

Exposing AI Inference as a Telco Network API over the 5G User Plane

Rodrigo Moreira¹, Larissa Ferreira Rodrigues Moreira¹, Joberto S. B. Martins²,
Tereza C. Carvalho³, Flávio de Oliveira Silva⁴

¹Federal University of Viçosa (UFV), Minas Gerais, Brazil

²University of Salvador (UNIFACS), Salvador, Brazil

³University of São Paulo (USP), São Paulo, Brazil

⁴University of Minho (UMinho), Braga, Portugal

Emails: rodrigo@ufv.br, larissa.f.rodrigues@ufv.br,
joberto@ieee.org, terezacarvalho@usp.br, flavio@di.uminho.pt

Abstract—Supporting edge Artificial Intelligence (AI) inference as a network service is challenging because current mobile systems can transport tensors but do not expose or understand the inference context that gives those tensors operational meaning. Prior studies have addressed this gap through dynamic partitioning, collaborative inference, and split pipelines; however, these solutions remain application-bound, network-blind, or context-free. This work proposes Inference on Demand (IoD), an inference-on-demand abstraction that exposes split inference through a CAMARA-like Application Program Interface (API) and carries a split-aware tensor context in-band across the 5G user plane path. We evaluate the IoD on a programmable user plane function (UPF)-based testbed using a partitionable visual AI workload, showing that adaptive split selection can operate as a network-facing service while avoiding the rigid cost profile of static configurations. Including our findings, split inference becomes more than offloading when the service intent, runtime partitioning, and tensor transport are made visible to the mobile data plane. This opens a path toward network-native AI services, in which programmable telco infrastructure coordinates computation, context, and connectivity for user-facing applications.

I. INTRODUCTION

Modern applications increasingly require networks to perform more than just packet delivery [1]. Real-time Artificial Intelligence (AI), immersive sensing, and safety-critical visual analytics depend on latency, compute placement, and context that are not fully captured by best-effort connectivity alone [1], [2]. This gap has motivated open telco infrastructure initiatives, such as the GSMA Open Gateway and CAMARA, which expose network capabilities through common Application Programming Interfaces (APIs) and make operator infrastructure programmable for developers [3], [4], [5]. At the same time, the growth of 5th Generation Mobile Network (5G) traffic and edge-native workloads has made softwarized mobile cores, programmable user planes, and AI-assisted service control practical enablers for turning connectivity into a common

programmable substrate for user-facing, context-aware applications [2].

However, this broader role for the network is not available for edge inference today [6], [7]. The problem is that split inference is still mostly treated as an application-offloading technique, rather than as a programmable network service [8], [9], [10]. Existing studies have explored dynamic partitioning, but often outside the mobile user plane; collaborative inference, but without a telco service abstraction; and split-computing pipelines, but without inference-specific in-band context [11], [12]. The network can carry tensors, but it does not yet understand the inference session to which they belong. We propose Inference-on-Demand (IoD), which turns split inference into a network-facing service primitive: the application requests an inference envelope, the User Equipment (UE) adapts the split point at runtime, and the user plane carries the inference context with the tensor fragments, making the mobile network an active substrate for adaptive AI execution rather than a passive transport path [2].

To address this gap, we designed and evaluated IoD, a CAMARA-inspired service interface and user-plane mechanism that exposes adaptive split inference as a programmable network service. Rather than treating the mobile network as a passive path for tensor fragments, the IoD lets the application negotiate an inference envelope, enables the UE to adapt the split point at runtime, and binds activation traffic to inference-session context through in-band metadata carried in the GPRS Tunnelling Protocol User Plane (GTP-U)/Segment Routing over IPv6 (SRv6) packet stack. The contributions are threefold: (i) an application-facing abstraction for requesting and controlling split-inference sessions; (ii) a data-plane mechanism that transports inference-specific context together with activation fragments; (iii) and an experimental validation over a 5G user plane path showing how this service primitive can support adaptive AI execution inside the mobile network.

The remainder of this work is organized as follows. Section II reviews related work. Section III presents the proposed method. Section III-A describes the experimental testbed. Section IV discusses the results. Section V concludes the paper.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Accepted author manuscript for publication in the Proceedings of the 2026 IEEE Future Networks World Forum (FNWF 2026), Lisbon, Portugal, October 26–28, 2026.

II. RELATED WORK

Recent work has mainly advanced split inference as an optimization problem over computation, communication, and service constraints. Several studies adapt the split point or layer placement to improve latency, energy, throughput, or Service-Level Agreement (SLA) compliance in Mobile Edge Computing (MEC) systems: [13] and [14] target runtime partitioning and placement of Large Foundation Models (LFMs), while [15], [16], [17], [18], [19], [20], and [21] optimize collaborative Deep Neural Network (DNN) inference across mobile devices, edge servers, or Device-to-Device (D2D) settings. Related learning-oriented works, such as [22], [23], and [24], extend this direction toward Extended Reality (XR), satellite edge computing, and vehicular 6G systems, showing that adaptive partitioning is increasingly treated as a runtime control problem rather than a static model-design choice.

A second line of work moves closer to the network infrastructure itself. [11] proposes 5G architectural enhancements for split-computing service pipelines using Computing-Aware Traffic Steering (CATS), Segment Routing over IPv6 (SRv6), and Mobile User Plane (MUP), whereas [25] studies functional split of in-network Deep Learning (DL) across enhanced User Plane Functions (UPFs). These works show that split Artificial Intelligence (AI) is no longer only an application-layer concern. However, they do not expose inference as a CAMARA-style telco Network Application Programming Interface (API), nor do they carry inference-specific split metadata in-band with the user-plane tensor traffic.

TABLE I. SHORT STATE-OF-THE-ART SURVEY

Approach	Dynamic Split	QoS Control	Tensor Transport	User Plane	In-Band Metadata	CAMARA API
Djuhera et al. [13]	●	●	○	○	○	○
Shah et al. [22]	●	●	○	○	○	○
Koch et al. [14]	●	●	○	○	○	○
Zhang et al. [15]	●	●	○	○	○	○
Tran et al. [11]	○	○	●	●	○	○
Lim et al. [16]	●	●	○	○	○	○
Fang et al. [23]	○	●	○	○	○	○
He et al. [25]	○	○	●	●	○	○
Rasheed & Mostafa [24]	●	●	○	○	○	○
Lim & Kim [17]	●	●	○	○	○	○
Fang et al. [18]	○	●	○	○	○	○
Tan et al. [19]	●	●	○	○	○	○
Tan et al. [20]	●	○	○	○	○	○
Guo et al. [21]	●	○	○	○	○	○
Our	●	●	●	●	●	●

Table I summarizes this landscape through six dimensions: Dynamic Split means runtime partition adaptation; QoS Control means service objective enforcement; Tensor Transport means activation payload transfer; User Plane means telco datapath integration; In-Band Metadata means packet-carried inference context; CAMARA API means programmable service exposure.

III. PROPOSED METHOD

Figure 1 shows the core idea of our approach: if telco networks are to host new services, their capabilities must be exposed as programmable network APIs, not only as connectivity pipes. IoD follows this direction by exposing edge inference as a CAMARA-style [4] service. The split point k is the service knob: smaller k protects the UE but pushes larger

tensors into the user plane, whereas larger k spends more local computing resources but sends fewer bytes to the Multi-access Edge Computing (MEC). For catastrophic scenarios, where device and network conditions change over time, fixing k in advance is excessively rigid. IoD keeps the model partitionable and lets the service adapt where inference crosses from the UE to the network.

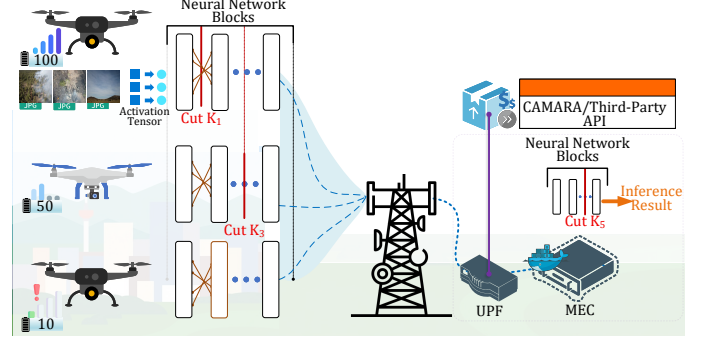


Fig. 1. Inference-on-Demand problem setting.

Let a neural model be represented as an ordered sequence of layers $\mathcal{M} = L_1, L_2, \dots, L_N$. A split point k partitions the model into a UE-side head $\mathcal{M} * \text{UE}^k = L_1, \dots, L_k$ and an edge-side tail $\mathcal{M} * \text{MEC}^k = L_{k+1}, \dots, L_N$. Given an input frame x_t , the UE computes an intermediate activation $z_t = \mathcal{M}_{\text{UE}}^k(x_t)$ and sends this activation, rather than the original frame, to the MEC. The final prediction is then computed as $\hat{y} * t = \mathcal{M} * \text{MEC}^k(z_t)$. Smaller values of k offload earlier layers and reduce UE-side computation, but they generate larger intermediate tensors. Larger values of k execute more layers locally and reduce the transmitted activation size, but increase UE-side processing. IoD exploits this trade-off by selecting k online from an admissible set $\mathcal{K}$, instead of using a single static split throughout the service session.

Algorithm 1 Runtime workflow of IoD split inference.

Require: Service intent $I = (\mathcal{K}, \tau_{\text{tx}}, \tau_{\text{head}}, q)$, frame x_t , Tunnel Endpoint Identifier (TEID)

- 1: Observe UE and path state s_t from local execution and recent transmission
- 2: Select $k_t \leftarrow \pi_I(s_t)$, with $k_t \in \mathcal{K}$
- 3: Compute $z_t \leftarrow \mathcal{M}_{\text{UE}}^{k_t}(x_t)$
- 4: Encode and fragment z_t according to quantization mode q
- 5: **for** each fragment p_j **do**
- 6: Write $(k_t, q, j, m, \text{shape}(z_t))$ into the split metadata Type-Length-Value (TLV)
- 7: Encapsulate p_j as GTP-U(TEID)/IPv6/SRH/TLV/payload and send on N3
- 8: **end for**
- 9: UPF matches the TEID, decapsulates GTP-U, parses the split TLV, and forwards to N6/MEC
- 10: MEC reassembles z_t and computes $\hat{y}_t \leftarrow \mathcal{M}_{\text{MEC}}^{k_t}(z_t)$

Figure 1 summarizes the runtime workflow. The service intent defines the admissible split range, thresholds, and quantization mode; the UE then selects $k_t = \pi_I(s_t)$ from its local execution and recent transmission state. Unlike a fixed

split, IoD does not replace the model or reconfigure the MEC service at each frame: it only changes where the current inference is partitioned. The UE transmits encoded activation tensors rather than raw frames, with compact metadata for split identification, shape, type, and reassembly. This makes the abstraction relevant to privacy-sensitive visual applications, where reducing the exposure of raw images in the user plane is part of the service requirement.

In our approach we build a CAMARA-like [4] IoD API. The API does not micromanage every frame; it opens the telco infrastructure to a new class of services by negotiating an inference envelope. As shown in Listing 1, the application requests a model, an admissible range of split points, and a quantization policy. The network grants a service session with a MEC target and a data-plane metadata schema. The concrete split point k_t is then selected at runtime by the UE, inside the admitted range, and carried in-band with the activation tensor.

```

1 // --- 1) Service Request ---
2 {
3   "applicationId": "aerial-fire-detection",
4   "serviceType": "inference-on-demand",
5   "model": {"name": "flame2-yolo-like", "task":
6     ↪ "fire-detection"},
7   "latencyObjectiveMs": 120,
8   "splitPolicy": {
9     "allowedK": [2, 3, 4, 5],
10    "defaultK": 3,
11    "decisionScope": "ue-runtime"
12  },
13  "quantization": {"allowed": ["fp16", "int8"], "default":
14    ↪ "fp16"},
15  "triggers": {"txDelayMs": 22, "headLatencyMs": 80}
16 }
17 // --- 2) Session Response ---
18 {
19   "sessionId": "iod-7f31",
20   "status": "granted",
21   "mecTarget": {
22     "nodeId": "mec-gpu-01",
23     "n6Address": "192.168.90.2",
24     "port": 5001
25   },
26   "admittedSplitRange": {"minK": 2, "maxK": 5,
27     ↪ "fallbackK": 3},
28   "dataPlaneMetadata": {
29     "carrier": "SRv6-SRH-TLV",
30     "tlvType": 124,
31     "fields": [
32       "k", "dtype", "fragmentOffset",
33       "fragmentTotal", "scale", "tensorShape"
34     ]
35   }
36 }

```

Listing 1: CAMARA-like API interaction. The request (top) defines the envelope, while the response (bottom) binds it to a MEC target and datapath metadata.

The User Plane Function (UPF) is the user-plane anchor between the radio-side path and the MEC-side service. In our architecture, the UE sends activation fragments through the N3 interface using GPRS Tunneling Protocol (GTP)-U encapsulation toward the UPF. Inside the GTP-U payload, the packet carries an Internet Protocol version 6 (IPv6) and Segment Routing Extension Header (SRH) stack with a split-inference TLV and the raw activation fragment. The relevant packet structure is according to Eq. 1:

$$\text{Ethernet/IP/UDP}_{2152}/\text{GTP-U}(\text{TEID})/\text{IPv6}/\text{SRH}+\text{TLV}/\text{payload}. \quad (1)$$

The TLV carries the selected split point k , the tensor

data type, fragment offset, total number of fragments, scaling information for quantized tensors, and tensor shape. Therefore, the split decision travels in-band with the data plane packet, while Packet Forwarding Control Protocol (PFCP) remains responsible for session-level state and TEID installation.

In our implementation, the pipeline receives N3 packets, parses the outer headers, matches the session by TEID through the Packet Detection Rule (PDR) state, and performs GTP-U decapsulation. The split-inference handling is placed immediately after decapsulation, when the inner IPv6/SRH packet becomes visible to the data-plane pipeline. At this point, the UPF can parse the split TLV, account the traffic by TEID and split point, and forward the resulting payload toward the MEC through the N6 side. The UPF only exposes the inner packet after GTP-U decapsulation, reads the in-band metadata, and keeps the inference traffic observable and forwardable inside the mobile user plane.

At the MEC, TLV metadata drives fragment reassembly, tail selection, and inference completion. The same partitionable model therefore admits several runtime operating points: later splits save user-plane bytes, earlier splits save UE compute. IoD ties these choices across three planes: the CAMARA-like API defines the intent, the UE selects k_t , and the UPF carries and observes the activation traffic toward the MEC.

A. Experimental Testbed

We evaluate our approach⁰ on the FABRIC testbed [26]. Figure 2 depicts our testbed. Traffic is generated as GTP-U slices, by a Kubernetes cluster with multiple pods acting as video source generation and forwarded through a Berkeley Extensible Software Switch (BESS)-based UPF [27], which decapsulates the tunnels and extract the TLV and process the tensor data accordingly, and forwards (N6 interface) toward MEC or the data network.

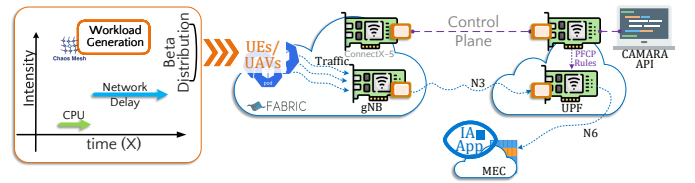


Fig. 2. Our testbed.

UE stress is generated (through ChaosMesh tool [28]) independently per pod as a seeded event process: event starts follow exponential inter-arrivals with mean 35 s over each 180-sstage, while event durations are sampled uniformly from 10 to 45 s and may overlap. For each event, ChaosMesh applies Central Processing Unit (CPU) stress with four workers and a load sampled from a bounded Beta(2,2) distribution over [15, 85]%, along with network delay and jitter sampled from bounded Beta(2,2) distributions over [5, 90] ms and [1, 20] ms, respectively.

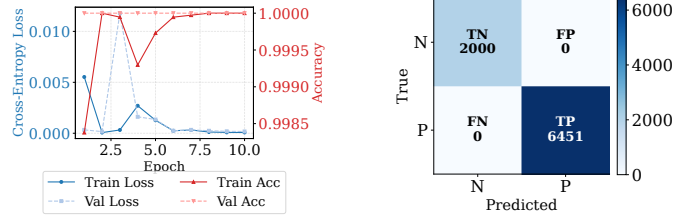
⁰ Available at https://github.com/romoreira/NetAI-APP-Inference_on_Demand.

IV. RESULTS AND DISCUSSION

A. Dataset and Neural Network Training

We use FLAME 2 (*Fire detection and modeLing: Aerial Multi-spectral image dataset*) as the visual workload for the split-inference service [29]. FLAME 2 is originally a Unmanned Aerial Vehicle (UAV) video dataset, containing raw RGB/IR video pairs and labeled frame pairs derived from those videos; in this work, we use the 254p RGB frames and the binary *Fire/NoFire* task.

The split model is derived from a ResNet18 backbone pre-trained on ImageNet, with the final fully connected layer replaced by a two-class classifier. Frames are resized to 224×224 and normalized with ImageNet statistics; training uses random horizontal flips, small rotations, and color jitter. To reduce temporal leakage between adjacent video frames, samples are grouped into temporal blocks of 1000 consecutive frames before the train/validation/test split. The baseline model is trained for 10 epochs with batch size 128, AdamW optimizer, learning rate 10^{-4} , weight decay 10^{-4} , cosine annealing, mixed precision on CUDA, and class-weighted cross-entropy.



(a) Learning curves of the baseline model.

(b) Confusion matrix on the test split.

Fig. 3. Baseline model validation on FLAME 2: (a) training and validation dynamics over epochs and (b) the resulting test-set confusion matrix.

Figure 3 reports the convergence behavior and the final confusion matrix of the trained baseline. This model is not the contribution of the paper; it provides a validated partitionable visual workload whose intermediate activations are used by the static and adaptive split-inference experiments.

B. Inference-on-Demand Split Adaptation

We carried out experiments in order to evaluate our approach. We aim to answer whether IoD can operate as a dynamic split-selection mechanism under changing user-plane load. Our results show whether IoD can recover a useful operating region without selecting k a priori, react to increasing contention, and keep the UPF load below bytes-heavy static configurations.

Figure 4 shows the compute-bytes operating frontier. The static baselines follow the expected trade-off: shallow splits reduce UE-side head inference time but transmit more bytes, whereas deeper splits reduce transmitted bytes at the cost of more UE-side computation. The IoD operating point lies near the middle of this frontier, at approximately 4.0 ms of UE head inference and 1302 KB per frame. This is close to the region represented by $k=3$, but it is reached without configuring a fixed split in advance. This result shows that the adaptive policy recovers a balanced operating point dynamically.

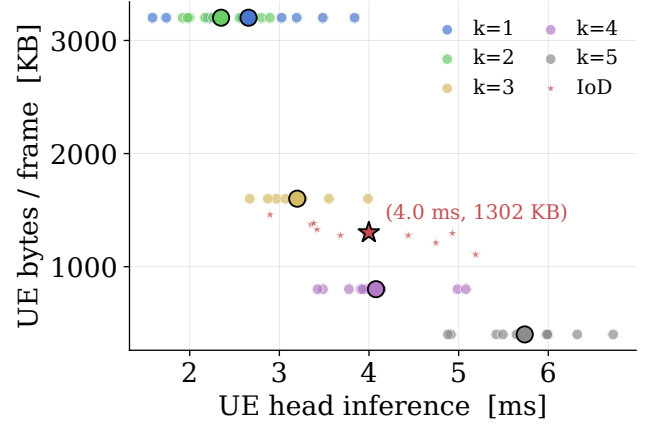


Fig. 4. Compute-bytes operating frontier across fixed split points and IoD. IoD operates near the middle of the frontier without a priori selection of k .

Figures 5 and 6 show how this operating point changes as the number of active UE replicas increases. In Figure 5, the fraction of rounds in the protection regime rises from 7% with one active replica to 45% with ten active replicas. This indicates that the controller uses the protection branch more often as contention increases. The trend is not strictly monotonic, which is expected because each seed produces its own scale and chaos sequence, but the overall direction is clear. Figure 6 confirms the same behavior through the observed split point: the mean value of k increases from 3.14 to 3.90 across the same scale range. The wider variation at higher replica counts indicates that IoD does not simply replace one static split with another. It alternates between regimes at runtime according to the measured condition of each round.

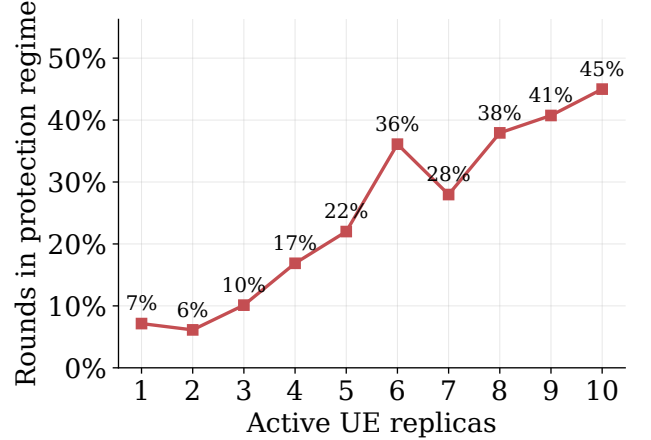


Fig. 5. Fraction of IoD rounds in the network-protection regime as the number of active UE replicas increases.

Figure 7 clarifies which controller branches were exercised by the campaign. Two regimes dominate the execution: the nominal regime, represented by $k=3$, and the network-protection regime, represented by $k=5$. At one replica, 93% of the rounds remain in the nominal regime, whereas at ten replicas this share falls to 55%. Conversely, the protection share increases from 7% to 45%. The other designed branches

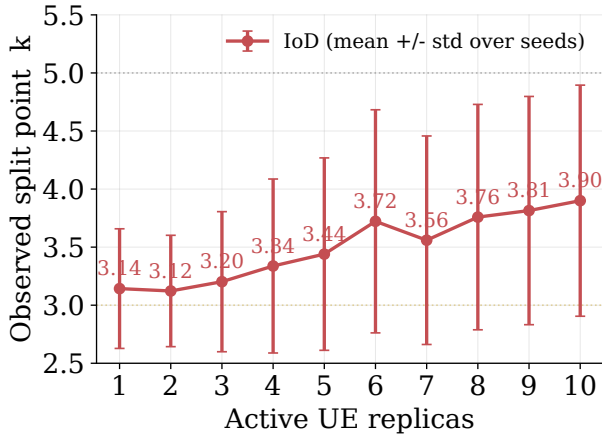


Fig. 6. Observed IoD split point as a function of active UE replicas. Error bars show variation across seeds.

were not activated because the workload did not create UE-side compute pressure: head inference remained far below the configured 80 ms threshold. Therefore, this campaign demonstrates the network-protection behavior of IoD, while the compute-protection branches should be treated as part of the controller design rather than as empirically exercised regimes in this experiment.

Figure 8 reports the UPF-counted load as the number of active UE replicas changes. The static baselines grow almost linearly with scale. At ten replicas, the shallowest split reaches approximately 171.84 MB per round, whereas $k=3$ reaches 85.95 MB and $k=5$ reaches 21.53 MB. IoD reaches 56.97 MB per round at the same scale, about one third of the load imposed by $k=1$. This confirms that IoD substantially reduces user-plane load compared with bytes-heavy split points. The shape of the IoD curve is also informative: between five and six replicas, the load increases only from 35.89 MB to 37.62 MB, which coincides with the controller using the protection regime more often. Thus, the reduction is not only visible in the average operating point; it also appears as a scale-dependent response in the UPF load curve.

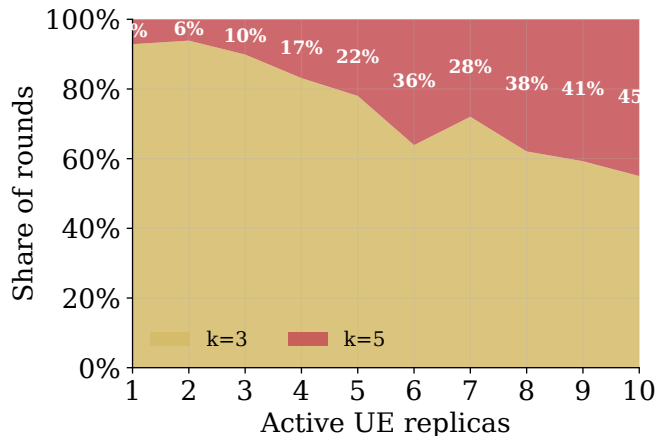


Fig. 7. Share of IoD rounds assigned to the nominal and network-protection regimes.

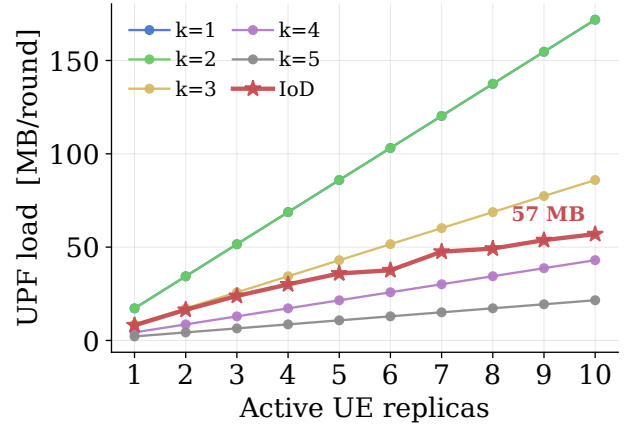


Fig. 8. UPF-counted load under increasing numbers of active UE replicas.

Finally, Figure 9 compares IoD with each static baseline using paired seeds. Negative values indicate that IoD transmits fewer bytes per frame than the corresponding static split. The result is consistent across seeds: IoD reduces transmitted bytes by approximately 1899 KB per frame relative to $k=1$ and $k=2$, and by approximately 299 KB relative to $k=3$. As expected, IoD transmits more bytes than the more compute-heavy configurations, with average differences of approximately 501 KB relative to $k=4$ and 901 KB relative to $k=5$. This paired view reinforces the intended reading of the method: IoD is not a universal minimizer of bytes or computation. It is a dynamic compromise and a novel mobile network data plane service that avoids the high communication cost of shallow splits, stays near a balanced operating region, and increases protection when the network becomes more pressured.

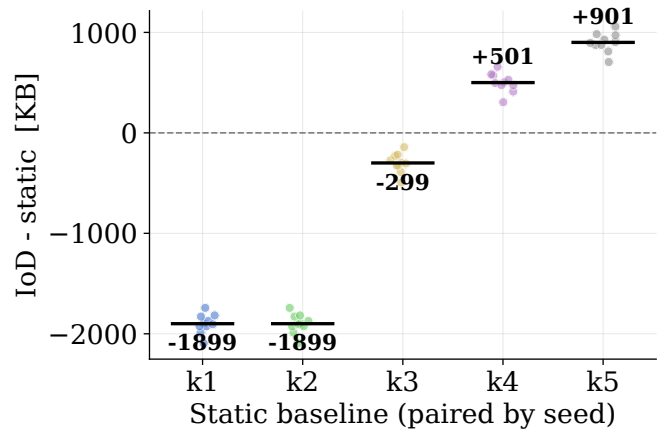


Fig. 9. Per-seed paired difference between IoD and each static baseline in UE bytes per frame. Negative values indicate that IoD transmits fewer bytes than the corresponding static split.

Overall, the results support the feasibility of IoD as an online adaptation mechanism for split inference over a mobile user-plane path. The method recovers a balanced operating region without prior split selection, reacts to increasing contention by shifting toward protection, and keeps UPF load below bytes-heavy static baselines.

V. CONCLUDING REMARKS

This work addressed the gap between edge inference as an application-side offloading mechanism and edge inference as a programmable telco service. The results show that the IoD can adapt the split point at runtime, keep tensor transport observable in the 5G user plane, and reduce the pressure imposed by bytes-heavy static configurations. This finding suggests that mobile networks can support AI execution not only by providing connectivity but also by exposing service intent and carrying inference context along the data path. Future work will extend IoD toward tighter CAMARA API integration, policy-driven split selection, and larger multi-tenant MEC deployments. Overall, the IoD reframes split inference as a network-native service primitive for adaptive AI over a programmable mobile infrastructure.

ACKNOWLEDGMENT

The authors thank FAPEMIG (Grant #APQ00923-24), FAPESP MCTIC/CGI Research project 2018/23097-3 - SFI2 - Slicing Future Internet Infrastructures, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, PROPP/UFU, and Fundação para a Ciência e Tecnologia (FCT) within the R&D Unit Project Scope UID/00319/2025 - Centro ALGORITMI (ALGORITMI/UM) for supporting this work. The authors also acknowledge IMAGINE-B5G (EU Horizon Europe, Grant No. 101096452) for the experimental infrastructure used in this work.

REFERENCES

- [1] X. Tang, Q. Li, J. Peng, Y. Tan, Z. Yuan, Y. Song, G.-M. Muntean, and Y. Jiang, "Low-latency video streaming: Applications, challenges, and trends," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 6384–6414, 2026.
- [2] X. Niu, W. Yuan, L. Tan, N. Su, Y. Zhang, and T. Q. S. Quek, "Beyond connectivity: How large ai models unlock semantic 6g networks?" *IEEE Communications Magazine*, vol. 64, no. 4, pp. 106–112, 2026.
- [3] GSMA, "Gsm open gateway initiative," 2024, acessado em: 29 jun. 2026. [Online]. Available: <https://www.gsma.com/opengateway>
- [4] Linux Foundation and GSMA, "CAMARA: The Telco Global API Alliance," 2024, acesso em: October 14, 2026. [Online]. Available: <https://camaraproject.org/>
- [5] J. Ordóñez-Lucena and F. Dsouza, "Pathways towards network-as-a-service: the camara project," in *Proceedings of the ACM SIGCOMM Workshop on Network-Application Integration*, ser. NAI '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 53–59. [Online]. Available: <https://doi.org/10.1145/3538401.3546825>
- [6] S. Lin, S. Xu, B. He, H. Liu, D. Kong, X. Chen, D. Zhang, C. Wu, M. Li, X. Liu, Y. Wu, and M. K. Khan, "Ndif: A distributed framework for efficient in-network neural network inference," *Computers & Security*, vol. 157, p. 104593, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167404825002822>
- [7] J. Liu, Y. Du, K. Yang, J. Wu, Y. Wang, X. Hu, Z. Wang, Y. Liu, P. Sun, A. Boukerche, and V. C. M. Leung, "Edge-cloud collaborative computing on distributed intelligence and model optimization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 28, pp. 5049–5080, 2026.
- [8] J. V. Fatur Lessa, J. A. Marques, R. Parizotto, and L. P. Gaspar, "In-network inference with neuralp4: Auto-generating full-fledged nn models for pdps," in *NOMS 2025-2025 IEEE Network Operations and Management Symposium*, 2025, pp. 1–10.
- [9] H. Kim, S. Yoon, C. Bae, S. Lee, and S. Pack, "Tinnee: Traffic-aware adaptive in-network intelligence via early-exit strategy," *IEEE Transactions on Networking*, vol. 34, pp. 5037–5052, 2026.
- [10] R. Moreira, L. F. R. Moreira, and F. d. O. Silva, "Unleashing ai-empowered slices on mobile networks for natively cognitive service delivery," *IEEE Access*, vol. 13, pp. 147 757–147 771, 2025.
- [11] M.-N. Tran, K. Nguyen Trung, and Y. Kim, "Design of 5g architecture enhancements for supporting edge split computing service pipeline," *IEEE Access*, vol. 13, pp. 191 515–191 530, 2025.
- [12] G. Zhang, Q. Zhang, L. Feng, F. Zhou, P. K. Donta, and S. Dustdar, "Task-aware collaborative inference and fine-grained dnn partitioning in mec networks," *IEEE Transactions on Mobile Computing*, vol. 25, no. 6, pp. 8911–8927, 2026.
- [13] A. Djuhera, F. Koch, and A. Binotto, "Joint partitioning and placement of foundation models for real-time edge ai," in *2026 International Conference on Computing, Networking and Communications (ICNC)*, 2026, pp. 459–464.
- [14] F. Koch, A. Djuhera, and A. Binotto, "Intelligent orchestration of distributed large foundation model inference at the edge," *Computer Networks and Communications*, p. 111–128, 2025. [Online]. Available: <http://dx.doi.org/10.37256/cnc.3220256807>
- [15] G. Zhang, Q. Zhang, L. Feng, F. Zhou, P. K. Donta, and S. Dustdar, "Task-aware collaborative inference and fine-grained dnn partitioning in mec networks," *IEEE Transactions on Mobile Computing*, vol. 25, no. 6, pp. 8911–8927, 2026.
- [16] J.-A. Lim, J. Lee, J. Kwak, and Y. Kim, "Cutting-edge inference: Dynamic dnn model partitioning and resource scaling for mobile ai," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3300–3316, 2024.
- [17] J.-A. Lim and Y. Kim, "Real-time dnn model partitioning for qoe enhancement in mobile vision applications," in *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2022, pp. 407–415.
- [18] J. Fang, Y. An, Y. Liu, Z. Teng, X. Zhai, H. Tang, and H. Chen, "Mae: Collaborative inference acceleration with efficient dnn partitioning and resource allocation in resource-constrained edge computing," *Computer Networks*, vol. 278, p. 112073, 2026. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S138912862600085X>
- [19] L. Tan, K. Guo, Z. Tan, S. Guo, Z. Kuang, J. Zhao, and D. Niyato, "Darl: Demand-aware partitioned collaborative inference for on-device models," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2026.
- [20] L. Tan, P. Zhou, S. Guo, J. Zhao, Z. Kuang, D. Qiao, and L. Yang, "Partitioned collaborative inference for on-device models via evolutionary reinforcement learning," in *2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS)*, 2025, pp. 714–724.
- [21] K. Guo, Y. Xiao, X. Lu, and L. Xiao, "Learning-based low-latency collaborative inference for multi-branch models in d2d-assisted mec," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 01–06.
- [22] S. D. A. Shah, F. Ali, A. A. AlZubi, and D. Kwak, "Collaborative intelligence: Orchestrating ultra-low latency xr experiences in 6g consumer networks," *IEEE Consumer Electronics Magazine*, pp. 1–11, 2026.
- [23] C. Fang, C. Xu, Z. Li, H. Xiao, S. Yang, L. Zhong, and G.-M. Muntean, "Non-intrusive handover strategy optimization for model-partitioned dnn inference in satellite edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–17, 2026.
- [24] I. Rasheed and H. Mostafa, "Mobility-aware predictive split federated learning for 6g vehicular networks with ultra-low latency guarantees," *Computer Networks*, vol. 270, p. 111553, 2025. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1389128625005201>
- [25] J. He, H. Wu, X. Xiao, R. Bassoli, and F. H. P. Fitzek, "Functional split of in-network deep learning for 6g: A feasibility study," *IEEE Wireless Communications*, vol. 29, no. 5, pp. 36–42, 2022.
- [26] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.
- [27] Open Networking Foundation (ONF), "Aether: Open source 5g connected edge platform," <https://opennetworking.org/aether/>, 2021, acessado em: 18 de junho de 2026.
- [28] Chaos Mesh Authors, "Chaos Mesh: A Powerful Chaos Engineering

Platform for Kubernetes,” <https://chaos-mesh.org/>, 2020, gitHub repository: <https://github.com/chaos-mesh/chaos-mesh>.

- [29] B. Hopkins, L. O’Neill, F. Afghah, A. Razi, E. Rowell, A. Watts, P. Fule, and J. Coen, “Flame 2: Fire detection and modeling: Aerial multi-spectral image dataset,” 2022. [Online]. Available: <https://dx.doi.org/10.21227/swyw-6j78>