POLITECNICO DI TORINO
Repository ISTITUZIONALE

Traffic-aware DNN Inference Task Offloading in the Mobile Device-Edge Continuum

Original
Traffic-aware DNN Inference Task Offloading in the Mobile Device-Edge Continuum / Chukhno, Olga; Singh, Gurtaj; Campolo, Claudia; Chiasserini, Carla Fabiana; Molinaro, Antonella. - (2025). (Intervento presentato al convegno IEEE INFOCOM 2025 ICCN tenutosi a London (UK) nel 19 May 2025).

Availability:
This version is available at: 11583/2997387 since: 2025-02-09T14:29:20Z

Publisher:
IEEE

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

23 July 2025

Traffic-aware DNN Inference Task Offloading in the Mobile Device-Edge Continuum

Olga Chukhno*†, Gurtaj Singh*†, Claudia Campolo*†, Carla Fabiana Chiasserini‡†, Antonella Molinaro*†
*University Mediterranea of Reggio Calabria, Italy
†Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), Italy
‡Politecnico di Torino, Italy

***Abstract*—The rapid growth of mobile devices and machine learning (ML)-based applications is driving a surge in data traffic. Even when inference tasks are considered, a huge amount of data needs to be transferred into the network, e.g., large Deep Neural Network (DNN) models retrieved for on-device inference, streams of input data sent from the device to the edge if the task is offloaded. To address the resulting potential network congestion, we formulate a novel optimization problem aimed at deciding where to execute streams of DNN inference tasks from multiple devices across the mobile device-edge continuum in order to minimize the amount of exchanged data traffic, while satisfying accuracy, latency, and battery constraints. The formulated problem also selects the model variant (in terms of size and accuracy) that best suits the placement decision (device, edge). Results have been collected under a wide variety of different settings, which showcase the validity of our proposal and its supremacy over the considered benchmark schemes, with gains in terms of saved bandwidth up to 98%.**

***Index Terms*—DNN model compression, inference, edge computing, offloading**

I. Introduction

Several applications rely on Deep Neural Networks (DNNs) to significantly advance decision making in real-world application domains, e.g., video surveillance, autonomous driving, cognitive assistance, etc. Although less computation-hungry compared to training, inference through DNN is expected to be pervasively requested [1]. Hence, similarly to other computing tasks, it is crucial to decide whether inference has to be either locally executed in potentially constrained end-devices or offloaded to more powerful edge servers. Although task offloading at the edge has been extensively investigated in the literature [2], inference offloading raises additional challenges because it requires also a decision *about which model variant to be used*, besides *whether to offload or not* an inference task. Indeed, compact models can be deployed in constrained devices at the expenses of a potentially lower, but still reasonable, accuracy [3].

In terms of data exchange, the execution of an inference task requires *(i)* either the (potentially huge) model available at the edge to be retrieved by the end-device if it needs to be executed locally or *(ii)* raw input data to be transferred from the source to the edge if the task is offloaded. Even in the latter case, with input data typically of smaller sizes compared to the DNN models, a task can exhibit a data-intensive nature, when several inference requests are issued from a single device upon incoming high-data rate sensor streams (e.g., voice, video). For those types of inference requests, the decision about whether to offload them or not, if not wisely taken, may imply huge network traffic when inference tasks are massively requested. Notwithstanding, such potential drawbacks from the perspective of the network operator, which may face congestion issues, are typically neglected. Energy and delay minimization [4], [5], as well as accuracy maximization [6], are instead the common goals in the literature about inference task offloading.

To fill this gap, in this work, we formulate an optimization problem placing inference tasks either at the end-device or the edge, with the aim of *minimizing the amount of data exchanged into the network, while satisfying accuracy, latency, and battery constraints*. Besides the targeted optimization goal, this work jointly features crucial novelties and contributions compared to the majority of the existing literature:

- the *accurate characterization of the computing latency* in the formulated problem, *(i)* through an analytical approach estimating the inference execution time, validated through real measurements, and *(ii)* through queuing theory, to account for load conditions at the edge and end-device, which may affect the offloading decision, as preliminarily assessed in [3];
- the *selection of the proper model version* as an additional output of the formulated optimal placement problem;
- the decision about the placement not of a single inference task only, but of a *stream of multiple requests per each device*;
- a wide set of results to assess the performance of the proposed offloading policy against different benchmarks, under different settings, in terms of number of devices, number of requests per device and heterogeneity of edge/device capabilities and demands in terms of accuracy and latency.

The remainder of this paper is organized as follows. Section II provides the system model. The formulated optimization problem is described in Section III. Results are reported in Section IV, before concluding in Section V.

II. System Model

A. Reference scenario and main assumptions

We consider a set of N heterogeneous mobile devices, $\mathcal{N}=\{1,\ldots,N\}$ within the coverage area of a Base Station

This work was partially supported by the SNS-JU-2022 project ADROIT6G under the European Union's Horizon Europe research and innovation programme under Grand Agreement No. 101095363 and partially by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU (PE00000001 - program "RESTART").

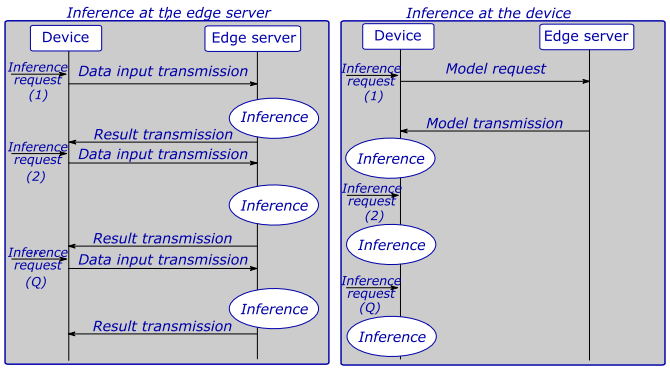


Fig. 1: Wokflows for DNN inference execution at the edge server (left) and at the mobile device (right).

(BS) co-located with an edge server, denoted by e. The edge server is equipped with computing capabilities c, whereas each mobile device $n \in \mathcal{N}$ has CPU capability c_n and a residual battery capacity ε_n.

Without loss of generality, we consider a DNN inference model (e.g., for the execution of object detection or speech recognition tasks). The model has a set of possible configurations, i.e., pruned versions, $\mathcal{K}=\{1,\ldots,K\}$, each associated with a different value of pruning factor to better match the resource-constrained capability of a requesting device. Each version of the DNN model, which we assume stored at the edge, is characterized by an accuracy level that was achieved during training, a_k. Given the evolving nature of data distribution, DNN models have a limited lifetime, beyond which an update is required [7]. Within this period, referred to as *prediction horizon*, H, each device is expected to request Q_n inference tasks for a sequence of different input samples. Considering independent and non-synchronized requests, as in [8], [9], the generation of inference requests can be modeled at the single device n as a Poisson process with parameter λ_n, and $Q_n=\lambda_n \cdot H$.

Each inference request is characterized by specific requirements in terms of maximum response latency, L_n, referring to the time since the request for the inference task has been issued at the device until the output is available, and minimum accuracy, A_n.

An edge orchestrator is responsible for deciding whether to perform an inference task locally on the end device or offload it to the edge server. In addition, it determines the optimal model instance (i.e., full model or pruned by a given factor) that best suits the capabilities of the requesting end-device and the application requirements.

The resulting workflows for the DNN inference execution pipelines are depicted in Fig. 1 and detailed below.

Inference at the edge server. First, *(i)* the inference input sample (of size ξ) is transmitted from the device to the edge server through the BS. The edge server then *(ii)* executes inference and *(iii)* sends back the inference output to the requesting device. These steps apply for any Q_n.

Inference at the mobile device. In this case, *(i)* after the request has been issued by the device, *(ii)* a DNN model (of size ν_k) is transmitted to the device. Then, *(iii)* inference is executed at the device. These steps apply only for $Q_n=1$. Upon receiving the DNN model, the device can store it locally and use it for subsequent inference tasks ($Q_n>1$) until a model update is necessary.

B. Data exchange

We assume that the devices and the BS operate at 5G mid and/or high bands (i.e., microwave and/or mmWave). We model channel propagation according to the 3rd Generation Partnership Project (3GPP) urban micro (UMi) street canyon specification. Devices can experience blockage by small- and large-scale objects, referred to as line-of-sight (LoS)/non-line-of-sight (NLoS) states. The associated UMi path loss (measured in dB) is given by:

$$\gamma_{\text{dB}}(\delta_n)=\beta+10\zeta\log_{10}(\delta_n)+20\log_{10} f, \tag{1}$$

where f is the operating frequency in GHz, δ_n is the three-dimensional distance between the BS and device n, whereas β and ζ are the blockage and propagation coefficients (resp.).

Then, the total received signal power in the uplink or downlink direction, involving the BS and device n, is defined as:

$$p_n^{\text{rx}}=\frac{p^{\text{tx}} D_0 \rho(\alpha_n)}{\gamma(\delta_n)}, \tag{2}$$

where p^{tx} is the transmit power, $\gamma(\delta_n)$ is the linear path loss, D_0 refers to the maximum directivity along the antenna boresight, α_n represents the angular deviation of the transmit/receive direction between device n and the BS, and $\rho(\alpha_n)\in[0,1]$ is a function that scales directivity D_0 [10].

According to Shannon's theorem, the achievable data rate can be written as:

$$r_n=W\log_2\left(1+\frac{p_n^{\text{rx}}}{W\, N_0\, N^{\text{F}}}\right) \tag{3}$$

where W denotes the available bandwidth, N_0 is the power spectral density of noise per 1 Hz, and N^{F} is the noise figure.

Notice that, when inference is executed on the device n, the DNN model with configuration k needs to be transmitted from the BS, which the edge server is co-located with, to the end-device at the downlink data rate, r_n^{d}. Conversely, if inference is performed at the edge server, the input sample has to be transferred from the device to the edge server via the BS at the uplink data rate, r_n^{e}. The delay associated with such transmissions can thus be computed, respectively, as follows:

$$T_{n,k}^{\text{m}}=\frac{\nu_k}{r_n^{\text{d}}}, \quad T_n^{\text{s}}=\frac{\xi}{r_n^{\text{e}}}. \tag{4}$$

The energy consumption at the device n for the transmission of data samples and for receiving the DNN model with configuration k are, respectively, computed as follows:

$$E_n^{\text{s,d}}=p^{\text{tx,d}}T_n^{\text{s}}, \quad E_{n,k}^{\text{m,d}}=(p^{\text{rx,d}}+p^{\text{E}})T_{n,k}^{\text{m}}, \tag{5}$$

where $p^{\text{tx,d}}$ is the device transmission power, $p^{\text{rx,d}}$ is the power spent in the reception stage at the device, and p^{E} is the baseband electric circuit power unit.

In the case of inference execution at the edge server, we neglect the time (and relevant energy) needed to transmit the result of the computation from the edge to the device. Indeed,

for many applications, such as object/face recognition, the size of the computation output is much smaller than the input data [11]. Similarly, when executing inference at the device, the time (and relevant energy) spent to request the model is also neglected, because the size of the request issued by a device is significantly lower than that of the model itself.

C. Inference time and energy consumption

Instead of relying on heavy profiling for each DNN model and computing capabilities settings [6], to compute the inference time of the DNN model, we first assess the number of floating-point operations per second (FLOPS) required to execute individual layers, according to the analytical approach in [12]. DNN models typically comprise convolutional layers and fully connected layers. For a convolutional layer of the DNN model with configuration k, the aggregate FLOPS are calculated as:

$$\phi_k^{\mathrm{c}} = (2 \cdot g_k z_k v_k) \cdot w_k h_k u_k \cdot b \,, \tag{6}$$

where b is the batch size, g_k, z_k are the input tensor dimensions, and v_k represents the dimension of the filter kernel, whereas u_k, w_k, and h_k indicate the output tensor dimensions. For fully connected layers of the DNN model with configuration k, the FLOPS are given by:

$$\phi_k^{\mathrm{f}} = (2 \cdot \iota_k - 1) \cdot \eta_k \cdot b \,, \tag{7}$$

where ι_k and η_k denote the dimensions of the input and output tensors, respectively. Thus, the total number of FLOPS required for executing a DNN inference model is given by:

$$\phi_k = \sum_{i=1}^{l^{\mathrm{c}}} \phi_{i,k}^{\mathrm{c}} + \sum_{j=1}^{l^{\mathrm{f}}} \phi_{j,k}^{\mathrm{f}}, \tag{8}$$

where l^{c} and l^{f} are the number of convolutional and fully connected layers, respectively.

The time to execute the inference of the DNN model with configuration k at the edge server and at device n, and the energy expenditure incurred during the execution of inference for the DNN model with configuration k at device n are, respectively:

$$T_k^{\mathrm{inf,e}} = \frac{\phi_k}{c} \,, \quad T_{n,k}^{\mathrm{inf,d}} = \frac{\phi_k}{c_n} \,, \quad E_{n,k}^{\mathrm{inf,d}} = p_n^{\mathrm{d}} T_{n,k}^{\mathrm{inf,d}} \,, \tag{9}$$

where p_n^{d} is the power consumption of device n (in Watt).

D. Queuing and task completion times

The operation of inference execution at the edge server can be characterized by a $M/M/1-PS$ (processor sharing) queuing model, as done, e.g., in [13]. The total arrival rate at the edge for DNN inference tasks from different devices, which follows a Poisson distribution, is given by $\lambda^{\mathrm{e}} = \sum_{n=1}^{N} \lambda_n x_n^{\mathrm{e}}$ where $x_n^{\mathrm{e}} = 1$ if inference from device n is executed at the edge server, and 0 otherwise.

The average service rate of inference tasks at the edge can then be written as:

$$\mu^{\mathrm{e}} = \frac{1}{\sum_{n=1}^{N} x_n^{\mathrm{e}}} \left[x_n^{\mathrm{e}} y_{n,k} \mu_k^{\mathrm{e}} + \sum_{\substack{n'=1, \\ n' \neq n}}^{N} \sum_{k'=1}^{K} x_{n'}^{\mathrm{e}} y_{k'} \mu_{k'}^{\mathrm{e}} \right], \tag{10}$$

where $\mu_k^{\mathrm{e}} = 1/T_k^{\mathrm{inf,e}}$ is the service rate for the DNN model with configuration k, and $y_{n,k}$ is the decision variable for the model and configuration selection, which is equal to 1 if the DNN model with configuration k is executed, and 0 otherwise.

The time to execute inference at the edge can be thus defined as:

$$T^{\mathrm{inf,e}} = \frac{1/\mu^{\mathrm{e}}}{1-\rho^{\mathrm{e}}} \,, \text{ where } \rho^{\mathrm{e}} = \lambda^{\mathrm{e}} / \mu^{\mathrm{e}}. \tag{11}$$

On the other hand, a device can execute only one inference at a time, which makes the $M/M/1$ queuing model more suitable in this case. The arrival rate for a single device n can be written as: $\lambda^{\mathrm{d}} = \lambda_n x_n^{\mathrm{d}}$, where $x_n^{\mathrm{d}} = 1$ if inference is executed at device n, and 0 otherwise. The service rate is $\mu_{n,k}^{\mathrm{d}} = 1/T_{n,k}^{\mathrm{inf,d}}$, which depends on configuration k. The queuing time can then be computed as:

$$T_{n,k}^{\mathrm{q,d}} = \frac{1}{\mu_{n,k}^{\mathrm{d}} - \lambda^{\mathrm{d}}} \,. \tag{12}$$

The overall time required to obtain the output of an inference task using the DNN model with configuration k for $Q_n = 1$ is given by:

$$T_{n,k}(Q_n = 1) = x_n^{\mathrm{e}} [T_n^{\mathrm{s}} + T_k^{\mathrm{inf,e}}] + x_n^{\mathrm{d}} [T_{n,k}^{\mathrm{m}} + T_{n,k}^{\mathrm{inf,d}} + T_{n,k}^{\mathrm{q,d}}] \,. \tag{13}$$

Instead, for the subsequent inference tasks ($Q_n > 1$) using the same instance of the DNN model, the time to get the output reduces to:

$$T_{n,k}(Q_n > 1) = x_n^{\mathrm{e}} [T_n^{\mathrm{s}} + T_k^{\mathrm{inf,e}}] + x_n^{\mathrm{d}} [T_{n,k}^{\mathrm{inf,d}} + T_{n,k}^{\mathrm{q,d}}] \,. \tag{14}$$

III. Optimization Framework

Our proposal provides an optimization framework which decides whether to offload or not an inference task in order to minimize the data transferred over the network, subject to the application requirements (in terms of accuracy and latency) and the system constraints (device battery capacity). The targeted goal well reflects the operator's perspective that aims to reduce the traffic flowing over the radio links as well as the user's demand.

The amount of data exchanged for executing the Q_n inference requests issued by the device n for the DNN model with configuration k can be written as:

$$D_{n,k} = y_{n,k} x_n^{\mathrm{d}} \nu_k + x_n^{\mathrm{e}} Q_n \xi, \tag{15}$$

where the first term on the right hand side of the above expression accounts for the delivery of the (full/pruned) model from the edge server to the requesting device in the case of inference execution at the device, while the second one accounts for the transfer of of a stream of input data from the end-device to the edge server when inference is offloaded to the edge.

The edge orchestrator has to make the following decisions: *(i)* whether the inference model should be executed at the device or at the edge server, and *(ii)* which DNN model (full or pruned, and, if the latter, by which factor) best matches the accuracy and computational constraints of the device requesting the inference task. These decisions must be periodically re-evaluated for every optimization interval (i.e.,

every τ seconds). Accordingly, we can formulate the following optimization problem:

$$\min_{\{x,y\}} \sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}} D_{n,k} \tag{16}$$

$$x_n^{\mathrm{e}}+x_n^{\mathrm{d}}=1, \sum_{k\in\mathcal{K}} y_{n,k}=1, \forall n\in\mathcal{N}\ , \tag{17}$$

$$a_k y_{n,k} \geq A_n, \forall n\in\mathcal{N}, k\in\mathcal{K}\,, \tag{18}$$

$$T_{n,k}(Q_n>1)y_{n,k} \leq L_n, \forall n\in\mathcal{N}, k\in\mathcal{K}\,, \tag{19}$$

$$E_{n,k}^{\mathrm{d}} y_{n,k} \leq \varepsilon_n, \forall n\in\mathcal{N}, k\in\mathcal{K}\,, \tag{20}$$

$$x_n^{\mathrm{d}}, x_n^{\mathrm{e}}, y_{n,k}\in\{0,1\}, \forall n\in\mathcal{N}, k\in\mathcal{K}\,, \tag{21}$$

where $E_{n,k}^{\mathrm{d}}$ represents the overall energy expenditure at device n, given by:

$$E_{n,k}^{\mathrm{d}}=x_n^{\mathrm{d}} y_{n,k}(E_{n,k}^{\mathrm{m,d}}+Q_n E_{n,k}^{\mathrm{inf,d}})+x_n^{\mathrm{e}} Q_n E_n^{\mathrm{s,d}}\,. \tag{22}$$

Constraint (17) ensures that each device n is assigned to only one inference execution mode as well as one configuration of the DNN model. Constraint (18) guarantees that the accuracy of the selected instance of the DNN model at least equals the minimum required accuracy. Constraint (19) sets the maximum inference task completion time. Constraint (20) ensures that the energy consumption of the device does not exceed its residual battery capacity. Finally, Constraint (21) imposes that the decision variables are binary.

The following propositions hold.

Proposition 1. *The optimization problem in Section III is a nonlinear programming problem.*

Proof. The decision variables are binary, and the objective function and constraints (i.e., (19)-(20)) are nonlinear functions of such variables. Thus, the problem is nonlinear. □

Proposition 2. *The optimization problem in Section III is NP-hard.*

Proof. To prove Proposition 2, we reduce the 0-1 Knapsack problem, which is known to be NP-hard, into our optimization problem [14]. In our case, each item in the knapsack problem corresponds to a decision to execute an inference task on either the device or the edge. The value of each item corresponds to a reduction in the data transfer resulting from executing an inference task either locally or offloaded. The weights of each item correspond to the accuracy, latency, and energy consumption associated with executing an inference task on the device, which must be within the limits of the weight capacity of the knapsack, that is, to satisfy constraints (18), (19), and (20). Given that there is polynomial mapping between any instance of the 0-1 Knapsack Problem and our inference task distribution problem, such that solving the latter problem provides a solution to the former, our problem must be at least as difficult as the 0-1 Knapsack problem. □

In the following section, we use the Python-based Gurobi solver [15] to determine the optimal inference placement and model configuration selection. Indeed, its advanced algorithms make Gurobi suitable for addressing small-medium instances of the problem at hand.

TABLE I: Inference time validation for the full model

Mobile device		Edge	
analytical [s]	**measured [s]**	**analytical [s]**	**measured [s]**
4.33	3.47	0.21	0.20

IV. Performance Evaluation

A. Validation of inference time

For our experimental study, we consider a Linux Ubuntu VM as mobile device, equipped with 4 cores of an Intel Core i7-9750H processor, 4 GB of RAM, and 30 GB of disk space. The edge server, instead, is represented by an Intel Core i9-14900K processor, 64 GB of RAM. As DNN model, we selected MobileNet for object recognition. Each image sample used for inference has a size of about 300 kB. Table I compares the inference time for the full model estimated through (9) against the one measured through our experiments on the mobile device and the edge server. Analytical results closely match the experimental results, which confirms the validity of our model[1].

B. Settings, scenarios, and metrics

Settings. Building upon the validation of the inference execution time, we now delve into the evaluation of our inference task offloading policy. To compute the data exchange latency, we simulate a 5G New Radio (NR) BS operating at 3.5 GHz. We assume a 100 MHz available bandwidth, and that both transmitter and receiver use a 4×4 antenna array. The sub-6 GHz channel is modeled using the 3GPP Urban Micro (UMi) street canyon channel model. As DNN compression technique, we adopt magnitude-weight pruning, a method that removes connections with the smallest weights [16]. A summary of our experimental settings is presented in Table II.

Scenarios. We consider three scenarios to evaluate system performance under varying prediction horizons and accuracy

[1]Although not shown, due to length constraints, the validity has been tested and confirmed also for other models.

TABLE II: Experimental settings

Notation and description	**Value**
N: number of mobile devices	100, 5-300
K: number of pruning factors	10 {0,10,20,30,40,50,60,70,80,90} [%]
a_k: training accuracy	92, 90.4, 88.8, 87.3, 85.7, 84.2, 82.6, 81.1, 79.5, 78 [%] [3]
ν_k: DNN model size	16, 14, 12, 11, 10, 9, 8, 6, 4, 2 [MB] [3]
ξ: input data size	0.3 [MB]
H: prediction horizon	3600 [s] (Scenario ShortH&HighA) 14400 [s] (other scenarios)
Q_n: number of requests	1-500
L_n: maximum task completion time	H/Q_n [s]
ε_n: battery capacity	[3.7, 18.5] [Wh]
A_n: minimum model accuracy	91 [%] (Scenario ShortH&HighA) [78, 90] [%] (other scenarios)
f: carrier frequency	3.5 [GHz]
W: bandwidth	50 [MHz]
β: non-blocked/blocked coefficients	32.4/47.4
ζ: LoS/NLoS coefficients	2.1/3.19
N_0: power spectral density of noise per 1 Hz	-174 [dBm/Hz]
$p^{\mathrm{tx}}/p^{\mathrm{tx,d}}$: transmit power	23 (BS) / 10 (device) [dBm]
-: service area radius	400 [m]
p^{E}: baseband circuit power	5.34 [Watt]
p_n^{d}: inference power consumption	45 [Watt]

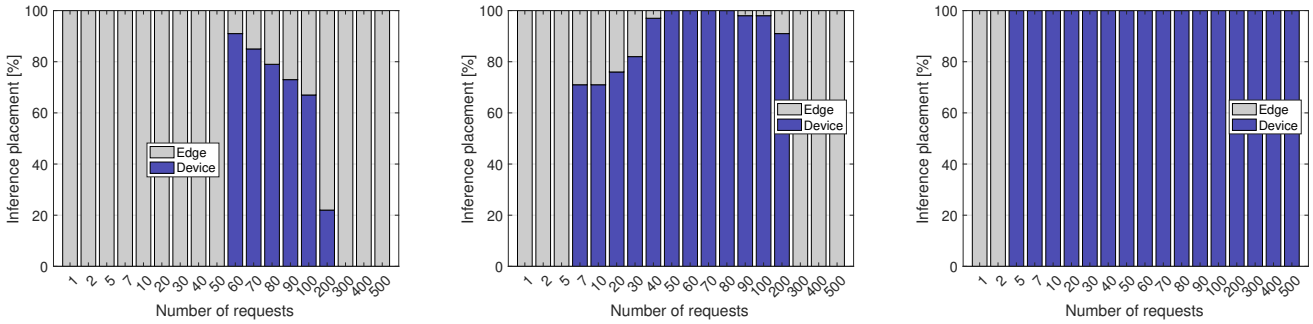
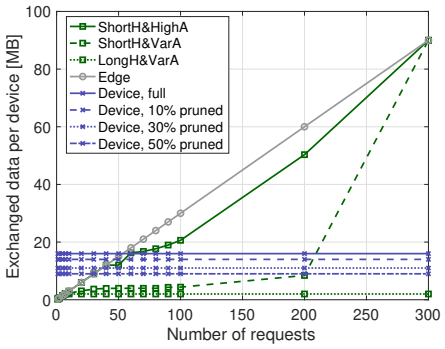
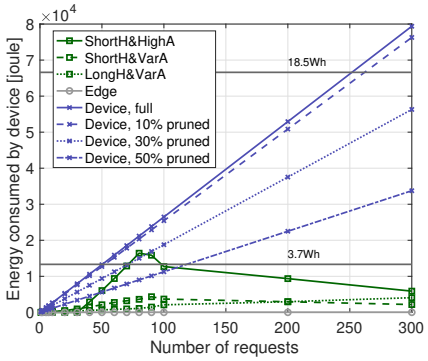


Fig. 2: Metrics for the optimal policy (N=100): ShortH&HighA (left), ShortH&VarA (center), LongH&VarA (right).



(a) Exchanged data per device.



(b) Energy consumed by device through ML pipeline.

Fig. 3: Metrics measured per the entire stream of requests Vs. number of inference requests, under our optimal policy and the considered benchmarks (N=100).

requirements. The first scenario, named *ShortH&HighA*, focuses on high accuracy, A_n, thus requiring the use of the full model, with a short prediction horizon, i.e., H=1 hour. The second scenario, named *ShortH&VarA*, still considers a short time horizon while exploring the potential benefits of relaxed accuracy constraints. The required model accuracy is generated randomly within the [78%, 90%] range. This allows to investigate the possibility of using pruned models that may offer faster execution times while still maintaining acceptable accuracy levels. The third scenario, named *LongH&VarA*, builds upon *ShortH&VarA* scenario with the same range of accuracy requirements, but the prediction horizon, H, is increased by a factor of 4. In all three scenarios, the maximum inference completion time, L_n, is set so as to ensure that each inference request is fulfilled before the subsequent one arrives (L_n=H/Q_n). The battery capacities of devices are set within the [3.7 -18.5 Wh] range [17]–[19]. The number of devices is set to 100.

Benchmarks and metrics. The benchmark schemes considered in this work are adapted from [20], which investigates the inference execution of full or pruned models either at the *edge* server only or at the mobile *device* only. By referring to such extreme cases, these benchmarks upper/lower bounds for the following key metrics: average *time required to fulfill a single inference request*, average *amount of data exchanged over the network*, and average *energy consumed by the device* throughout the ML pipeline, with the latter two both measured per device and per the entire stream of requests.

C. Numerical results

We begin by examining the offloading decisions of our optimal policy under the three considered scenarios.

In the *ShortH&HighA* scenario (Fig. 2 (left)), executing inferences at the edge emerges initially as the best choice. Indeed, for a small number of requests, the overhead of transmitting a few samples to the edge is lower than the cost of transmitting the full model to each device. When the number of requests (Q_n) reaches 60, performing inference at the device side becomes more advantageous in terms of the exchanged data. However, as the number of requests (Q_n) further increases, the delay and the battery capacity become the dominant factors (see (19) and (20), respectively). We recall that such a delay encompasses the queuing time at both the edge server and the device, and it is influenced by both the prediction horizon and the need to use the full model. As Q_n grows (and with it the arrival rate of inference requests per device, for a fixed H), the queuing delay at the device inevitably increases due to limited processing capacity, thus leading to the full utilization of the edge server.

In the *ShortH&VarA* case (Fig. 2 (center)), with a relaxed accuracy constraint, executing inferences directly on the device becomes more efficient for a moderate number of inference requests, namely, $7<Q_n<200$. The reason is that the reduced size of pruned models outweighs the overhead of transmitting multiple requests to the edge. However, as Q_n further increases, the delay and the battery again become the limiting

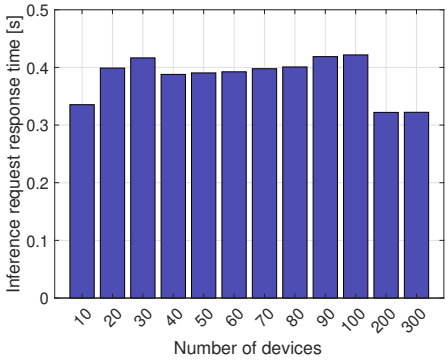


Fig. 4: Response time per inference request versus number of devices, under our optimal policy, *ShortH&VarA*, $Q_n = 10$.

factors, leading to first a decrease in the fraction of inferences run at the device; and eventually, all inferences are executed at the edge ($300<Q_n<500$).

Finally, in *LongH&VarH* scenario (Fig. 2 (right)), for $Q_n>2$, the system can accommodate all inference requests at the device. This is due to the combined effect of less stringent accuracy constraints, which allows to select more pruned models and of a longer prediction horizon, which, in its turn, implies a lower arrival rate of inference requests.

Regarding the pruning factor, in the *ShortH&HighA* scenario, the full model is always used due to the accuracy constraint. In the *ShortH&VarA* and *LongH&VarA* cases, again, the full model or a model with 10% pruning factor are selected for a small number of requests (when only the edge server is used). However, as the number of inference requests increases and more inference tasks are run at the devices, more heavily pruned models are employed.

Our optimal placement, in most cases, demonstrates a reduced amount of transmitted data as well as device consumed energy compared to our benchmarks, as illustrated in Fig. 3. Although some benchmark schemes (namely, those executing inference on the device only) imply less exchanged data, they often violate either the accuracy, or battery constraint, or all. In contrast, our solution always closely approximates the energy consumption yielded by the benchmarks where inference is executed at the edge server.

Finally, in Fig. 4, we fix the number of inference requests to 10 and show the average delay of a single inference request when varying the number of devices, under the *ShortH&VarH* scenario. The metric keeps below 0.5s and scales quite well, confirming the capability of the proposal to properly select the model version and the placement for its execution.

V. Conclusions and future Work

We investigated the optimal placement of inference tasks and selection of the DNN model configurations across the mobile device-edge continuum. To this end, we proposed an optimization framework with the primary objective of minimizing data transfer over the device-edge radio link, subject to application requirements and system constraints. We then assessed the performance of the proposal against different benchmarks under different application scenarios.

Our results demonstrate that: *(i)* Edge server offloading is advantageous for a small stream of inference requests, since, in such cases, the overhead of transmitting a few samples to the edge is lower than that of transmitting the full model to each device; *(ii)* Edge server offloading is also beneficial for scenarios with strict accuracy requirements; *(iii)* For a moderate inference task rate and relaxed accuracy requirements, on-device inference execution using pruned models can be more efficient than offloading.

Future work will address the design of a low-complexity heuristics that can still effectively solve the investigated problem, while ensuring a very swift solution.

References

[1] C. Campolo *et al.*, "Network for Distributed Intelligence: a Survey and Future Perspectives," *IEEE Access*, vol. 11, pp. 52840 – 52861, 2023.

[2] S. Dong *et al.*, "Task offloading strategies for mobile edge computing: A survey," *Computer Networks*, p. 110791, 2024.

[3] G. Singh *et al.*, "Compressing and Fine-tuning DNNs for Efficient Inference in Mobile Device-Edge Continuum," in *IEEE MeditCom*, pp. 305–310, 2024.

[4] D. Kafetzis *et al.*, "DNN Partitioning and Inference Task Offloading in 6G Resource-Constrained Networks," in *EuCNC/6G Summit*, pp. 753–758, 2024.

[5] Z. Xu *et al.*, "Energy-aware inference offloading for DNN-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2020.

[6] A. B. Sada *et al.*, "Selective Task offloading for Maximum Inference Accuracy and Energy efficient Real-Time IoT Sensing Systems," *arXiv preprint arXiv:2402.16904*, 2024.

[7] L. Piano *et al.*, "Detecting drift in deep learning: A methodology primer," *IT Professional*, vol. 24, no. 5, pp. 53–60, 2022.

[8] X. Xu *et al.*, "Adaptive Computation Offloading with Edge for 5G-Envisioned Internet of Connected Vehicles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5213–5222, 2020.

[9] T. K. Rodrigues *et al.*, "Offloading Decision for Mobile Multi-Access Edge Computing in a Multi-Tiered 6G Network," *IEEE Trans. on Emerging Topics in Computing*, vol. 10, no. 3, pp. 1414–1427, 2021.

[10] O. Chukhno *et al.*, "A Holistic Assessment of Directional Deafness in mmWave-based Distributed 3D Networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7491–7505, 2022.

[11] J. Yan *et al.*, "Pricing-Driven Service Caching and Task Offloading in mobile Edge Computing," *IEEE Trans. on Wireless Communications*, vol. 20, no. 7, pp. 4495–4512, 2021.

[12] F. Yu *et al.*, "Easiedge: A novel global deep neural networks pruning method for efficient edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1259–1271, 2020.

[13] X. Li *et al.*, "Automated Service Provisioning and Hierarchical SLA Management in 5G Systems," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4669–4684, 2021.

[14] C. Puligheddu *et al.*, "SEM-O-RAN: Semantic O-RAN Slicing for Mobile Edge Offloading of Computer Vision Tasks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 7, pp. 7785–7800, 2023.

[15] GUROBI OPTIMIZATION, LLC, "UNMATCHED PERFORMANCE. Solve Faster. Explore countless possible solutions, in seconds, with the world's fastest solver." [Online] (accessed January 18, 2025) https://www.gurobi.com/features/gurobi-optimizer-delivers-unmatched-performance/, 2025.

[16] J. Frankle *et al.*, "Pruning Neural Networks at Initialization: Why Are We Missing the Mark?," *arXiv preprint arXiv:2009.08576*, 2020.

[17] EEMB Battery, "Datasheet: LP603449-Standard Type Lithium Polymer Battery." [Online] (accessed January 18, 2025) https://www.eemb.com/product-147, 2023.

[18] Microsoft, "Datasheet: Lumia 950." [Online] (accessed January 18, 2025) https://news.microsoft.com/wp-content/uploads/2016/04/Lumia-950-Datasheet.pdf, 2023.

[19] BatterySpace.com, "Datasheet: LiFePO4 battery." [Online] (accessed January 18, 2025) https://www.batteryspace.com/polymerli-ionbattery74v2500mah185whwithpcb.aspx, 2023.

[20] O. Chukhno *et al.*, "Machine Learning Performance at the Edge: When to Offload an Inference Task," in *Workshop on Networked Sensing Systems for a Sustainable Society*, pp. 180–186, 2023.