scenarios for devices (e.g., simulated stress conditions or full resource availability), allowing the service

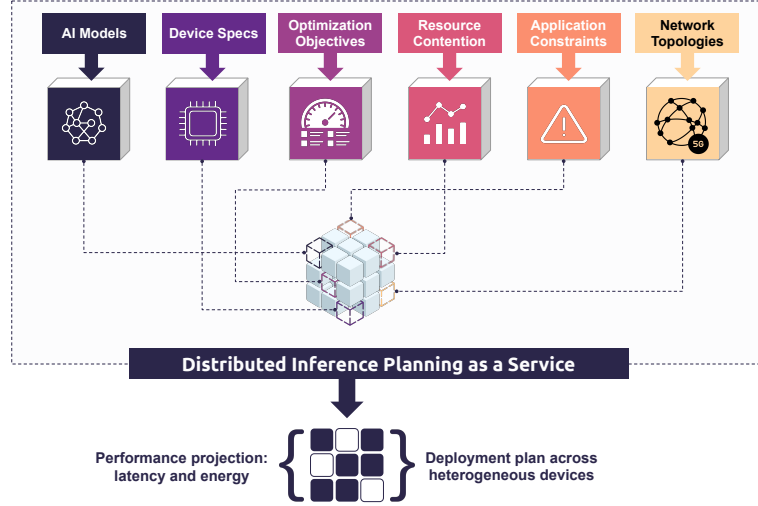


Fig. 1: DIPaaS overview. The service receives modular inputs to construct a multidimensional search space (the central cube). It then resolves these inputs to generate optimized partitioning schemes with verifiable performance metrics (the 2D grid).

to estimate the residual resources available for distributed inference.

- **Optimization Objectives:** Selects the target metric to minimize (e.g., total energy consumption, overall cost, or inference latency) and the underlying search method (ranging from fast greedy heuristics to comprehensive single- or multi-stage exact models), allowing infrastructure providers to analyze specific deployment goals.

By integrating these six dimensions, we introduce the first modular pre-deployment planning approach for evaluating distributed inference across the heterogeneous hardware spectrum of the AI-native edge. This dynamic construction allows the service to represent any infrastructure scenario and seamlessly support diverse use cases, ranging from internal network intelligence tasks to future user-centric applications. To translate these modular inputs into resource-aware partitioning schemes, DIPaaS employs an optimization engine that explores this multidimensional search space, guided by a target objective (e.g., minimizing latency or energy) and constrained by the application SLA. The service then generates the resulting scheme, specifying partition-to-device assignments and projecting energy and latency metrics to guide deployment decisions. This outcome is visually represented by the filled blocks of the 2D allocation grid in Fig. 1. Next, we detail how these inputs are integrated within the DIPaaS pipeline.

Workload Profiling and Resource Allocation

To explore the search space defined in the previous section, DIPaaS characterizes the interaction between the AI model and edge resources through a systematic five-step pipeline. As detailed in Fig. 2, this workflow correlates the DNN’s structural properties with hardware capabilities to produce a unified performance profile. This profile projects latency and energy costs, thereby bridging computational demand with the physical infrastructure.

The pipeline begins by converting the pre-trained DNN into an IR using the Open Neural Network Exchange (ONNX)

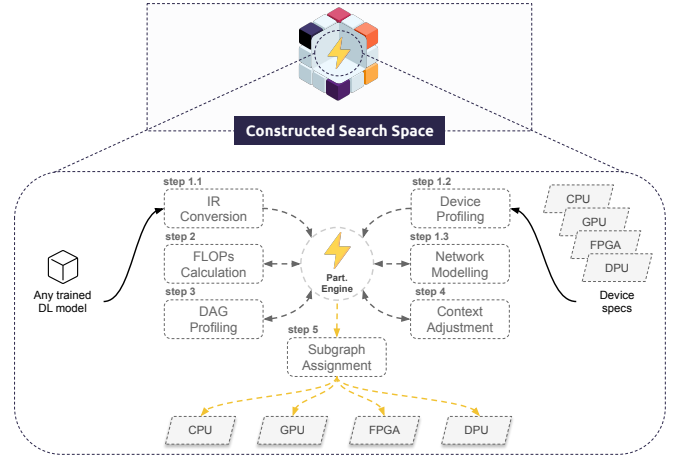


Fig. 2: The DIPaaS workflow. The service first abstracts the AI model and edge infrastructure into a unified performance profile. These baseline metrics are then adjusted for operational context, enabling the partitioning engine (Part. Engine) to derive a suitable deployment strategy that satisfies global latency constraints.

format (Step 1.1), in which the model is inherently structured as a directed acyclic graph (DAG). In this representation, each node defines a specific layer-level operation (such as a convolution), while the connecting edges denote the multi-dimensional data arrays (tensors) flowing between them. Concurrently, the service synthesizes the hardware specifications to encode a comprehensive infrastructure profile (Step 1.2). This profile captures critical device metrics, including processing throughput (GFLOPS), thermal design power (TDP), memory capacity, and internal bandwidths (memory and PCIe). To complete this physical abstraction, the system simultaneously models the network topology (Step 1.3), quantifying the transmission latency and available bandwidth between resources distributed at the edge.

With the infrastructure and model abstracted, DIPaaS de-

termines the computational demand of each task within the DAG (Step 2). Instead of relying on physical benchmarking, the service parses the structural attributes of every node, such as kernel dimensions, stride, and channel counts. By analyzing these parameters, it calculates the exact number of Floating Point Operations (FLOPs) required to execute each layer and its corresponding memory footprint. This profiling is conducted on an optimized DAG, ensuring these estimations account for graph optimizations performed by ONNX Runtime prior to inference execution¹ (e.g., fusing convolution and batch normalization nodes).

To translate these computational demands into projected performance metrics (Step 3), DIPaaS combines the DAG profile with the infrastructure profile to construct a composite latency model. First, the raw computation time is estimated by the ratio between each node's FLOP count and each device's processing throughput. Then, this computing time is aggregated with data movement latency. Specifically, this communication overhead is determined by dividing tensor sizes by the applicable bandwidth constraint: device memory for intra-device execution (reading input tensors, weights, and biases, and writing the resulting output), PCIe for inter-device transfers within the same server (transferring data across distinct computing units), and network bandwidth for cross-server distribution.

With the total execution duration established, the service then projects the corresponding energy footprint by scaling it against the target device's TDP. While TDP does not capture dynamic runtime power fluctuations, it serves as a conservative proxy for theoretical power consumption at peak performance [12]. Furthermore, TDP is the only energy-related metric available on vendor datasheets, making it essential for deployment planning when physical hardware is not yet acquired. Ultimately, these estimates compose the comprehensive performance profile that quantifies the expected execution time, communication overhead, and energy consumption for mapping any specific DAG node to any available edge device.

With the baseline costs established, DIPaaS integrates the pre-defined resource contention parameters (Step 4) to model specific operational contexts. To achieve this, the service inversely scales the nominal latency by the target device's available computational capacity (calculated as one minus its specified background utilization rate). The nominal energy footprint is then proportionally increased to account for this extended execution duration. Concurrently, it retrieves the global latency budget specified within the application constraints. This acts as a strict SLA, establishing the absolute boundary conditions that the partitioner must respect when distributing the workload.

Finally, the partitioning engine (Step 5) leverages the performance profile to derive a feasible deployment strategy. To this end, we formulate the resource allocation task as an optimization problem aimed at minimizing the specified objective, such as total energy consumption or inference latency. The engine then employs the designated search method to navigate

the resulting search space. Both the optimization goal and the search method are defined during the initial configuration phase. This exploration is strictly bounded by three critical constraints: the precedence dependencies of the DAG, the application's global latency budget, and the available memory capacity of each target device.

Within this framework, the partitioning engine determines a suitable splitting granularity by evaluating the ratio between available computing capacity and workload complexity. Accordingly, individual DAG nodes are systematically mapped to specific resources. This approach leverages the ONNX ecosystem to provide three key advantages for pre-deployment planning. First, it ensures device interoperability, allowing distinct hardware platforms to execute different partitions of the model without compatibility issues [13]. Second, it enables model-agnostic generalization, as the profiling and partitioning logic operates on standard graph structures provided by ONNX. Third, by analytically deriving computational demand directly from node-level structural attributes, DIPaaS bypasses the need for both physical hardware benchmarking and ML-based performance predictors. On the infrastructure side, devices and network topologies are modeled by parameterized inputs (datasheet specifications and link characteristics), so extending the analysis to new hardware or network configurations requires no modification to the pipeline itself.

IV. EXPERIMENTAL EVALUATION

To evaluate the proposed paradigm, we employed DIPaaS in a Mobile Augmented Reality (MAR) scenario. MAR is a key user-centric application that illustrates the conflict between device resource limitations and strict application demands: it relies on compute-intensive DL models for real-time object detection, while requiring a stringent 20 ms end-to-end latency to ensure optimal Quality of Experience (QoE) [10], [14]. Crucially, this deadline acts as a strict feasibility constraint rather than a minimization objective; thus, any deployment plan meeting this threshold is deemed valid, allowing the partitioning engine to optimize for the primary objectives defined in the search space.

Hence, we validated the service's scheduling capabilities with two fundamental models for MAR: YOLOv11 nano and ResNet-18. To split these models, we investigated two approaches: an exact solver (bounded by a 24-hour runtime or a relative optimality gap less than 1%) to establish a theoretical baseline, and heuristics to reduce solution times – specifically, the state-of-the-art Heterogeneous Earliest Finish Time (HEFT) algorithm [15], and two greedy allocation strategies designed to minimize energy (heu_es) and completion time (heu_ct). To assess these approaches, we defined two distinct deployment scenarios, incorporating contention rates on computing and communication (intra- and inter-device). These rates were not derived from empirical benchmarks, but rather assumed as synthetic stress parameters to simulate congested edge environments.

1) *Standalone Edge Server*: Scenario 1 represents a single edge server equipped with diverse accelerators connected via PCIe: an Nvidia A30 (NV-A30) GPU, an Nvidia BlueField 2

¹Graph optimizations are assumed to be performed offline, so optimization overhead is excluded from latency estimations.

(NV-BF2) DPU, and an Alveo U55C FPGA configured with 2 Deep Learning Processing Units (AL-U2). To model a shared infrastructure (Step 4, Fig. 2), we restricted the GPU to 30% of its available capacity while keeping the DPU and FPGA fully available. Using the YOLOv11 nano model, we evaluated the trade-off between energy consumption (static and dynamic) and end-to-end latency. In this context, latency accounts for computation, transmission, and synchronization overhead (idle gaps). Since these gaps delay task completion and tie up shared resources, minimizing them is vital in a shared scenario. To deal with this conflict, we employed the exact solver with two hierarchical optimization stages: Stage 1 minimizes total energy within the latency deadline (1-stage), while Stage 2 minimizes the execution makespan to eliminate these idle gaps (2-stage), preserving the energy baseline established in Stage 1.

2) *Disaggregated Edge Servers*: While Scenario 1 focused on maintaining strict latency targets despite a saturated GPU, Scenario 2 investigates a more fundamental challenge: assessing the feasibility of executing the ResNet-18 model on an edge infrastructure completely devoid of a GPU. To simulate this highly constrained environment, we shifted from the standalone configuration of Scenario 1 to a disaggregated architecture comprising three distinct servers. Each server is provisioned with a dedicated computing unit: Server 1 features a low-power multi-core Intel Xeon D-2832NT CPU (Xeon-NT); Server 2 hosts an Alveo U55C FPGA configured with a single Deep Learning Processing Unit (AL-U1); and Server 3 contains an Nvidia BlueField-2 (NV-BF2) DPU. Since this topology requires intermediate inference results to be routed across the network, we introduced a 40% network contention rate to assess DIPaaS’s adaptability to constrained bandwidth. Furthermore, we applied a 60% compute contention rate across the CPU (OS and orchestration overheads), DPU, and FPGA (data plane processing). Finally, to guarantee execution feasibility under these severe resource limitations, we added an initial load-balancing stage (3-stage) to prevent any compute unit from exceeding its resource capacity. This preceding step ensures a feasible search space for the energy and makespan optimizations defined in Scenario 1.

A. Experimental Results

To account for the practical divergence between nominal datasheet metrics and real-world execution, we conducted a sensitivity analysis by scaling estimated execution times (encompassing both computation and data movement) and energy costs across multiple degradation factors (1.25, 1.50, 1.75, 2.00). Specifically, all experimental results reported in this section correspond to the worst-case feasible degradation bounds: 2.00 for Scenario 1 and 1.75 for Scenario 2, representing 100% and 75% penalties over baseline performance estimations, respectively.

Figures 3 and 4 illustrate the workload and energy footprint for Scenario 1, respectively, grouping the evaluated scheduling strategies by target device. As noted in Section III, the energy estimate is derived from a simplified model based on TDP scaling, capturing energy proportional to the application’s

infrastructure use but excluding full-system overheads (further discussed in Section V.C). Notably, the exact solver reached the gap threshold in both stages, taking 5 seconds to find a feasible solution for Scenario 1 and 53 minutes for Scenario 2, demonstrating that as the deployment complexity increases (e.g., model architecture and operational constraints), finding an optimal partition becomes computationally prohibitive.

These results show that while *heu_ct* assigned most of the workload to the GPU, the exact solver, *heu_es*, and HEFT allocated the majority of operations to the FPGA (Fig. 3). Consequently, 1-stage, 2-stage and *heu_es* exhibited an identical energy footprint, closely followed by HEFT (Fig. 4). In contrast, *heu_ct* incurred the highest energy cost (1.78 J) by additionally assigning a marginal fraction of the workload to the DPU. Regarding the 20 ms latency SLA, both the exact solver and HEFT met the constraint. Specifically, the second optimization stage compressed the makespan to 18.8 ms, while HEFT completed it in 15.7 ms, as it operates without the strict energy constraints of the multi-stage formulation. However, by not accounting for complex DAG dependencies and inter-device communication overhead, *heu_es* and *heu_ct* failed to meet the SLA, taking 29.8 ms and 36.3 ms, respectively.

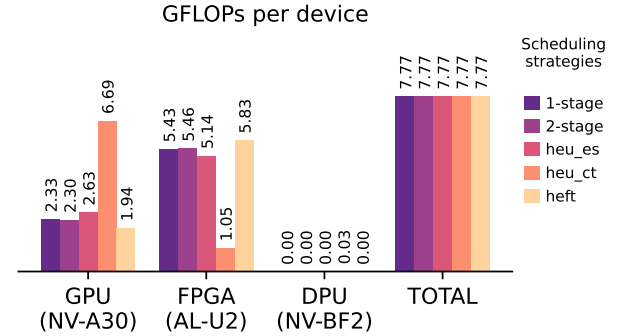


Fig. 3: Estimated workload distribution of YOLOv11 nano.

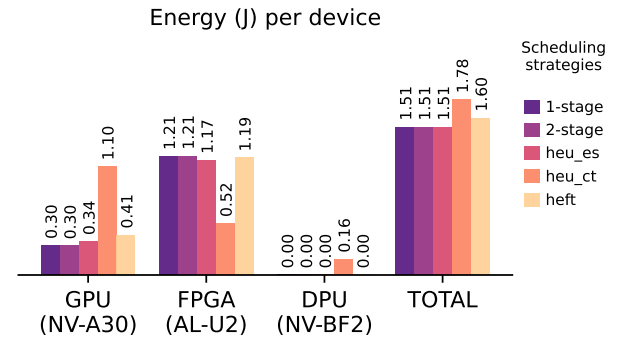


Fig. 4: Estimated energy footprint of YOLOv11 nano.

In Scenario 2, while the exact solver utilized multiple units to maintain system viability, it allocated 3.60 GFLOPs of the total 3.64 GFLOPs to the FPGA and assigned the remainder to the CPU (Fig. 5), resulting in a total energy footprint of 1.97 J. This heavy reliance on the FPGA occurs because 37 of the 40 nodes in the ResNet-18 DAG lie on the critical path. Combined with the strict 20 ms SLA and severe hardware contention, this topological bottleneck limits inter-device parallelism and

prevents a more distributed execution. Similarly to Scenario 1, the final optimization stage successfully compressed the makespan (from 20 to 19.8 ms). However, due to the exact solver’s three-stage hierarchical architecture, this makespan minimization is strictly bounded by the load-balancing and energy-minimization profiles locked in during the first two stages. This prevents the solver from exploring schedules that might offer lower latency at the cost of load imbalance or a marginal energy increase. A crucial insight from this data is the comparable total energy consumption between Scenarios 1 (1.51 J) and 2 (1.97 J), which shows that by balancing the workload according to hardware capabilities, a strict SLA can be met even under restricted conditions, eliminating the need for costly dedicated GPUs in exchange for a modest energy trade-off. Similarly, all three heuristics produced highly concentrated schedules (Fig. 5). By allocating all nodes exclusively to the FPGA, HEFT managed to match the exact solver’s 19.8 ms makespan; however, as in Scenario 1, the greedy approaches failed to satisfy the SLA constraint (20.9 ms for heu_es and 20.4 ms for heu_ct).

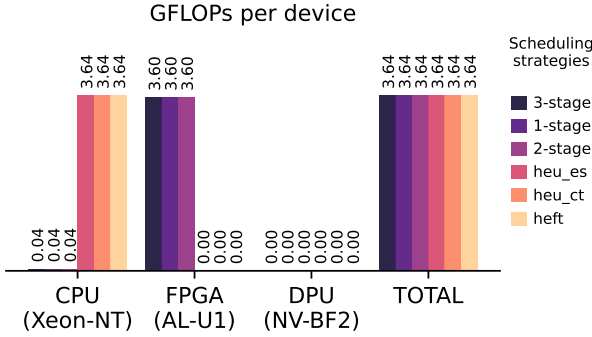


Fig. 5: Estimated workload distribution of ResNet-18.

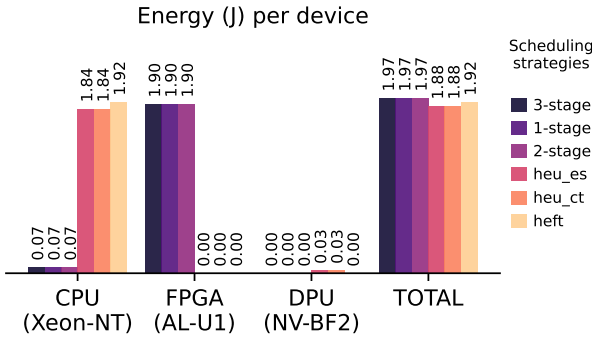


Fig. 6: Estimated energy footprint of ResNet-18.

Overall, these results confirm the proposed approach’s effectiveness in guiding hardware provisioning before deployment. Crucially, the evaluation highlights that the ideal scheduling approach is highly context-dependent. In balanced environments such as Scenario 1, HEFT provided an exceptional trade-off between makespan and energy consumption. Conversely, in highly constrained setups such as Scenario 2, the multi-stage optimization approach proved necessary to successfully guarantee SLA compliance. It successfully navigates the severe architectural limitations that otherwise

restrict execution options and force concentrated allocations. Furthermore, this hierarchical refinement offers a clear operational benefit: reducing the makespan directly increases server availability for subsequent requests. Even when computational limits prevent the solver from reaching absolute mathematical optimality within practical timeframes, it remains highly valuable as a theoretical baseline to verify the feasibility of a given scenario and to benchmark the quality of fast heuristics. It is worth noting, however, that these scenarios serve as an illustrative use case. Due to DIPaaS’s modular nature, scheduling strategies can be seamlessly interchanged to explore diverse what-if scenarios. As our energy footprint findings demonstrate, this flexibility is vital for uncovering critical operational advantages – such as meeting strict SLAs without a dedicated GPU – before making costly hardware procurement decisions. Finally, while the heuristics’ millisecond-scale execution times (in both scenarios) suggest promising potential for on-the-fly workload splitting, model redeployment overhead remains a challenge for applications with strict SLAs, such as MAR.

V. TAKEAWAYS AND OPEN CHALLENGES

While this study demonstrates the DIPaaS paradigm, we recommend the following research directions to guide future work toward a robust and fully autonomous edge ecosystem:

A. Architectural Semantics: Currently, DIPaaS treats each DAG node as a discrete unit. While flexible, this approach lacks the semantic awareness needed to preserve indivisible structures such as Transformer attention blocks and residual connections. We recommend transitioning to architecture-aware partitioning to group tightly coupled nodes, avoiding memory and synchronization bottlenecks across high-latency interconnects. This refined abstraction also exposes new pipeline parallelism opportunities, which are crucial for efficiently distributing compute- and memory-intensive workloads such as Large Language Models (LLMs), a key enabler for the future AI-native edge.

B. Operational Dynamics and Network Volatility: While DIPaaS incorporates a resource contention variable for multi-tenant interference, its current communication model assumes a relatively stable network state. In practice, volatility (jitter, packet loss, throughput variability) causes transfer times to deviate from static estimates, introducing uncertainties to the scheduling process. We recommend enhancing DIPaaS with dynamic network profiles and stochastic link quality modeling to generate resilient distribution strategies despite shifting communication performance.

C. System-Wide Sustainability and Carbon Accounting: Our current energy model focuses on isolated compute units. In production environments, operational costs also encompass host idle power, networking, and cooling systems. Hence, we recommend expanding DIPaaS’s energy profiling to cover this full-system footprint. Additionally, as sustainability becomes a critical metric for infrastructure providers, future work could also integrate hardware-specific carbon-per-FLOP signatures into DAG profiles to allow workload assignments to greener devices.

D. DIPaaS as an Edge Infrastructure Digital Twin: We envision DIPaaS evolving into a comprehensive Digital Twin of the heterogeneous edge. In this role, DIPaaS would enable infrastructure providers to perform risk-free, what-if analyses for high-concurrency inference pipelines and evaluate hardware provisioning strategies before committing physical resources. By synchronizing with live telemetry, future work can transition DIPaaS into an autonomous closed-loop orchestrator that recalculates deployment plans and migrates workloads during hardware failures or environmental shifts, ensuring the edge infrastructure operates at its theoretical optimum despite real-world volatility.

E. Robust Approximation via Meta-Heuristics: Although exact optimization provides a valuable reference, its computational complexity increases combinatorially with the number of DNN partitions, heterogeneous devices, and communication links. Consequently, exact solvers are best suited for offline infrastructure planning and baseline evaluation rather than real-time orchestration. For instance, our ResNet-18 scenario required nearly 30 minutes to reach the target optimality gap, indicating that larger foundation models and richer infrastructures will demand significantly longer solve times. In such scenarios, approximation techniques offer a more practical alternative, with the exact model serving as a quality benchmark. However, realizing this requires balancing a trade-off between execution time and solution quality. Our evaluation highlighted that while greedy heuristics can execute in milliseconds, their performance varies across complex DAG structures and hardware topologies. Hence, we recommend future work to explore robust meta-heuristics (e.g., Genetic Algorithms, Simulated Annealing), which offer a practical compromise when exact solutions are computationally prohibitive and greedy approximations fall short of performance targets.

VI. CONCLUSION

As the roadmap to 6G transforms the network into an AI-Native platform, bridging the gap between compute-intensive workloads and constrained edge infrastructure has become a fundamental necessity. In this work, we address this *resource paradox* by introducing the DIPaaS paradigm. By shifting from physical benchmarking to systematic performance modeling, it empowers infrastructure providers to navigate the vast design space of ASICs and programmable network devices without the economic burden of hardware over-provisioning.

The service's modularity allows it to simultaneously evaluate diverse model architectures and heterogeneous hardware configurations, delivering two critical outcomes: it generates feasible deployment plans that satisfy strict performance requirements, and it provides comprehensive latency and energy projections, allowing providers to optimize hardware investments prior to acquisition. While DIPaaS currently enables the analytical dimensioning of edge resources and rigorous SLA verification, it has the potential to evolve well beyond initial planning, ultimately serving as a foundational tool for architecting the high-performance, resilient, and intelligent edge networks of the future.

ACKNOWLEDGMENTS

This work was supported by Ericsson Telecomunicações Ltda., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CAPES, Brazil - Finance Code 001.

REFERENCES

- [1] Z. Lin *et al.*, "Pushing Large Language Models to the 6G Edge: Vision, Challenges, and Opportunities," *IEEE Commun. Mag.*, vol. 63, no. 9, pp. 52–59, 2025, doi:10.1109/MCOM.001.2400764.
- [2] C. Zheng *et al.*, "In-network machine learning using programmable network devices: A survey," *IEEE Commun. Surv. Tutor.*, vol. 26, no. 2, pp. 1171–1200, 2024.
- [3] S. Lee *et al.*, "Forecasting GPU Performance for Deep Learning Training and Inference," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. Rotterdam, Netherlands: ACM, 2025, pp. 493–508.
- [4] D. Zhou *et al.*, "Making In-Network Computing a Shared Infrastructure in Datacenters: Architecture, Challenges and Open Issues," *IEEE Commun. Mag.*, vol. 63, no. 7, pp. 139–145, 2025, doi:10.1109/MCOM.001.2400367.
- [5] S. Kumar *et al.*, "Simulating llm training workloads for heterogeneous compute and network infrastructure," in *Proceedings of the 2nd Workshop on Networks for AI Computing*. Coimbra, Portugal: ACM, 2025, p. 105–107, doi:10.1145/3748273.3749212.
- [6] S. Ko *et al.*, "Dfmodel: Design space optimization of large-scale systems exploiting dataflow mappings," *arXiv preprint arXiv:2412.16432*, pp. 1–22, 2024.
- [7] M. M. H. Shuvo *et al.*, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, 2023, doi:10.1109/JPROC.2022.3226481.
- [8] W. Wang *et al.*, "A survey of ai inference technologies for on-device systems," *IEEE Internet of Things Journal*, vol. 12, no. 24, pp. 51 927–51 950, 2025, doi:10.1109/JIOT.2025.3613691.
- [9] G. Sun *et al.*, "Toward ai-native task orchestration for collaborative computing in sagsins," *IEEE Commun. Mag.*, vol. 63, no. 12, pp. 112–118, 2025, doi:10.1109/MCOM.004.2400304.
- [10] K. Chuang *et al.*, "From Dusk to Dawn: An Industry Perspective on the Transition From 5G to 6G Wireless Communications Infrastructure," *IEEE Microw. Mag.*, vol. 26, no. 12, pp. 44–63, 2025, doi:10.1109/MMM.2025.3602738.
- [11] Next G Alliance, "AI-Native Wireless Networks," Alliance for Telecommunications Industry Solutions (ATIS), Tech. Rep., 2023, available at: https://nextgalliance.org/white_papers/ai-native-wireless-networks/ Accessed: 2026-01-11.
- [12] P. Henderson *et al.*, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 1–44, 2020, doi:10.5555/3455716.3455964.
- [13] T. Alonso *et al.*, "Elastic-df: Scaling performance of dnn inference in fpga clouds through automatic partitioning," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 15, no. 2, pp. 1–34, 2021.
- [14] J.-P. Stauffert *et al.*, "Latency and Cybersickness: Impact, Causes, and Measures. A Review," *Front. Virtual Real.*, vol. 1, no. 582204, pp. 1–10, 2020.
- [15] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, 2002, doi:10.1109/71.993206.

BIOGRAPHY

Washington R. D. Silva (washingtonrds@estudante.ufscar.br) is a PhD student in Computer Science at UFSCar.

Luciano S. Fraga (lucianosouza@discente.ufg.br) is a PhD student in Computer Science at UFG.

Andrew Williams (andrew.williams@ericsson.com) is a Master Researcher at Ericsson Research.

Omer Rana (ranaof@cardiff.ac.uk) is a Professor at Cardiff University.

Luiz F. Bittencourt (bit@ic.unicamp.br) is an Associate Professor at UNICAMP.

Kleber V. Cardoso (kleber@ufg.br) is a Full Professor at UFG.

Fábio L. Verdi (verdi@ufscar.br) is a Full Professor at UFSCar.