IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 21, 2021, accepted March 9, 2021, date of publication March 17, 2021, date of current version March 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066559

Wireless Edge Machine Learning: Resource Allocation and Trade-Offs

MATTIA MERLUZZI, (Graduate Student Member, IEEE), PAOLO DI LORENZO, (Senior Member, IEEE), AND SERGIO BARBAROSSA, (Fellow, IEEE)

Department of Information Engineering, Electronics, and Telecommunications, Sapienza University, 00184 Rome, Italy

Corresponding author: Mattia Merluzzi (mattia.merluzzi@uniroma1.it)

This work was supported in part by the H2020 EU/Taiwan Project 5G CONNI under Grant 861459, and in part by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) through the Progetti di Rilevante Interesse Nazionale (PRIN) Liquid Edge contract.

ABSTRACT The aim of this paper is to propose a resource allocation strategy for dynamic training and inference of machine learning tasks at the edge of the wireless network, with the goal of exploring the trade-off between energy, delay and learning accuracy. The scenario of interest is composed of a set of devices sending a continuous flow of data to an edge server that extracts relevant information running online learning algorithms, within the emerging framework known as Edge Machine Learning (EML). Taking into account the limitations of the edge servers, with respect to a cloud, and the scarcity of resources of mobile devices, we focus on the efficient allocation of radio (e.g., data rate, quantization) and computation (e.g., CPU scheduling) resources, to strike the best trade-off between energy consumption and quality of the EML service, including service end-to-end (E2E) delay and accuracy of the learning task. To this aim, we propose two different dynamic strategies: (i) The first method aims to minimize the system energy consumption, under constraints on E2E service delay and accuracy; (ii) the second method aims to optimize the learning accuracy, while guaranteeing an E2E delay and a bounded average energy consumption. Then, we present a dynamic resource allocation framework for EML based on stochastic Lyapunov optimization. Our low-complexity algorithms do not require any prior knowledge on the statistics of wireless channels, data arrivals, and data probability distributions. Furthermore, our strategies can incorporate prior knowledge regarding the model underlying the observed data, or can work in a totally data-driven fashion. Several numerical results on synthetic and real data assess the performance of the proposed approach.

INDEX TERMS Edge machine learning, multi-access edge computing, computation offloading, stochastic optimization, resource allocation, energy-latency-accuracy trade-off.

I. INTRODUCTION

We live at the edge of a new revolution, characterized by a massive growth of data traffic and a pervasive introduction of artificial intelligence tools aimed to extract meaning from the data. 5G networks provide an efficient way to enable many different new services using a single communication platform. The efficient deployment of 5G (and beyond) communication technologies is leading to an ever deeper synergy among communication, computation, control, and content delivery [1]. The next generation of wireless networks will hinge on two main thrusts: i) a significant communication performance enhancement, building on three main pillars: ultra-reliable and low-latency communications (URLLC), enhanced mobile broadband (EMBB), and Massive Machine Type Communications (mMTC) [2]; ii) a pervasive deployment of cloud capabilities at the wireless network edge, to enable a plethora of services for different sectors (*verticals*), such as Industry 4.0, Internet of Things (IoT), autonomous driving, remote surgery, etc. This paradigm is well-known under different names in the literature, such as Edge Computing, or Fog Computing. An example of standardization of the main functionalities and interfaces is carried out by the European Telecommunications Standards Institute (ETSI), under the name of Multi-Access Edge Computing (MEC).[1] MEC offers cloud computing capabilities,

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Tsun Cheng.

[1] https://www.etsi.org/technologies/multi-access-edge-computing

albeit limited, at the edge of the network, typically within the Radio Access Network or at an aggregation point of the core network. In particular, recent surveys on 5G architectures place the MEC functionalities and facilities behind the User Plane Function (UPF) [3]. Thus, the main advantage of MEC is its proximity to the end users, which enables low end-to-end (E2E) latency services. For a recent survey of MEC in 5G and beyond systems, the interested reader can refer to [4]. Furthermore, the convergence of communication, computation and control, is fundamental to enable mission critical applications in many scenarios. As an example, 5G and beyond networks, aided by edge computing, are foreseen to enable the industrial automation in real-time,[2] within low E2E delay, extremely high reliability, possibly with an energy efficient perspective to reduce the global carbon footprint of the ICT industry [5].

In this new heterogeneous context, the notions of latency and reliability need to evolve from classical communication-related concepts. The traditional definition of E2E delay takes into account the overall latency from the transmission of a packet until its successful decoding at the receiver. However, since future services will involve communication *and* computation, the E2E delay must take into account the time elapsing from the generation of a new request/data unit/task by a peripheral device, to the time in which the edge server completes the computations necessary to fulfill the request. In parallel, also the definition of reliability needs a revision. From a pure communication perspective, reliability is associated to the probability of successfully decoding the received packets. However, whenever the goal of communication is that the Edge Server (ES) is able to take decisions about the data transmitted by the peripheral devices, a new definition of reliability should be adopted, associated to the reliability of the decisions taken on the basis of the received packets. This opens a new perspective calling for a holistic vision building on the integration of communication, computation, caching, and control.

To be more specific, one of the key applications of edge computing is the development of Machine Learning (ML) algorithms that run at the edge, e.g., in close proximity of the industrial facilities for different purposes such as control decision making, anomaly detection, monitoring and maintenance. The new paradigm of integrating wireless networks with ML at the edge is known in the literature as *Edge Machine Learning* (EML). In the EML context, it is important to control not only the reliability from the communication point of view, but also from the computation point of view, assessing the accuracy of the decisions taken by the edge server, which can involve fulfilling tasks such as prediction, estimation, classification, and so on. In a nutshell, the goal of EML is to devise resource allocation strategies that enable machine learning at the wireless network edge with low energy consumption, low E2E delay and high learning/inference accuracy. It is then clear that enabling edge machine learning introduces novel fundamental problems in terms of *jointly optimizing* communication (e.g., power, bits, source encoding, etc.), computation (e.g., CPU cycles, number of active servers/cores, etc.), and inference/training (e.g., choice and splitting of a (deep) learning architecture, model and/or data partitioning, etc.) *to meet the system constraints* (e.g., E2E delay, reliability, energy) while *guaranteeing a prescribed performance* of the inference task. The main contribution of this paper is to propose a joint optimization leading to a control action, to be performed in real-time, whose goal is to strike an optimal trade-off between energy, delay and accuracy, while coping with the time-variability of radio channels, data arrivals, computation loads, memory availability, etc. In particular, as illustrated in Fig. 1, it is possible to consider different trade-offs among the main variables involved in EML, e.g., the trade-off between energy consumption and delay, accuracy and delay, energy consumption and accuracy, or their joint combination. In this way, the performance requirements (e.g., accuracy, convergence rate, etc.) of the learning task play a key role in selecting the best allocation of radio and computation resources.

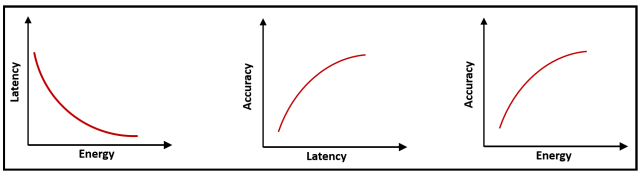


FIGURE 1. Edge Machine Learning trade-offs.

A. RELATED WORKS

In the last few years, there has been a huge interest in edge computing, from communication to computation and caching perspectives, as well as the investigation of a tight integration of the computing paradigm in the context of wireless communication networks [4], [6], [7]. Since our work focuses on dynamic resource allocation strategies for computation offloading of machine learning tasks, in the sequel we review the general literature on dynamic computation offloading and the recent advances in EML.

1) COMPUTATION OFFLOADING

The goal of computation offloading is to move the execution of computationally heavy applications from mobile or IoT devices to nearby edge servers. The motivation for using computation offloading is threefold: i) empower simple IoT devices with superior computational capabilities, as available at the server; ii) reduce the energy consumption at the resource-hungry mobile devices; iii) reduce E2E service delay, whenever the sum of transmission and computation time at the edge is smaller than the computation time at the mobile device. In particular, dynamic computation offloading refers to applications that continuously generate tasks or data to be sent to the edge server for processing. A typical example is the continuous sensor data acquisition, with the aim of performing real time data analytics for different purposes,

[2]H2020 EU/Taiwan Project 5G CONNI, https://5g-conni.eu/

such as anomaly detection or prediction. Several works investigate the problem of radio and computation resource allocation for dynamic computation offloading [8]–[17]. In [8], a dynamic formulation is proposed, with a strategy based on Lyapunov optimization in a cloud computing framework. In [9], the authors consider a fog-enabled Device to Device (D2D) scenario and propose a strategy for the mutual association of mobile devices to offload tasks among each other. User assignment is also addressed in [10], with the goal of optimizing the average delay under energy constraints, with a penalty function that discourages frequent handovers. The assignment strategy is based on a multi-armed bandit algorithm to learn the optimal association. In [11], we propose a dynamic computation offloading algorithm to jointly optimize computation and communication resources and mobile users assignment to APs and edge servers, merging tools from stochastic optimization and matching theory. In [12], a Lyapunov based strategy is proposed, for the joint optimization of radio and computation resources, to minimize the users' energy consumption under E2E delay constraints. In [15], the authors investigate a scenario with multiple APs and edge servers, where an assignment strategy based on matching theory is proposed, coupled with the tools of Lyapunov optimization and Extreme Value Theory to control reliability. The interested reader is referred to the recent surveys related to MEC and computation offloading [4], [6], [18], [19].

All the aforementioned works present general formulations for computation offloading, without explicitly taking into account the requirements of the offloaded tasks. They mainly focus on energy efficiency with latency guarantees and/or reliability over the wireless interface, with a high level and general description of the application in terms of computational requests, but without investigating the requirements associated to the application layer, e.g., the accuracy of the learning tasks to be offloaded.

2) EDGE MACHINE LEARNING

A first general introduction to EML can be found in [20], where the authors present several possible trade-offs. Other recent general surveys can be found in [21]–[23]. Going into more specific contributions, the authors of [24] consider an edge machine learning system, where an edge processor runs an algorithm based on Stochastic Gradient Descent (SGD). In particular, they investigate the trade-off between latency and accuracy, by optimizing the packet payload size, given the overhead of each data packet transmission and the ratio between the computation and communication rates. In [25], the authors propose an algorithm to maximize the learning accuracy under latency constraints, while the authors of [26] present a distributed machine learning algorithm at the edge, where wireless devices collaboratively minimize an empirical loss function with the help of a remote server. The authors of [27] propose a communication-efficient decentralized machine learning algorithm that dynamically optimizes a stochastic quantization method, with applications to regression and image classification. The authors of [28] consider generic distributed machine learning algorithms at the edge, based on SGD, investigating the trade-off between local update and global aggregation. In [29], the authors present a data compression algorithm to reduce the communication burden and energy consumption of an IoT network, to enable machine learning with a desired target accuracy. Finally, an important research topic related to EML is *federated learning* (FL) [30]–[36]. In FL, multiple edge devices perform local model updates on collected data, and the server then takes a weighted average of the resulting models. Since no data are sent to the ES, but only local gradient updates, an energy efficient, low latency, privacy preserving training at the edge is enabled. In [31], two update methods to reduce the uplink communication costs for FL are proposed. In [34], the authors present a practical update method for a deep FL algorithm with an extensive empirical evaluation for different FL models. Reference [36] investigates the problem of joint power and resource allocation for ultra-reliable low latency communication in vehicular networks. The interested reader can refer to [30] for a comprehensive survey on FL.

B. CONTRIBUTION

In this work, we propose a dynamic algorithmic framework, whose goal is to strike an optimal balance between energy consumption (both for communication and computation), E2E delay, and learning/inference performance, enabling training and inference tasks at the edge of the wireless network. Differently from previous works on computation offloading and EML, we assume a *goal-oriented* communication perspective, where the scope of the communication is not necessarily to convey all bits reliably within a given time constraint, but to send enough data to enable the edge server to take decisions with the desired accuracy, thus striking the best trade-off between energy consumption, E2E delay and accuracy. To achieve this goal, we dynamically act on the source encoder to adjust the transmission rate, while still fulfilling the goal of the learning task. The idea is to tolerate a small amount of distortion on the received data, to achieve a better energy-delay trade-off, but still being able to satisfy the accuracy requirements of the learning task. In particular, we focus on two different resource allocation strategies:

1) **Minimum energy under latency and learning performance constraints.** For this first class of algorithms, we consider two different sub-classes:
   - *Model-based EML*: In this case, we exploit the fact that, for some learning algorithms and data models, it is possible to provide closed form expressions for the accuracy, which can be used to seek the minimum energy strategy with guaranteed E2E delay and learning performance constraints;
   - *Data-driven EML*: In this case, without assuming any model for the data, we hinge on performance metrics that can be practically measured online, to devise a dynamic method that aims to minimize

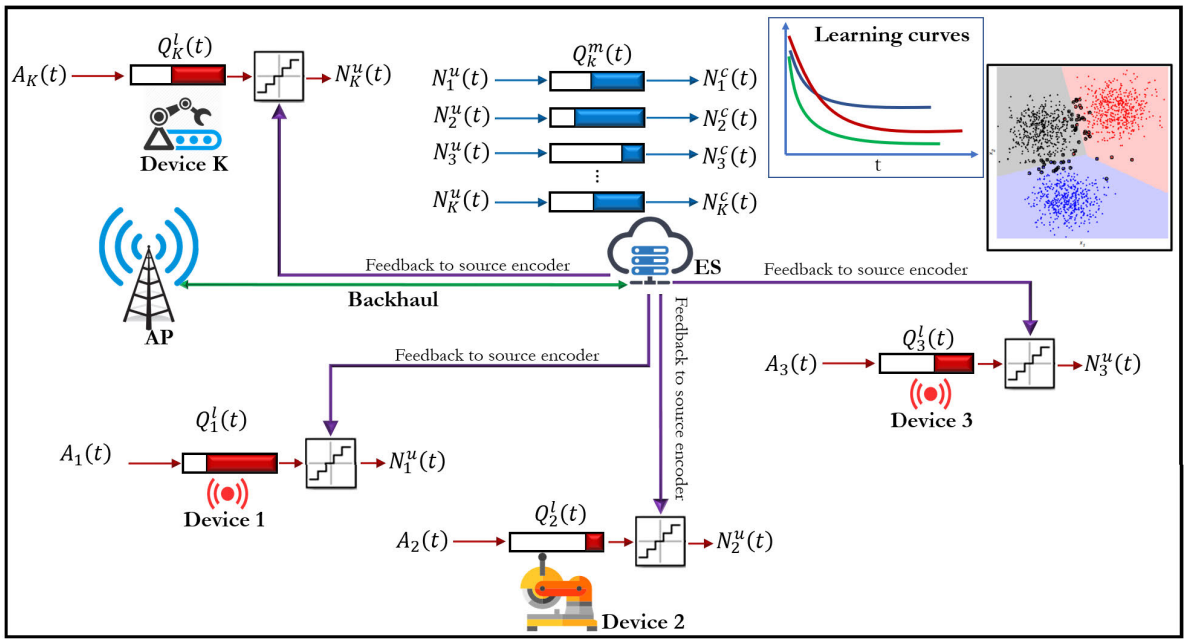


FIGURE 2. Network model: Sensors offloading data via the Access Point (AP) to the Edge Server (ES). The ES runs the learning/inference task (e.g. estimation, prediction, classification) and feeds the quantization levels back to the sensors.

the energy consumption under E2E delay and learning guarantees. Indeed, for some learning tasks such as estimation and prediction, the accuracy (e.g. the Mean Squared Error), can be estimated online from the data within a limited delay.

2) **Best learning/inference performance under latency and energy constraints.** In this case, we assume that no model is available, and the performance cannot be measured online. This is typical of some learning tasks such as, e.g., classification. In this case, it might be impossible to set a constraint on the learning performance, and it is more convenient to rely on a best accuracy strategy, under E2E delay and energy constraints, which are strongly related to application requirements and physical needs (e.g., battery lifetime).

For the first class of problems (i.e., minimum energy), we present an application involving estimation/prediction based on Least Mean Squares (LMS), using a synthetic and a real dataset. For the second class of problems (i.e., best learning accuracy) we provide results on classification over two different real datasets, involving a Support Vector Machine (SVM) and a Neural Network (NN). The simulations are carried out over both synthetic and real data sets to show how, without any prior knowledge on the statistics of radio channels, pattern arrivals, and data distributions, the proposed methods are able to strike the desired trade-off between energy, latency and learning/inference accuracy.

C. OUTLINE

The paper is organized as follows: in Sec. II we present the system model, comprising energy consumption, delay and learning accuracy; in Sec. III, we introduce the minimum energy strategy, starting from the problem formulation and then presenting the algorithmic solution; in this case, we consider both a model-based and a data-driven approach. Similarly, in Sec. IV we present the best accuracy strategy with E2E delay and energy constraints. In Sec. V, we customize our frameworks to LMS estimation, SVM and NN classification, showing numerical results for both resource allocation strategies. Finally, Sec. VI draws some conclusions and future directions.

II. SYSTEM MODEL

In this section, we present the system model. In particular, we first present the energy consumption model, both for the communication side (devices' energy consumption), and for the computation side (ES's energy consumption). Then, we present the performance metrics used throughout the paper, namely latency and learning accuracy. We consider a scenario with K sensors and an Access Point (AP) equipped with an ES, as illustrated in Fig. 2. Each sensor captures data from the environment and uploads them to the server through the wireless connection with the AP. The server collects data and runs a learning/inference algorithm that requires certain performance in terms of E2E delay and accuracy.

A. ENERGY CONSUMPTION

In the sequel, we illustrate the model used to quantify the energy consumption of mobile devices/sensors and of the ES running the EML tasks.

1) DEVICE ENERGY CONSUMPTION

Since we deal with a dynamic scenario, we consider time as organized in time slots of equal duration τ. A generic sensor

device k, in each time slot t, transmits a certain number of data depending on its data rate $R_k(t)$ (expressed in bit/sec), as it will be detailed later on in this section. Then, inverting the well-known Shannon capacity formula, the power spent for transmission during time slot t is given by

$$p_k(t) = \frac{B_k N_0}{h_k(t)} \left(\exp\left(\frac{R_k(t) \ln(2)}{B_k} \right) - 1 \right), \tag{1}$$

where B_k is the bandwidth allocated to sensor k, $h_k(t)$ is the time varying channel power gain, and N_0 represents the noise power spectral density at the receiver. Then, the overall sensor energy consumed during time slot t is given by

$$E_d(t) = \tau \sum_{k=1}^{K} p_k(t). \tag{2}$$

2) EDGE SERVER ENERGY CONSUMPTION

The dynamic energy consumption of a CPU is highly dependent on its clock frequency $f_c(t)$, which we assume to be dynamically scaled, when possible, to reduce the energy consumption of the processor [37], [38]. In particular, we assume that $f_c(t)$ can be selected from a discrete finite set $\mathcal{F}$, with a maximum CPU cycle frequency denoted by $f_{\max}$, and we exploit a widely used cubic model for the energy consumption, described as

$$E_s(t) = \tau \kappa f_c^3(t), \tag{3}$$

where κ is the effective switched capacitance of the processor [37], [38]. Furthermore, since we consider a multiuser scenario, where the edge server has not the virtually infinite computational capabilities of a central cloud, we assume that the CPU time is shared across the tasks required by each device. This is equivalent to allocate a portion $f_k(t)$ of the CPU clock frequency $f_c(t)$ to each agent k, while imposing

$$\sum_{k=1}^{K} f_k(t) \leq f_c(t).$$

Finally, from (2) and (3), the total system energy consumption at time t is given by

$$E_{\text{tot}}(t) = E_d(t) + E_s(t). \tag{4}$$

B. DELAY AND QUEUEING MODEL

In this paper, we consider a continuous flow of data that are generated locally by the sensor devices, and uploaded to the ES, which processes them by running an online learning algorithm. Then, the overall delay experienced by a data unit from its generation to its processing at the ES is given by the sum of: i) the uplink queueing delay, ii) the transmission delay, iii) the queueing delay at the ES, and iv) the computation time at the ES. Thus, proceeding as in [12], [39], we define an uplink transmission queue $Q_k^l(t)$, and a remote queue $Q_k^m(t)$ of data to be processed at the ES. The uplink queue is fed by the new task arrivals, and drained by the uplink data transmission. Since the goal of our system is to accomplish tasks, e.g. to perform image recognition, we need to identify the dimension of the smallest data unit that can be processed singularly. For example, in image processing, the data unit is one image. Each task is characterized by the following quantities: the amount of samples composing each data unit to be processed and the amount of CPU cycles necessary to process each data unit. We denote by M_k the number of samples in one data unit, for example the number of features extracted from a data set or the number of pixels in an image. In our dynamic resource allocation strategy, we adapt the number of bits per sample in order to find the best trade-off between energy consumption, service delay and learning accuracy. Denoting by $n_k^q(t)$ the number of bits per sample used by device k in time slot t, a transmitted data unit is represented by $M_k n_k^q(t)$ bits. We assume here, for simplicity, that the data to be quantized are statistically independent, as they are the result of a source encoder that has removed any unnecessary redundancy. Furthermore, we assume that the granularity with which we adapt the quantization level is the time slot, so that within each time slot $n_k^q(t)$ is constant, i.e. all the data units transmitted in the same slot are quantized with the same number of bits. Hence, since in each slot we transmit a number of bits equal to $\tau R_k(t)$, the number of data units/tasks transmitted during time slot t is

$$N_k^u(t) = \left\lfloor \frac{\tau R_k(t)}{M_k n_k^q(t)} \right\rfloor, \tag{5}$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than x. Then, the local queue, indicating the number of tasks to be completed, evolves as follows:

$$Q_k^l(t+1) = \max\left(0, Q_k^l(t) - N_k^u(t)\right) + A_k(t) \tag{6}$$

where $A_k(t)$ denotes the new data arrivals, for example the number of images generated in time slot t. The arrivals are assumed to be random with unknown statistics. The role played by the quantization level will be clear later on in this section, when we will introduce the accuracy of the edge machine learning task.

At the server side, the remote queue is fed by the uplink task arrivals, and it is drained by the task computation. In this work, proceeding as in [12], we assume that there exists a linear relation between the data units/tasks, and the number of CPU cycles necessary to run the task. Thus, denoting by $1/J_k$ the number of CPU cycles necessary to process one data unit/task, the number of data units processed in slot t is

$$N_k^c(t) = \lfloor \tau f_k(t) J_k \rfloor. \tag{7}$$

Then, the computation queue at the server side, counting the number of tasks to be executed, evolves then as follows:

$$Q_k^m(t+1) = \max(0, Q_k^m(t) - N_k^c(t)) + \min\left(Q_k^l(t), N_k^u(t)\right). \tag{8}$$

The overall service delay is directly related to the sum of the local and the computation queues

$$Q_k^{\text{tot}}(t) = Q_k^l(t) + Q_k^m(t).$$

In fact, from Little's law [40], given a data arrival rate $\bar{A}_k = \mathbb{E}\{A_k(t)/\tau\}$ and a stationary queueing system, the overall long-term average latency experienced by a new data unit from its generation to its computation at the ES is

$$\bar{D}_k^{\infty} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left\{ \frac{Q_k^{\text{tot}}(t)}{\bar{A}_k} \right\}, \quad \forall k \tag{9}$$

where the expectation is taken with respect to the radio channel and data arrival statistics. Our goal is to guarantee a long-term average delay constraint D_k^{avg}, which can be written as follows

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left\{ Q_k^{\text{tot}}(t) \right\} \leq Q_k^{\text{avg}}, \quad \forall k, \tag{10}$$

where $Q_k^{\text{avg}} = D_k^{\text{avg}} \bar{A}_k$. Although $\bar{A}_k$ is unknown a priori, it can be estimated online.

C. LEARNING ACCURACY

The metric used to quantify the inference accuracy depends on the specific learning task (e.g., prediction, estimation, or classification), and on the adopted machine learning algorithm (e.g. Least Mean Squares, Support Vector Machine, Neural Networks, etc.). Since we aim to control the accuracy of the learning task, we introduce an instantaneous performance metric $G_k(t) = G_k(n_k^q(t))$, which depends on the number of quantization bits used to represent the data and then it is also a function of the time slot index t. Intuitively, the larger is the number of quantization bits, the better will be the learning performance. Given the instantaneous performance $G_k(t)$, we can impose a long-term constraint on the learning accuracy as follows:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{G_k(t)\} \leq \mathcal{G}_k^{\text{avg}}, \quad \forall k. \tag{11}$$

$G_k(t)$ may represent, for example, the missclassification rate in a classification task, the Mean Squared Error (MSE) or Mean Squared Deviation (MSD), in an estimation or prediction task, etc. In Secs. III and IV, we will keep this function generic on purpose, so that it can be suitably adapted to the specific edge learning task we are interested in. In more specific cases, when a data model is available, the expression of $G_k(t)$ is known and then it can be directly exploited in the optimization (see Sec. III.A); otherwise, the value of $G_k(t)$ can be estimated from the data and used to control the system in a fully data-driven fashion (see Sec. III.D). In Sec. V, we will customize the proposed framework to some specific learning tasks, showing how to set and update $G_k(t)$ for both model-based and data-driven strategies.

As mentioned before, typically $G_k(t) = G_k(n_k^q(t))$ is a function of the number of quantization bits n_k^q, since a coarser representation of the data can lead to deteriorated performance of the learning task. However, on the other hand, using a finer quantization level yields more bits to be transmitted and then a higher energy consumption at the transmit side (cf. (1),(5),(6)) to meet the desired latency constraint. In this work, we control the learning/inference accuracy acting on the source encoder, at the transmit side, and then on the number of quantization bits used to encode the data. To enforce this strategy, we introduce a feedback loop, as depicted in Fig. 2, from the edge server to the source encoder, to feed back the information about the number of quantization bits, in order to find the desired balance between energy consumption, E2E delay and learning accuracy.

III. MINIMUM ENERGY UNDER E2E DELAY AND ACCURACY CONSTRAINTS

In this section, we introduce the first dynamic resource allocation strategy for EML. Our goal is to devise an online strategy striking a good trade-off between energy, latency, and learning accuracy. To this aim, we formulate the following long-term average optimization problem:

$$\min_{\mathbf{\Phi}(t)} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left\{E_{\text{tot}}(t)\right\}$$

subject to

$$\begin{aligned}
&(a) \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left\{ Q_k^{\text{tot}}(t) \right\} \leq Q_k^{\text{avg}}, \quad \forall k; \\
&(b) \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{G_k(t)\} \leq \mathcal{G}_k^{\text{avg}}, \quad \forall k; \\
&(c) \quad 0 \leq R_k(t) \leq R_{k,\max}(t), \qquad \forall k, t; \\
&(d) \quad n_k^q(t) \in \mathcal{N}_k^q, \qquad \forall k, t; \\
&(e) \quad f_c(t) \in \mathcal{F}, \qquad \forall t; \\
&(f) \quad f_k(t) \geq 0, \quad \forall k, t; \\
&(g) \quad \sum_{k=1}^{K} f_k(t) \leq f_c(t), \qquad \forall t;
\end{aligned} \tag{12}$$

where $\mathbf{\Phi}(t) = \left[\{R_k(t)\}_k, \{n_k^q(t)\}_k, \{f_k(t)\}_k, \{f_c(t)\}\right]$. The constraints of (12) have the following meaning: (a) the long-term average E2E delay experienced by a data unit from its generation to its computation at the server must be smaller than a predefined threshold[3]; (b) the long-term average of the function quantifying the accuracy of the learning task must be smaller than a predefined threshold; (c) the data rate of each device is non negative and is lower than a maximum value obtained by plugging the maximum transmit power budget $p_k^{\max}$ into the Shannon formula; (d) the number of quantization bits is selected from a discrete set $\mathcal{N}_k^q$; (e) the CPU clock frequency is selected from a finite set $\mathcal{F}$; (f) the CPU cycle frequency assigned to each device k is non negative; (g) the sum of the CPU cycle frequencies assigned to all devices cannot exceed the CPU clock frequency. Problem (12) is very difficult to solve, especially because of the lack of knowledge

[3]Note that more sophisticated probabilistic constraints can also be imposed on the maximum tolerable delay [12].

of the statistics of the radio channels and task arrivals. Nevertheless, in the sequel, we show how to tackle the problem effectively, hinging on stochastic Lyapunov optimization.

A. STOCHASTIC LYAPUNOV OPTIMIZATION

The first step is to define a dynamic strategy to handle the long-term constraints (*a*) and (*b*) of (12). Proceeding as in [41], we introduce *virtual queues* to control the long term constraints (a) and (b). More specifically, we introduce the virtual queue $Z_k(t)$ for each device, to control latency, evolving as

$$Z_k(t+1) = \max\left(0, Z_k(t) + Q_k^{\text{tot}}(t+1) - Q_k^{\text{avg}}\right), \quad (13)$$

and the virtual queue $Y_k(t)$ for each device, to control accuracy, evolving as

$$Y_k(t+1) = \max\left(0, Y_k(t) + \nu_k\left(G_k(t) - \mathcal{G}_k^{\text{avg}}\right)\right), \quad (14)$$

where ν_k is a step size used to control the convergence of the algorithm.[4] The aim of the virtual queues is to keep track of how the system is behaving in terms of constraint violations. In particular, it can be shown that, imposing the mean rate stability of the virtual queues (13) and (14), is equivalent to ensuring constraints (*a*) and (*b*) of (12), respectively [41]. The mean rate stability of $Z_k(t)$ and $Y_k(t)$ is defined as follows [41]:

$$\lim_{T\to\infty}\frac{\mathbb{E}\{Z_k(T)\}}{T} = 0, \quad \lim_{T\to\infty}\frac{\mathbb{E}\{Y_k(T)\}}{T} = 0, \quad \forall k.$$

Using the general framework of [41], to impose the mean rate stability, we introduce now the Lyapunov function $L(\mathbf{\Theta}(t))$, defined as:

$$L(\mathbf{\Theta}(t)) = \frac{1}{2}\sum_{k=1}^{K}\left[Z_k^2(t) + Y_k^2(t)\right],$$

where $\mathbf{\Theta}(t) = \left[\{Z_k(t)\}_{k=1}^K, \{Y_k(t)\}_{k=1}^K\right]$. Having defined the Lyapunov function, we can introduce the conditional Lyapunov drift, which is the conditional expected change of the Lyapunov function over one slot:

$$\Delta(\Theta(t)) \triangleq \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t))|\Theta(t)\}. \quad (15)$$

It can be shown that, minimizing (15), the mean rate stability of the queues is imposed, so that constraints (*a*) and (*b*) of (12) are satisfied. However, directly minimizing the conditional Lyapunov drift can lead to an unnecessary system energy consumption. Then, as in [41], we introduce the *drift-plus-penalty* function, defined as

$$\Delta_p(\Theta(t)) = \Delta(\Theta(t)) + V\cdot\mathbb{E}\left\{E_{\text{tot}}(t)\middle|\Theta(t)\right\}, \quad (16)$$

where V is a control parameter used to assign more importance to the energy term with respect to the virtual queue backlogs. Now, instead of directly minimizing (16), we proceed by minimizing the following upper bound (the derivations are given in the Appendix):

$$\begin{aligned}\Delta_p(\mathbf{\Theta}(t)) \le \zeta + \mathbb{E}\Bigg\{\sum_{k=1}^{K}\Big[&\chi_k(t) - 2Q_k^l(t)N_k^u(t)\\ &+ 2Q_k^m(t)(N_k^u(t) - N_k^c(t))\\ &+ Z_k(t)\Big(\max\left(0, Q_k^l(t) - N_k^u(t)\right)\\ &+ \max(0, Q_k^m(t) - N_k^c(t))\Big)\\ &+ \nu_k Y_k(t)G_k(t)\Big] + VE_{\text{tot}}(t)\Big|\mathbf{\Theta}(t)\Bigg\}\end{aligned} \quad (17)$$

where ζ is a positive constant given by

$$\begin{aligned}\zeta = \frac{1}{2}\sum_{k=1}^{K}\Big[&2\left(A_k^{\max}\right)^2 + 4\left(N_{k,\max}^u\right)^2 + 2(N_{k,\max}^c)^2\\ &+ (Q_k^{\text{avg}})^2 + \nu_k^2\left(G_k^{\max} - \mathcal{G}_k^{\text{avg}}\right)^2\Big],\end{aligned} \quad (18)$$

and $\chi_k(t)$ is defined as follows:

$$\begin{aligned}\chi_k(t) = &(2Q_k^l(t) + Z_k(t))A_k(t) + (Q_k^l(t))^2 + (Q_k^m(t))^2\\ &+ Z_k(t)\min(N_{k,\max}^u(t), Q_k^l(t))\\ &- Z_k(t)Q_k^{\text{avg}} - \nu_k Y_k(t)\mathcal{G}_k^{\text{avg}}.\end{aligned} \quad (19)$$

Note that, when plugged into (17), because of the conditioning, the expected value containing $\chi_k(t)$ is a function that does not depend on the optimization variables. Now, hinging on stochastic optimization, we minimize the instantaneous realizations of (17), thus removing the expectation term. Then, neglecting all the constant terms, the per-slot optimization problem can be written as:

$$\begin{aligned}\min_{\mathbf{\Phi}(t)}\sum_{k=1}^{K}\Big[&2(Q_k^m(t) - Q_k^l(t))N_k^u(t) - 2Q_k^m(t)N_k^c(t)\\ &+ Z_k(t)\Big(\max\left(0, Q_k^l(t) - N_k^u(t)\right)\\ &+ \max(0, Q_k^m(t) - N_k^c(t))\Big)\\ &+ \nu_k Y_k(t)G_k(t)\Big] + VE_{\text{tot}}(t)\end{aligned}$$

subject to

$$\mathbf{\Phi}(t) \in \mathcal{Z}(t) \quad (20)$$

where $\mathcal{Z}(t)$ is the feasible set according to constraints (*c*)-(*g*) of (12). Problem (20) is still complicated to solve, due to its mixed integer nonlinear nature. However, the problem can be split in two simpler sub-problems: the first one selecting the data rate and number of quantization bits; the second one optimizing the edge server CPU scheduling. Then, as we will illustrate in the sequel, each subproblem can be solved using a low-complexity procedure.

[4] Note that the step size does not change the problem, since it comes from the multiplication of both sides of (*b*) in (12) by a scalar $\nu_k > 0$.

B. OPTIMAL DATA RATE AND QUANTIZATION BITS

We now proceed at solving (20) from the radio resource allocation perspective. To this aim, we first introduce an additional upper bound, used to handle the expression of $N_k^u(t)$ (cf. (5)) and simplify the structure of the sub-problem. In particular, using the fact that $x-1 \leq \lfloor x \rfloor \leq x$, we can remove the non-linearity introduced by the $\lfloor \cdot \rfloor$ operator. Moreover, without loss of generality, we transform constraint (c) of (12) as:

$$0 \leq R_k(t) \leq R'_{k,\max}(t), \quad \forall k, t,$$

with

$$R'_{k,\max}(t) = \min\left(R_{k,\max}(t), \frac{(Q_k^l(t)+1)M_k n_k^q(t)}{\tau}\right),$$

thus eliminating the non-linearity introduced by the max$(\cdot)$ operator. It should be also noticed that this constraint does not alter the problem, since it ensures that each device does not transmit more data than those available in the uplink queue at time t. Now, the optimization problem in (20) with respect to data rate and number of quantization bits can be split into simpler problems for each device. In particular, for each device k, the radio resource allocation problem can be formulated as follows (we omit the temporal index t for ease of notation):

$$\min_{R_k, n_k^q} \; -\frac{\widetilde{Q}_k^l \tau R_k}{M_k n_k^q} + \frac{\tau V N_0 B_k}{h_k} \exp\left(\frac{R_k \ln(2)}{B_k}\right) + \nu_k Y_k G_k$$

subject to

$$\begin{aligned} &a) \quad 0 \leq R_k \leq R'_{k,\max} \\ &b) \quad n_k^q \in \mathcal{N}_k^q, \end{aligned} \tag{21}$$

where $\widetilde{Q}_k^l = 2(Q_k^l - Q_k^m) + Z_k$. Problem (21) is a mixed integer problem and thus in principle it is difficult to solve. However, since $\mathcal{N}_k^q$ is a discrete finite set with typically low cardinality, it is possible to solve it exactly using an exhaustive search over the variable n_k^q. Since we have been able to split the optimization per device, there is no exponential increase of complexity with the number of connected devices. In particular, for any given $n_k^q \in \mathcal{N}_k^q$, if $\widetilde{Q}_k^l \leq 0$, the optimal solution is $R_k = 0$, since the first two terms of the objective function of (21) are monotone increasing functions of R_k, while the third one does not depend on R_k. Instead, if $\widetilde{Q}_k^l > 0$, given n_k^q, (21) is a convex optimization problem with respect to R_k, and its global optimal solution can be derived in closed-form. Indeed, the Lagrangian associated to problem (21) reads as:

$$\begin{aligned} \mathcal{L} = &-\frac{\widetilde{Q}_k^l \tau R_k}{M_k n_k^q} + \frac{\tau V N_0 B_k}{h_k} \exp\left(\frac{R_k \ln(2)}{B_k}\right) \\ &+ \nu_k Y_k G_k(n_k^q) - \alpha_k R_k + \beta_k \left(R_k - R'_{k,\max}\right) \end{aligned}$$

where α_k and β_k are the Lagrange multipliers associated to constraint (a) of (21). Since we are looking the solution corresponding to a given value of n_k^q, of course there is no need to introduce any multiplier for n_k^q. Then, in this case the Karush-Kuhn-Tucker (KKT) conditions [42] of (12) are:

$$\begin{aligned} &i) \; \frac{\partial \mathcal{L}}{\partial R_k} = -\frac{\tau \widetilde{Q}_k^l}{M_k n_k^q} + \frac{\tau V \ln(2) N_0}{h_k} \exp\left(\frac{R_k \ln(2)}{B_k}\right) \\ &\qquad - \alpha_k + \beta_k = 0; \\ &ii) \; R_k \geq 0, \quad \alpha_k \geq 0, \quad \alpha_k R_k = 0; \\ &iii) \; R_k \leq R'_{k,\max}, \quad \beta_k \geq 0, \quad \beta_k \left(R_k - R'_{k,\max}\right) = 0. \end{aligned}$$

Solving the KKT conditions, it is easy to see that the global optimal solution of (21), for a given n_k^q, is

$$R_k^* = \begin{cases} \left[\frac{B_k}{\ln(2)} \ln\left(\frac{\widetilde{Q}_k^l h_k}{M_k n_k^q V \ln(2) N_0}\right)\right]_0^{R'_{k,\max}}, & \text{if } \widetilde{Q}_k^l > 0; \\ 0, \quad \text{otherwise.} \end{cases} \tag{22}$$

Finally, to find the global optimal solution of (21), it is only necessary to compute (22) for all $n_k^q \in \mathcal{N}_k^q$ and choose the solution that yields the smallest value of the objective function of (21). The procedure is summarized in Algorithm 1, and it requires $K \times |\mathcal{N}_k^q|$ iterations (which can be directly parallelized if a multi-core architecture is available). Interestingly, the fact that $\mathcal{N}_k^q$ is a finite discrete set, usually with a few elements, allows us to achieve the global optimal solution of (21) within a few iterations, independently of the structure of $G_k(n_k^q)$, which can be non convex and/or non differentiable.

Algorithm 1: Radio Resource Allocation

```
In each time slot t, observe Q_k^l, Q_k^m, Z_k, Y_k, h_k, ∀k.
Define an |N_k^q| × 1 vector N_k^q for each device, with
  entries N_ki^q equal to the elements of N_k^q.
Define the K × |N_k^q| matrices {P_ki}_{k,i} and {ρ_ki}_{k,i}, with
  P_ki = ρ_ki = 0, ∀k, i.
for k = 1 : K do
    S1. Compute Q̃_k^l = 2(Q_k^l − Q_k^m) + Z_k;
    if Q̃_k^l ≤ 0 then
        S2. R_k^opt = 0;
    else
        for n = 1 : |N_k^q| do
            S3. Compute R_k^* as in (22) and save it in ρ_ki;
            S4. Compute the value of the objective
            function of (21) with n_k^q = N_ki^q and R_k = ρ_ki,
            and save it in P_ki;
        end
        S5. Set i* = arg min_i {P_ki}_i
        S6. Set R_k^opt = ρ_ki*, n_k^{q,opt} = N_ki*^q
    end
end
```

C. CPU SCHEDULING AT THE SERVER

We now proceed to the solution of (20) from the computation perspective, optimizing the CPU scheduling at the ES.

Proceeding in a similar way to the previous case, we exploit the inequality $x - 1 \leq \lfloor x \rfloor \leq x$ in (20), and we impose the following additional constraint:

$$f_k(t) \leq \min\left(f_c(t), \frac{Q_k^m(t)+1}{\tau J_k}\right),$$

which ensures that we cannot allocate more computation resources than those required to completely drain the remote queue $Q_k^m(t)$. In this way, the optimization problem for the computation resource allocation can be cast as (we omit the temporal index t for ease of notation):

$$\begin{aligned} \min_{\{f_k\}_k, f_c} \quad & -\sum_{k=1}^{K} \widetilde{Q}_k^m \tau f_k J_k + V\tau\kappa f_c^3 \\ \text{subject to} \quad & \\ & (a) \quad f_c \in \mathcal{F}; \\ & (b) \quad 0 \leq f_k \leq \min\left(f_c, \frac{Q_k^m+1}{\tau J_k}\right), \quad \forall k \\ & (c) \quad \sum_{k=1}^{K} f_k \leq f_c \end{aligned} \tag{23}$$

where $\widetilde{Q}_k^m = 2Q_k^m + Z_k$. Problem (23) is a mixed-integer program that, similarly as before, admits a simple solution. In fact, if the discrete variable f_c is fixed, (23) becomes a linear programming problem and its solution can be found via a simple fast iterative procedure, described by the steps S2-S5 of Algorithm 2. Thus, repeating these steps for all possible $f_c \in \mathcal{F}$, we can find by comparison the global optimal solution of (23), through the procedure summarized in Algorithm 2. Note that, denoting by $|\mathcal{F}|$ the cardinality of $\mathcal{F}$, Algorithm 2 requires at most $|\mathcal{F}| \times K$ iterations. Again, the complexity increases linearly with the number of connected devices.

Finally, the overall procedure for the proposed dynamic resource allocation strategy for minimum energy edge machine learning is summarized in Algorithm 3.

D. DATA-DRIVEN CONTROL OF LEARNING ACCURACY

In the previous section, we have proposed a model-based online algorithm for radio and computation resource orchestration, assuming the possibility of writing a closed form expression for the learning accuracy $G_k(t)$. Indeed, as we will show in Sec. V.A, some learning tasks admit closed form expressions for different accuracy metrics, thus allowing us to use Algorithm 3. However, in several other cases, it is not possible to provide a closed form expression for the performance metrics. In these cases, we need to propose an alternative strategy. To this aim, we now propose an alternative approach that is valid in the case in which the accuracy can be estimated online. As an example, if we consider a prediction task, once a sensor observes a new sample, it can compare it with its prediction from the previous samples and then measure the prediction error. The prediction can be fed back to the sensor by the edge server. Once the prediction accuracy is estimated, it is possible to actuate a control action accordingly, in order

Algorithm 2: Edge Server CPU Scheduling

In each time slot t, observe Q_k^m, Z_k, $\forall k$.
Define the $|\mathcal{F}| \times 1$ vector of the available CPU frequencies $\varphi = [0, \ldots, f_{\max}]^T$. Define the $|\mathcal{F}| \times K$ matrix $F = \{F_{ik}\}_{i,k}$, and the $|\mathcal{F}| \times 1$ vector $L = \{L_i\}_i$. Set $F_{ik} = 0\ \forall i, k$, and $L_i = 0\ \forall i$.
for $i = 1, \ldots, |\mathcal{F}|$ **do**
  S1. Let $\tilde{\varphi} = \varphi_i$ and $\mathcal{U} = \{k = 1, \ldots, K\}$.
  while $\tilde{\varphi} > 0$ **do**
    S2. $\tilde{k} = \arg\max_{k \in \mathcal{U}} \left\{J_k(2Q_k^m + Z_k)\right\}$.
    S3. $F_{i\tilde{k}} = \min\left(\frac{Q_{\tilde{k}}^r+1}{\tau J_{\tilde{k}}},\ \tilde{\varphi}\right)$.
    S4. $\mathcal{U} = \mathcal{U} \setminus \left\{\tilde{k}\right\}$.
    S5. $\tilde{\varphi} = \tilde{\varphi} - F_{i\tilde{k}}$.
    if $\mathcal{U} = \emptyset$ **then**
      break.
    end
  end
  S6. Compute the value of the objective function in (23) with $f_c = \varphi_i$ and $f_k = F_{ik}$, $\forall k$, and save it in L_i.
end
S7. Find $i^* = \arg\min_i\{L_i\}$, and then set

$$f_c^{\text{opt}} = \varphi_{i^*}, \quad f_k^{\text{opt}} = F_{i^*k}\ \forall k.$$

Algorithm 3: Model-Based Edge Machine Learning

Set the Lyapunov trade-off parameter V, $Z_k(0)$, $Y_k(0)$, ν_k, for all k. In each time slot t, repeat the following steps:
S1. Find the transmit data rate R_k and the number of quantization bits n_k^q, $\forall k$, using Algorithm 1;
S2. Solve the CPU scheduling through Algorithm 2;
S3. Run the online learning task;
S4. Update the physical queues as in (6) and (8), and the virtual queues as in (13) and (14).

to achieve the target performance within a given delay. More specifically, indicating with $y_k(t)$ the sample observed by device k at time t, and with $\hat{y}_k^w(t)$ its prediction based on the previous samples, using the data set W_k, the online learning accuracy can be estimated as follows:

$$\widehat{G}_k(t) = \frac{1}{|W_k|} \sum_{w \in W_k} (\hat{y}_k^w(t) - y_k(t))^2. \tag{24}$$

Using (24), the evolution of the virtual queue $Y_k(t)$ can be written as (14), replacing the closed form expression $G_k(t)$ with $\widehat{G}_k(t)$. Using $\widehat{G}_k(t)$ is useful for the virtual queue's update, but it is not directly related to the number of quantization bits, which affect the learning accuracy. Thus, the control action might still not be easily implementable, due to the lack of a closed form expression for G_k and $\widehat{G}_k(t)$. One possible

solution to this issue builds on the following assumption, which is largely verified both from a theoretical and a numerical point of view (practical examples follow in Sec. V).

Assumption 1: G_k is a monotone non-increasing function of the number of quantization bits n_k^q.

Assumption 1 hinges on the fact that a finer representation of the data generally leads to better performance of the learning task in terms of accuracy. Then, under Assumption 1, we propose to exploit a surrogate function for G_k in (20), say $\widetilde{G}_k(n_k^q)$, which approximates the non-increasing behavior of G_k. The rationale underlying this choice comes from the concept of Γ-additive approximation [41, p. 59] of the drift-plus-penalty method in (20), which makes possible to use inexact updates of the algorithm at each iteration, provided that the approximation error can be bounded within a finite error Γ. Of course, due to the boundedness of $G_k(n_k^q)$ over the finite discrete set $\mathcal{N}_k^q$, any bounded non-increasing discrete function of the number of bits leads to a valid Γ-approximation of the drift-plus-penalty method in (20). As an example, we can assume that the accuracy is inversely proportional to the distortion d induced by the quantization. More specifically, denoting by $d_k(t)$ the amount of distortion tolerated in time slot t for user k, to be adjusted online depending on the learner accuracy, the number of bits associated to $d_k(t)$ can be derived from fundamental rate-distortion theory limits [43]. Let us recall that, for a Gaussian random variable X, with zero mean and variance σ^2, the minimum number of bits necessary to quantize X providing a distortion at most equal to $d_k(t)$ is [43]

$$r(d_k(t)) = \frac{1}{2}\max\left(0, \log_2 \frac{\sigma^2}{d_k(t)}\right) \text{ bits.} \tag{25}$$

In the sequel, we will use a number of bits per sample $n_k^q(t) = \alpha\, r[d_k(t)]$, where $\alpha > 1$ is a margin coefficient introduced to take into account the fact that in practice the data may not follow a Gaussian distribution.

Inverting (25), we can choose $\widetilde{G}_k(n_k^q)$=c $2^{-2\,n_k^q}$, where c is a suitable coefficient. However, more general designs for $\widetilde{G}_k(n_k^q)$ can be exploited if some information on the shape of G_k is known in advance or inferred from data. Then, at a given time slot t, substituting G_k with $\widetilde{G}_k(n_k^q)$ in (21), we solve the following deterministic sub-problem for the data rate and quantization bits selection:

$$\begin{aligned}\min_{R_k, n_k^q} \quad & -\frac{\widetilde{Q}_k^l \tau R_k}{M_k n_k^q} + \frac{\tau V N_0 B_k}{h_k}\exp\left(\frac{R_k \ln(2)}{B_k}\right) \\ & + \nu_k Y_k \widetilde{G}_k(n_k^q) \\ \text{subject to} \quad & a) \quad 0 \le R_k \le R'_{k,\max} \\ & b) \quad n_k^q \in \mathcal{N}_k^q,\end{aligned} \tag{26}$$

where the virtual queues $\{Y_k(t)\}_{k=1}^K$ in (14) are updated using the online accuracy estimate $\widehat{G}_k(t)$ given by (24). The sub-problem in (26) can then be solved as in the previous case. The main steps of the proposed data-driven approach are summarized in algorithm 4. The performance of this data-driven strategy will be numerically assessed in Sec. V.

Algorithm 4: Data-Driven Edge Machine Learning

Set the Lyapunov trade-off parameter V, $Z_k(0)$, $Y_k(0)$, ν_k, $\forall k$. In each time slot t, repeat the following steps:
S1. Find the transmit data rate R_k and the number of quantization bits solving (26) with Algorithm 1;
S2. Solve the CPU scheduling through Algorithm 2;
S3. Run the online learning task;
S4. Update the physical queues as in (6) and (8), and the virtual queues as in (13) and (14), using (24) for the latter.

IV. BEST ACCURACY UNDER ENERGY AND E2E DELAY CONSTRAINTS

In this section, we present an alternative formulation of EML, useful in cases where there is no a priori specification about the accuracy of the learner, but the goal is rather to optimize the learning accuracy (without any particular assumption on the model and the final performance), subject to energy and latency constraints. This alternative formulation hinges on Assumption 1. In particular, if we aim to optimize the accuracy $G_k(n_k^q)$, exploiting the assumption that $G_k(n_k^q)$ is a monotone non-increasing function of the number of quantization bits, we can equivalently formulate the problem as the maximization of the number of quantization bits, subject to latency and energy constraints. Thus, using the notation introduced in the previous section, we formulate the problem as follows:

$$\min_{\mathbf{\Phi}(t)} \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{-\sum_{k=1}^{K} n_k^q(t)\right\}$$

subject to

$$\begin{aligned}
&(a) \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{Q_k^{\text{tot}}(t)\right\} \le Q_k^{\text{avg}}, \quad \forall k \\
&(b) \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{e_k(t)\right\} \le e_k^{\text{avg}}, \quad \forall k; \\
&(c) \quad \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left\{E_s(t)\right\} \le E_s^{\text{avg}}, \quad \forall k; \\
&(d) \quad 0 \le R_k(t) \le R_{k,\max}(t), \quad \forall k, t; \\
&(e) \quad n_k^q(t) \in \mathcal{N}_k^q, \quad \forall k, t; \\
&(f) \quad f_c(t) \in \mathcal{F}, \quad \forall t; \\
&(g) \quad f_k(t) \ge 0, \quad \forall k, t; \\
&(h) \quad \sum_{k=1}^{K} f_k(t) \le f_c(t), \quad \forall t;
\end{aligned} \tag{27}$$

where $\mathbf{\Phi}(t) = \left[\{R_k(t)\}_k, \{f_k(t)\}_k, \{f_c(t)\}, \{n_k^q(t)\}_k\right]$. Differently from (12), we now introduce the constraints (*b*) and (*c*) on the average device and ES energy consumption, respectively. The expectation is still taken with respect to the random wireless channels and data arrivals. Then, to tackle this long-term optimization problem, as before, we introduce a virtual queue for each long-term constraint, and we devise a strategy satisfying the desired constraints by enforcing the mean rate stability of the virtual queues. In particular, the virtual queue for the E2E delay constraint is the same as in the previous section (cf. (13)). For constraint (*b*) of (27), we introduce a virtual queue $S_k(t)$ evolving as follows:

$$S_k(t+1) = \max\left(0, S_k(t) + \lambda_k \left(e_k(t) - e_k^{\text{avg}}\right)\right), \quad (28)$$

where λ_k is a step size. Similarly, for constraint (*c*) of (27), we use a virtual queue $O(t)$ evolving as:

$$O(t+1) = \max\left(0, O(t) + E_s(t) - E_s^{\text{avg}}\right). \quad (29)$$

To stabilize the virtual queues while taking the objective of (27) into account, we first introduce the Lyapunov function

$$L(\mathbf{\Theta}_2(t)) = \frac{1}{2}\sum_{k=1}^{K}\left[Z_k^2(t) + S_k^2(t)\right] + \frac{1}{2}O^2(t),$$

with $\mathbf{\Theta}_2(t) = [\mathbf{Z}(t), \mathbf{S}(t), O(t)]$ and then its associated drift-plus-penalty function (cf. (16)), which in this case reads as

$$\Delta_p(\mathbf{\Theta}_2(t)) = \Delta(\mathbf{\Theta}_2(t)) - V \cdot \mathbb{E}\left\{n_k^q(t)\middle|\mathbf{\Theta}_2(t)\right\}. \quad (30)$$

In particular, it is easy to show (the details are given in the Appendix) that the drift-plus-penalty function enjoys the following upper-bound:

$$\begin{aligned}\Delta_p(\mathbf{\Theta}_2(t)) \le \zeta_2 + \mathbb{E}\Bigg\{\sum_{k=1}^{K}\Big[&\chi_{k,2}(t) - 2Q_k^l(t)N_k^u(t) \\ &+ 2Q_k^m(t)\left(N_k^u(t) - N_k^c(t)\right) \\ &+ Z_k(t)\Big(\max\left(0, Q_k^l(t) - N_k^u(t)\right) \\ &+ \max(0, Q_k^m(t) - N_k^c(t))\Big) \\ &+ \lambda_k S_k(t)e_k(t) - Vn_k^q(t)\Big] \\ &+ O(t)E_s(t)\Bigg|\mathbf{\Theta}_2(t)\Bigg\}, \end{aligned} \quad (31)$$

where ζ_2 is a positive constant given by

$$\begin{aligned}\zeta_2 = \frac{1}{2}\sum_{k=1}^{K}\Big[&2\left(A_k^{\max}\right)^2 + 4\left(N_{k,\max}^u\right)^2 + 2(N_{k,\max}^c)^2 \\ &+ (Q_k^{\text{avg}})^2 + \lambda_k^2\left(e_k^{\max} - e_k^{\text{avg}}\right)^2\Big] \\ &+ \frac{\left(E_s^{\max} - E_s^{\text{avg}}\right)^2}{2}, \end{aligned} \quad (32)$$

and $\chi_{k,2}(t)$ is defined as

$$\begin{aligned}\chi_{k,2}(t) = &(2Q_k^l(t) + Z_k(t))A_k(t) + \left(Q_k^l(t)\right)^2 + \left(Q_k^m(t)\right)^2 \\ &+ Z_k(t)\min(N_{k,\max}^u(t), Q_k^l(t)) - Z_k(t)Q_k^{\text{avg}} \\ &- \lambda_k S_k(t)e_k^{\text{avg}} - O(t)E_s^{\text{avg}}. \end{aligned} \quad (33)$$

This function is to be considered as a constant with respect to the optimization problem, because it does not depend on the optimization variables.

Now, following similar arguments as in the previous section, the problem can be split into two sub-problems. In particular, it can be easily shown that the final per slot problem used to find the data rate and the number of quantization bits for a generic device k is given by:

$$\begin{aligned}\min_{R_k, n_k^q} \quad &-\frac{\widetilde{Q}_k^l \tau R_k}{M_k n_k^q} + \frac{\lambda_k S_k \tau N_0 B_k}{h_k}\exp\left(\frac{R_k \ln(2)}{B_k}\right) - Vn_k^q \\ \text{subject to} \quad &a) \quad 0 \le R_k \le R'_{k,\max} \\ &b) \quad n_k^q \in \mathcal{N}_k^q. \end{aligned} \quad (34)$$

Again, if n_k^q is fixed, (34) is convex and differentiable, and its solution can be derived in closed form. In particular, solving the KKT conditions for a given n_k^q, the global optimal solution of (34) is given by:

$$R_k^* = \begin{cases}\left[\dfrac{B_k}{\ln(2)}\ln\left(\dfrac{\widetilde{Q}_k^l h_k}{M_k n_k^q \lambda_k S_k \ln(2) N_0}\right)\right]_0^{R'_{k,\max}} & \text{if } \widetilde{Q}_k^l > 0; \\ 0, \quad \text{otherwise.}\end{cases} \quad (35)$$

Thus, using (35) for each value of $n_k^q \in \mathcal{N}_k^q$, we find the global optimal solution by selecting the pair (R_k^*, n_k^q) that minimizes the objective function of (34), following the steps illustrated in Algorithm 5.

Similarly, the sub-problem for the optimal CPU clock frequency and scheduling is given by:

$$\begin{aligned}\min_{\{f_k\}_k, f_c} \quad &-\sum_{k=1}^{K}\widetilde{Q}_k^m \tau f_k J_k + O\,\tau\,\kappa f_c^3 \\ \text{subject to} \quad &(a) \quad f_c \in \mathcal{F}; \\ &(b) \quad 0 \le f_k \le \min\left(f_c, \frac{Q_k^m + 1}{\tau J_k}\right), \quad \forall k; \\ &(c) \quad \sum_{k=1}^{K} f_k \le f_c. \end{aligned} \quad (36)$$

Again, once f_c is fixed, (36) is a linear programming problem, which can be solved using Algorithm 2, where in step S6, the value of the objective function of (23) is substituted with the value of the objective function of (36).

To summarize, Algorithm 6 describes the overall EML dynamic resource allocation procedure that optimizes the learning accuracy under latency and energy constraints.

Algorithm 5: Radio Resource Allocation (Best Accuracy)

In each time slot t, observe $Q_k^l, Q_k^m, Z_k, S_k, h_k, \forall k$.
Define an $|\mathcal{N}_k^q| \times 1$ vector N_k^q for each device, with entries $N_{k,i}^q$ equal to the elements of $\mathcal{N}_k^q$.
Define the $K \times |\mathcal{N}_k^q|$ matrices $\{P_{ki}\}_{k,i}$ and $\{\rho_{ki}\}_{k,i}$, with $P_{ki} = \rho_{ki} = 0, \forall k, i$.
for $k = 1 : K$ **do**
  S1. Compute $\widetilde{Q}_k^l = 2(Q_k^l - Q_k^m) + Z_k$;
  if $\widetilde{Q}_k^l \leq 0$ **then**
    S2. $R_k^{\text{opt}} = 0$;
  else
    for $n = 1 : |\mathcal{N}_k^q|$ **do**
      S3. Compute R_k^* as in (35) and save it in ρ_{ki};
      S4. Compute the value of the objective function of (34) with $n_k^q = N_{ki}^q$ and $R_k = \rho_{ki}$, and save it in P_{ki};
    end
    S5. Set $i^* = \arg\min_i \{P_{ki}\}_i$
    S6. Set $R_k^{\text{opt}} = \rho_{ki^*}$, $n_k^{q,\text{opt}} = N_{ki^*}^q$
  end
end

V. APPLICATIONS AND NUMERICAL RESULTS

In this section, we illustrate some applications of our EML framework to specific learning problems, and then present numerical simulations to assess the performance of the proposed resource allocations strategies. In particular, in Sec. V-A, we customize our adaptive learning strategy to least mean squares estimation as a particular case, thus illustrating numerical results in Sec. V-A for the model-based and the data-driven solutions presented in Sec. III, considering both synthetic and real datasets. Finally, in Secs. V-B and V-C, we customize our framework to SVM and NN classification, illustrating numerical results using the best accuracy strategy presented in Sec. IV.

A. LEAST MEAN SQUARES ADAPTIVE ESTIMATION

Let us briefly recall the basic concepts of the LMS adaptive algorithm that we consider in this paper. Given a streaming sequence of $U_k \times 1$ input data vectors $\mathbf{u}_k^{(n)}$ and a parameter vector $\mathbf{w}_{k,0}$ (to be learnt), we assume the following linear input/output relation:

$$y_k^{(n)} = \mathbf{u}_k^{(n)T} \mathbf{w}_{k,0} + v_k^{(n)}, \tag{37}$$

where the output $y_k^{(n)}$ is a random streaming sequence of output data, $v_k^{(n)}$ is a realization of random observation noise with variance $\sigma_{k,v}^2$, and the superscript ^{T} denotes vector transposition. A typical approach to find the best estimate of the parameter vector $\mathbf{w}_{k,0}$ from the stream of input data $\mathbf{u}_k$ and noisy observations y_k consists in minimizing the Mean

Algorithm 6: Best Accuracy Edge Machine Learning

Set the Lyapunov trade-off parameter V, $Z_k(0)$, $S_k(0)$, λ_k, for all k, and $O(0)$. In each slot t, repeat the following steps:
S1. Find the transmit data rate R_k and the number of quantization bits n_k^q, $\forall k$, through Algorithm 5;
S2. Solve the CPU scheduling through Algorithm 2, with step S6 modified with the objective function of (36);
S3. Update the physical queues as in (6) and (8), and the virtual queues as in (13) and (28), and (29).

Squared Error given by:

$$\text{MSE}_k(\mathbf{w}_k) = \mathbb{E}\left\{\left(y_k^{(n)} - \mathbf{u}_k^{(n)T}\mathbf{w}_k\right)^2\right\}. \tag{38}$$

If the statistics of the data are known in advance, the MSE cost function can be optimized via the traditional gradient descent algorithm. However, if the statistics are unknown, one can follow the LMS approach that drops the expectation and uses an instantaneous approximated version of the gradient, thus obtaining the stochastic gradient descent recursion given by [44]:

$$\mathbf{w}_k^{(n)} = \mathbf{w}_k^{(n-1)} + \mu_k \mathbf{u}_k^{(n)}\left(y_k^{(n)} - \mathbf{u}_k^{(n)T}\mathbf{w}_k^{(n-1)}\right), \tag{39}$$

where $\mu_k > 0$ is a sufficiently small step-size. In our case, the data $\mathbf{u}_k^{(n)}$ and the observations $y_k^{(n)}$ are first quantized, using a finite number of bits, and then sent to the edge server for processing. The quantization introduces an additional noise to the data, which determines a biased estimate of the LMS algorithm [45]. To avoid the detrimental effect of the bias, we can use the following bias-compensated recursion [45]:

$$\begin{aligned}\mathbf{w}_k^{(n)} &= (I_{U_k} + \Sigma_{k,q})\mathbf{w}_k^{(n-1)} \\ &\quad + \mu_k^{(n)}\mathbf{u}_k^{(n)}\left(y_k^{(n)} - \mathbf{u}_k^{(n)T}\mathbf{w}_k^{(n-1)}\right)\end{aligned} \tag{40}$$

where $\Sigma_{k,q}$ is the covariance matrix of the noise affecting the input data (related to quantization effects), which is assumed to be diagonal, and I_{U_k} is the $U_k \times U_k$ identity matrix. In the sequel we will provide a closed form expression for $\Sigma_{k,q}$, which depends on the adopted quantization scheme. The stochastic recursion (40) will then be used in step S3 of Algorithm 3. As accuracy metric G_k, we choose the Mean Squared Deviation between the estimated parameter vector $\mathbf{w}_k^{(n)}$, and the true parameter vector $\mathbf{w}_{k,0}$, defined as:

$$\text{MSD}_k = \mathbb{E}\left\{\left|\left|\mathbf{w}_k^{(n)} - \mathbf{w}_{k,0}\right|\right|^2\right\}.$$

Interestingly, in the case of noisy data, denoting by $C_{k,u}$ the covariance matrix of the input data, and by $\sigma_{k,q,o}^2$ the variance of the noise affecting the output data, and defining

$$\begin{aligned}\gamma_k &= \mu_k^2 vec\big((\Sigma_{k,q} + C_{k,u})\sigma_{k,q,o}^2 \\ &\quad + \mathbf{w}_{k,0}^T \Sigma_{k,q}\mathbf{w}_{k,0}(\Sigma_{k,q} + C_{k,u}) \\ &\quad + \Sigma_{k,q}\mathbf{w}_{k,0}\mathbf{w}_{k,0}^T\Sigma_{k,q}\big),\end{aligned} \tag{41}$$

the MSD admits the following closed form expression [45]:

$$\mathrm{MSD}_k = \gamma_k^T \left(\mu_k I_{U_k} \otimes C_{k,u} + \mu_k C_{k,u}^T \otimes I_{U_k}\right)^{-1} vec(I_{U_k}), \tag{42}$$

where $\otimes$ denotes the Kronecker product and $vec(A)$ is the vectorization of matrix A. Then, as accuracy function $G_k(n_k^q(t))$ in (11)-(12), we use the values of MSD_k given by (42). Furthermore, since $\mathbf{w}_{k,0}$ in (41) is obviously unknown *a priori*, we replace it with the online LMS estimate $\mathbf{w}_k^{(n)}$ in (40), which is asymptotically (at convergence) unbiased and has a small steady-state error [45]. In this way, as time goes on, the true parameter $\mathbf{w}_{0,k}$ can be suitably replaced by $\mathbf{w}_k^{(n)}$ in (41).

NUMERICAL RESULTS: MINIMUM ENERGY STRATEGIES FOR LMS

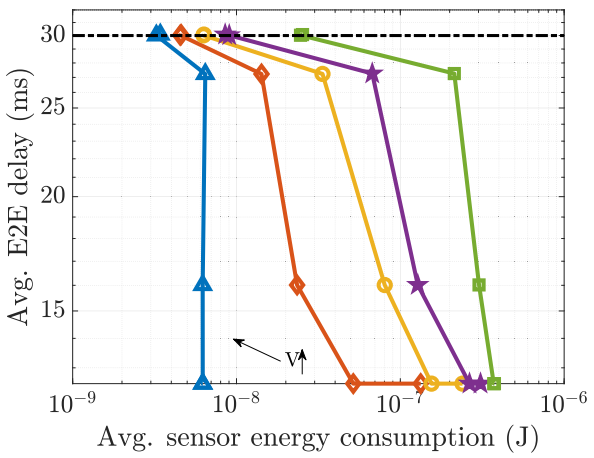
In this section, we present some numerical results obtained with computer simulations, for the resource allocation strategy devised in Sec. III. We present simulations for two different approaches: model-based and data-driven, which will be explained later on in this section. For these simulations, we use the following settings.

Scenario: We consider a single AP at the center of a squared area of side 200 m, with a carrier frequency $f_0 = 3.5$ GHz. The propagation model is the "Alpha-Beta-Gamma" model presented in [46], and we assume a Rayleigh fading with unit variance. We also assume a total available bandwidth $B = 180$ kHz, equally shared among $K = 5$ end devices. The noise spectral density is $N_0 = -174$dBm/Hz, with a noise figure $F = 5$ dB at the receiver. The slot duration is fixed to $\tau = 5$ ms, and the maximum transmit power of the end devices is $p_k^{\max} = 100$ mW, $\forall k$, used to compute the maximum achievable data rate $R_k^{\max}(t)$ in each time slot from the Shannon formula. We assume an edge server equipped with a CPU clock frequency to be chosen in $\varphi = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1] \times 3.3$ GHz. All devices are randomly distributed in a squared area around the AP. The conversion factor J_k (cf. (8)) has been estimated offline. In particular, we have run LMS in Matlab® R2019b on a Linux CentOS 7 workstation equipped with an Intel® Core™ i9-9940X CPU @ 3.30GHz. The CPU speed (or 'clock') measures the number of cycles per second: multiplying the CPU speed (in Hz) with the time (in seconds) required to accomplish a given calculation, we get the number of CPU cycles needed to accomplish the calculation. However, it is worth noting that this is a rough estimate because the CPU may be running, at the same time, some background task. To mitigate the effect of background tasks, we averaged the number of cycles per pattern across several trials. The output estimate is $J = 2 \times 10^{-4}$ data units/CPU cycle. This is the value we use for the following simulations.

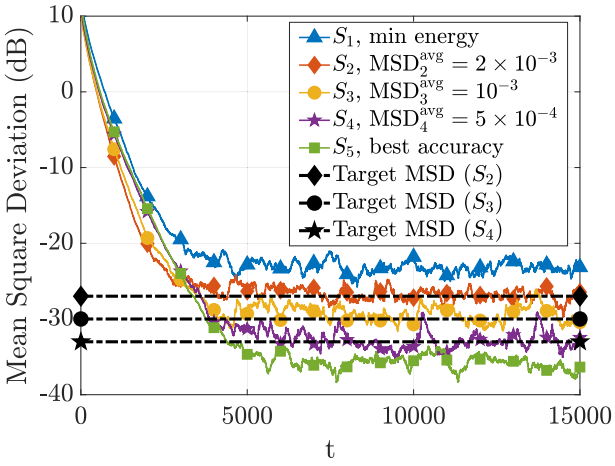
Model-based minimum energy LMS estimation. In this paragraph, we aim at assessing the performance of the model-based Algorithm 3, when applied to adaptive LMS estimation. For the first simulation, we use a synthetic dataset, obtained by generating regression parameters from a Gaussian distribution with zero mean and variance $\sigma_{k,u}^2 = 0.1$, with data unit dimensionality $U_k = 10$. Thus, the parameters to be transmitted to the edge server have dimensionality $M_k = U_k + 1$, since both the observation $y_k^{(n)}$ and the regressors $\mathbf{u}_k^{(n)}$ must be transmitted to run recursion (40). The observations $y_k^{(n)}$ (cf. (37)) are corrupted by Gaussian noise with zero mean and variance $\sigma_{k,v}^2 = 10^{-2}$. The true parameter vector $\mathbf{w}_{k,0}$ is the realization of a uniform random vector variable, whose elements are in [0, 0.5]. The step size used for the LMS recursion is $\mu_k = 5 \times 10^{-3}$ (cf. (40)). The set of the possible number of quantization bits is $\mathcal{N}_k^q = \{3, 4, 5, 6, 7, 8\}$. The input data $\mathbf{u}_k^{(n)}$ are quantized with a uniform dithered quantization [47], with dithering uniformly randomly distributed in $\left[-\frac{\Delta_k}{2}, \frac{\Delta_k}{2}\right]$, where $\Delta_k = \frac{2l_k}{2^{n_k^q}-1}$; here l_k represents the amplitude dynamic of the signal. Using this kind of dithered quantization, the noise affecting the data has covariance $\Sigma_{k,q} = \frac{\Delta_k^2}{6} I_{U_k}$ (cf. (41)).

Energy-delay-accuracy trade-off: In Fig. 3, we illustrate the performance of Algorithm 3, in terms of trade-off between energy, delay, and learning accuracy. In particular, the five curves refer to 5 different strategies: i) Strategy S_1 (blue curve ▲) is the minimum energy consumption strategy, obtained using always the minimum number of bits/sample, e.g., $n_k^q(t) = n_{k,\min}^q = 3$, for all t; ii) Strategy S_5 (green curve ■) is the best accuracy strategy, where the sensor always uses the maximum number of bits/samples, e.g., $n_k^q(t) = n_{k,\max}^q = 8$, for all t; iii) Strategies S_2, S_3, and S_4 (orange ♦, yellow • and purple ★ curves) represent the intermediate strategies, corresponding to different constraints on the value of the MSD, where the number of bits is adapted over time, depending on the values of the real and virtual queues. From these curves, it is interesting to highlight how the accuracy affects the performance of the overall system in terms of energy-delay trade-off.

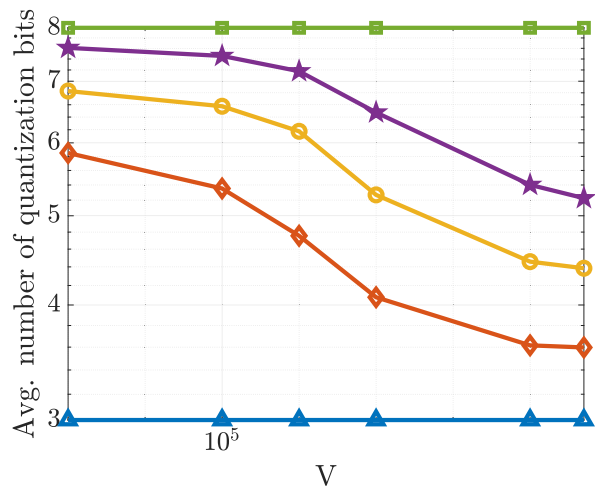
In particular, Fig. 3a shows the average E2E delay as a function of the average sensor energy consumption. The curves have been obtained by changing the parameter V in (16), in order to explore the energy-delay trade-off. More specifically, V increases from right to left, as shown in the figure. From Fig. 3a, we can notice how, by increasing V, the energy consumption decreases while the E2E delay increases, as expected. From Fig. 3c, we can observe how the average number of bits/sample varies as a function of the parameter V used to explore the energy-delay trade-off: while the strategies S_1 and S_5 keep that number constant, all the intermediate strategies adapt the number of bits assigning less bits to spend less energy, while respecting the long term accuracy constraint. Clearly, the number of bits increases as a better accuracy is required. For each value of V, represented by the markers in Fig. 3c, we also report the corresponding average ES energy consumption in Fig. 3d, to assess how much the energy can be reduced by acting on the Lyapunov parameter V.



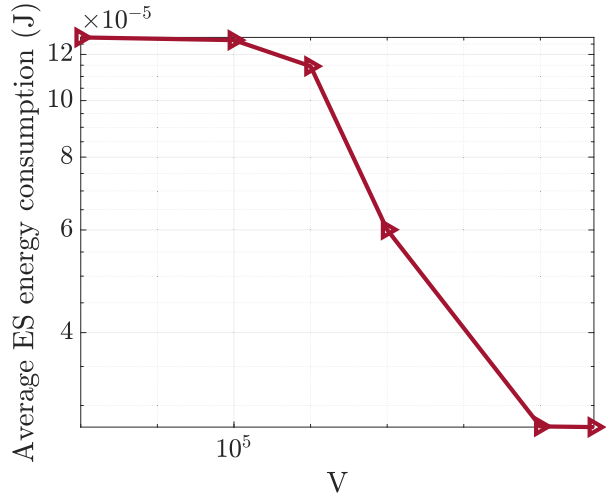
(a) Energy-delay trade-off for different accuracy constraints



(b) Mean square Deviation vs. t



(c) Avg. number of quantization bits vs. V



(d) Avg. ES energy consumption vs. V

FIGURE 3. **Energy-delay-accuracy trade-off.**

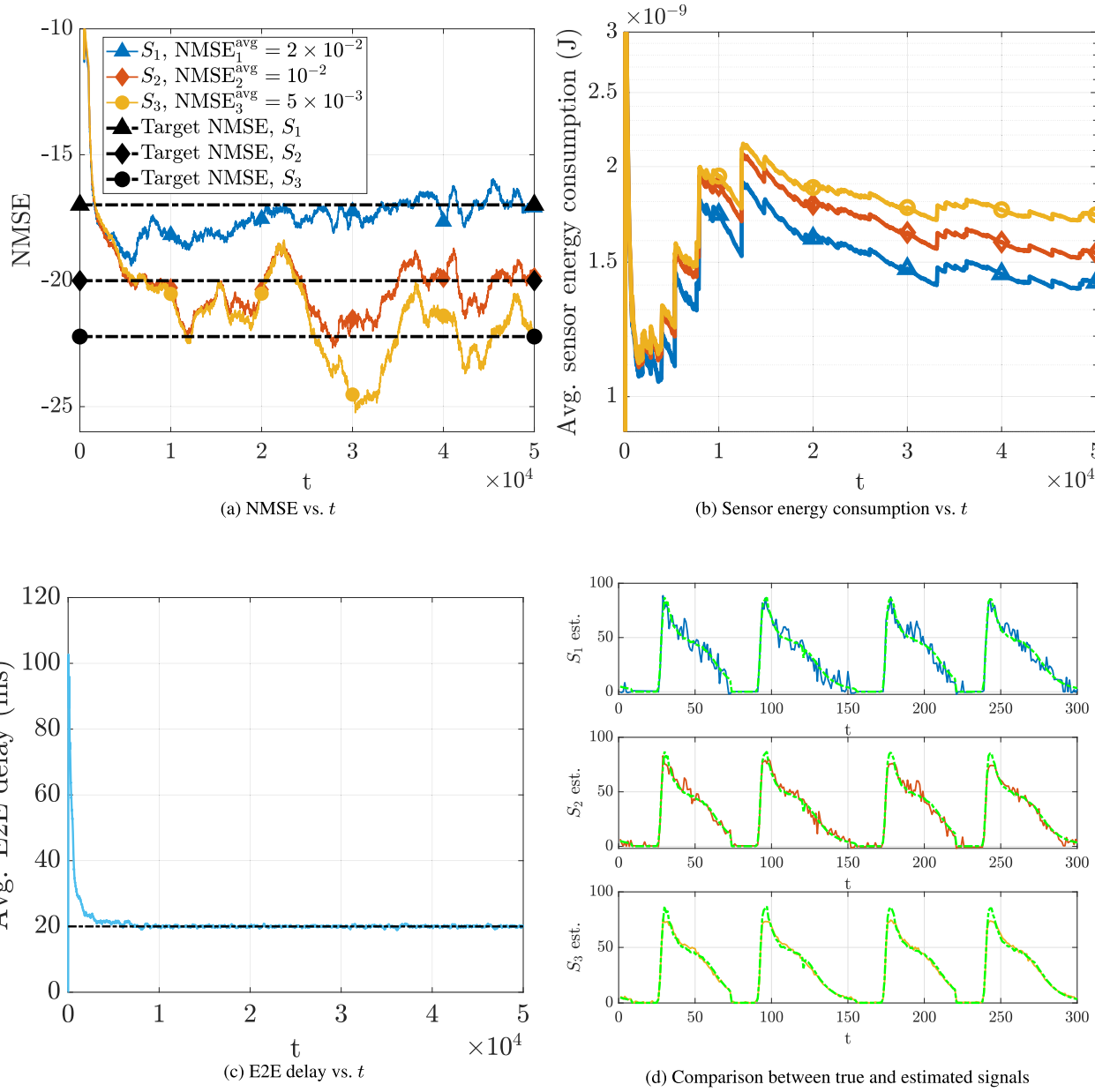
Interestingly, the trade-offs achieved by the different strategies are different, because of the different accuracy constraints. In particular, the best accuracy strategy (green curve ■) achieves, asymptotically, the minimum MSD, as evidenced in Fig. 3b, but at the same time exhibits the worst energy-delay trade-off, as shown in Fig. 3a. Conversely, the minimum energy strategy (blue curve ▲) achieves the best energy-delay trade-off, but at the same time it converges to the largest MSD. What is interesting to notice from Fig. 3, besides the two extreme cases, it is the behaviour of the intermediate strategies. We can notice how, tolerating some deterioration of the final MSD, with respect to the best accuracy strategy, we can achieve a substantially better energy-delay trade-off, as evidenced by the strategies S_2, S_3, and S_4.

In summary, the message coming from the results shown of Fig. 3 is twofold:

1) Our method is able to strike an optimal energy-delay trade-off, depending on the requirements concerning the accuracy level;
2) Acting on a single parameter, i.e. the V parameter, and adapting the number of bits/sample online, it is possible to significantly reduce the energy consumption, under a given E2E delay constraint, by slightly relaxing the accuracy constraint.

Clearly, it is not always desirable to obtain the best possible accuracy, if this leads to a too large energy cost. With our method, it is possible to evaluate how to achieve a better energy-delay trade-off, accepting some degradation of the final accuracy.

Data-driven minimum energy LMS estimation. In this paragraph, we illustrate the performance of the data-driven approach given by Algorithm 4. In this case, as a measurable performance metric for the accuracy (cf. (24)), we use the Normalized Mean Squared Error (NMSE) between the true signal and our online estimate/prediction. In particular, we consider the instantaneous estimation of the NMSE, which translates in choosing $|W_k| = 1$ in (24), with W_k being the set composed only by the last data sample. To compute this metric, there is an additional back and forth data exchange between AP and edge device. For simplicity, we neglect the time needed for this exchange of (scalar) information.



(a) NMSE vs. t

(b) Sensor energy consumption vs. t

(c) E2E delay vs. t

(d) Comparison between true and estimated signals

FIGURE 4. **Performance of data-driven Edge Machine Learning.**

To assess the performance of this approach, we use a real dataset composed by measurements of gas sensors exposed to dynamic mixtures of carbon monoxide (CO) and humid synthetic air inside a gas chamber[5] [48], [49]. Each of the K sensors transmits $U_k = 18$ regressors and a scalar observation to perform its LMS estimation. We assume that all K sensors use the same dataset for the sake of comparisons, assuming that each sensor has a different accuracy requirement on the NMSE. Then, we consider a similar scenario as in the previous simulation, with $K = 3$ sensors generating data with Poisson arrivals with parameter $A_k^{\text{avg}} = 1$. The data are quantized with dithered uniform quantization, and the set of number of bits/sample is $\mathcal{N}_k^q = \{3, 4, 5, 6, 7, 8, 9, 10\}$. The step size of LMS is set to $\mu_k = 10^{-6}$. The requirements on the NMSE for the three sensors are $\mathcal{G}_k^{\text{avg}} = \text{NMSE}_k^{\text{avg}} = [2, 1, 0.6] \times 10^{-2}$, $k = 1, 2, 3$; the average delay requirement is set to 20 ms. For this simulation, we set $V = 5 \times 10^8$, and choose $\widetilde{G}_k(n_k^q) = 2^{-2n_k^q}$ as the approximating decreasing function for the accuracy in (26).

In Fig. 4, we illustrate the temporal behavior of the NMSE (Fig. 4a), the average energy over time (Fig. 4b), and the average E2E delay over time (Fig. 4c), equal for all users. Finally, the comparison between the estimated and the true signals in a time window of 300 slots is shown in Fig. 4d. The target

[5]https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+temperature+modulation

NMSE values for the three sensors are indicated by the dashed lines on Fig. 4a. As we can notice from Fig. 4a, each device converges on average to the target NMSE. As expected, from Fig. 4b, we notice that the device with the best NMSE (yellow curve •) requires the highest energy consumption, due to the higher average number of quantization bits (around 6 bits are necessary on average to meet the constraint). On the contrary, the device requiring the worst NMSE (blue curve ▲) achieves the minimum energy (around 3.5 bits are sufficient on average to meet the constraint). The effect of the different accuracy constraints is clearly visible from Fig. 4d where, for each device, we show the time varying signal and its estimate/prediction. In particular, from Fig. 4d, setting a lower target NMSE (i.e., going from the first to the third sensor), we can see a noticeable improvement in estimation performance. Finally, in Fig. 4c we can notice how the E2E delay converges to the desired value. In summary, the proposed data-driven solution in Algorithm 4 is able to reduce the system energy consumption, while ensuring a target learning accuracy in terms of NMSE and a given E2E delay constraint.

B. CLASSIFICATION VIA SUPPORT VECTOR MACHINES

In this section, we customize our framework to SVM classification [50], [51]. SVMs are one of the most popular algorithms used for binary classification problems. In the case of a linearly separable dataset, the goal of SVMs is to find a hyperplane, amongst the infinite ones able to discriminate two classes, such that the distance between the hyperplane and the training data points that lie the closest to the hyperplane is maximized. This distance is called *margin*, while the closest points are called *support vectors*. In practice, it is quite typical to incur in situations in which the classes are not linearly separable. In these cases, the usage of *kernel functions* [52] helps in projecting the data into a higher dimension space, where the patterns become linearly separable.

In their native definition, SVMs are able to solve only binary classification problems. In the literature, two main strategies were proposed for multi-class SVMs. The first one, known as *one-against-one* [53], [54], trains $s(s-1)/2$ SVMs (with s being the number of classes), where each SVM is dedicated in separating two of the s classes. Once the training of all SVMs is completed, a new test pattern is elected by means of a majority vote. The second strategy is known as *one-against-all* [55], [56], where s SVMs are trained such that the i-th SVM sees the patterns belonging to the i-th class as positive and all other patterns as negative. The final classification is given by the SVM that marks the incoming pattern as positive. In the case of a tie among SVMs, a reliability score (e.g., the distance from the separating hyperplane) can be employed as a tie-breaker. Amongst the two, we choose the former strategy, being the most competitive in terms of training times and prediction accuracy [57].

Once an SVM is trained for a classification task, a meaningful performance metric is the correct classification rate. Denoting by y_i the true label of a given pattern, and by $\hat{y}_i$ its prediction, the correct classification rate reads as:

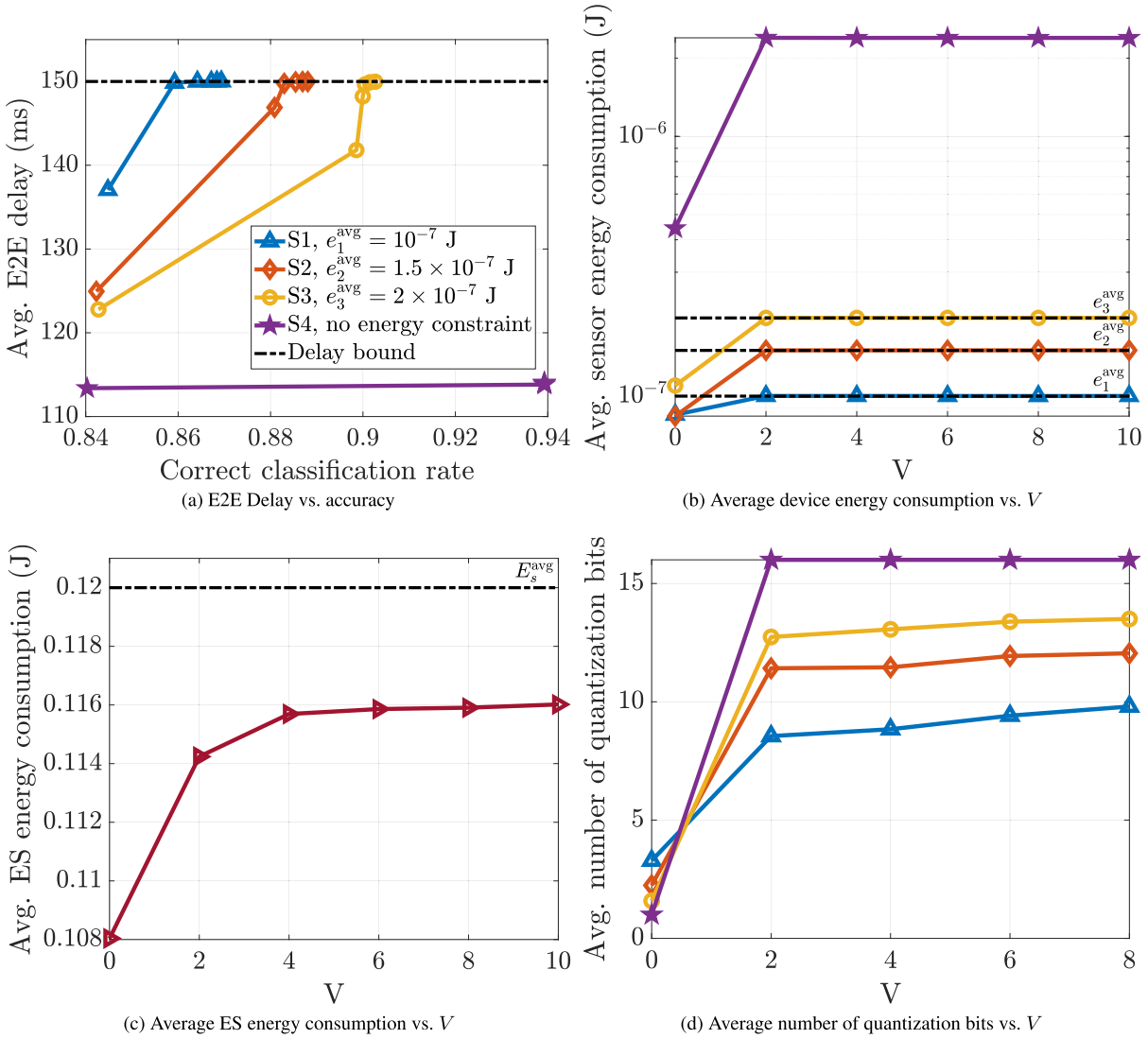
$$\mathcal{G} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{y_i = \hat{y}_i\}, \tag{43}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function, and N is the number of patterns in the test set. In the sequel, we will use (43) as the performance metric for the classification accuracy.

NUMERICAL RESULTS: BEST ACCURACY STRATEGY FOR SVM

We present now some numerical results for SVM classification at the edge, using the strategy proposed in Section IV, aimed at maximizing the accuracy under latency and energy constraints. In such a case, the ES runs an SVM classification task on the data uploaded by the end devices. We consider the same scenario of the previous section, with $K = 4$ sensors generating data from the MNIST dataset [58], with Poisson arrivals with parameter $A_k^{\text{avg}} = 2$. MNIST patterns have dimensionality $M_k = 784$ features/samples, i.e., the number of pixels. At the ES, an SVM with polynomial kernel classifies the data. Each sample is quantized with a number of bits $n_k^q \in \mathcal{N}_k^q = \{1, 2, 4, 8, 16\}$ bits. We assume that each of the 4 sensors has a different requirement on the energy spent for transmission, to simulate the situation in which the devices have a different battery energy level and then adapt their requirement in terms of energy consumption in order to prolong the battery lifetime. More specifically, the average energy constraints are set to $e_k^{\text{avg}} = [1, 1.5, 2] \times 10^{-7}$ J for the first 3 devices, while no energy constraint is imposed to the fourth one, meaning that it can possibly transmit with its maximum power. The ES average energy constraint is $E_s^{\text{avg}} = 0.12$ J, while the average E2E delay constraint is $D_k^{\text{avg}} = 150$ ms for all devices. The slot duration is $\tau = 50$ ms, the carrier frequency is $f_0 = 6$ GHz, with a 10 MHz bandwidth equally shared among devices. The conversion factor J_k (cf. (8)) has been estimated to be equal to 2.8×10^{-7}, with the same procedure used in Section V-A.

The performance of the proposed strategy is illustrated in Fig. 5, in terms of trade-off between energy, delay, and accuracy. In particular, Fig. 5a shows the average E2E delay vs. the obtained correct classification rate; whereas, Figs. 5b, 5c and 5d illustrate the average device and ES energy consumption, and the number of quantization bits, all as functions of the Lyapunov parameter V. Note that, Fig. 5a is obtained by increasing V from left to right, with the values visible in Figs. 5b, 5c and 5d. We can notice form Fig. 5 that the different sensors show different trade-offs, but all devices meet the required energy and delay constraints. In particular, the device without energy constraint (purple curve ★) shows the best accuracy, with the lowest average E2E delay, but it pays this performance with the highest energy consumption, as clearly visible from Fig. 5b. At the same time, the device with the lowest energy constraint (blue curve ▲), given the delay bound of 150 ms, shows the worst performance in terms of correct classification rate, due to the fact that it needs to



(a) E2E Delay vs. accuracy

(b) Average device energy consumption vs. V

(c) Average ES energy consumption vs. V

(d) Average number of quantization bits vs. V

FIGURE 5. Energy-delay-accuracy performance for SVM edge classification (MNIST dataset).

lower the average number of quantization bits (see Fig. 5d) to meet the energy and delay constraints. All other devices show intermediate performance in terms of energy, delay and accuracy. Finally, from Fig. 5c, we can notice that the ES always meets its energy constraints.

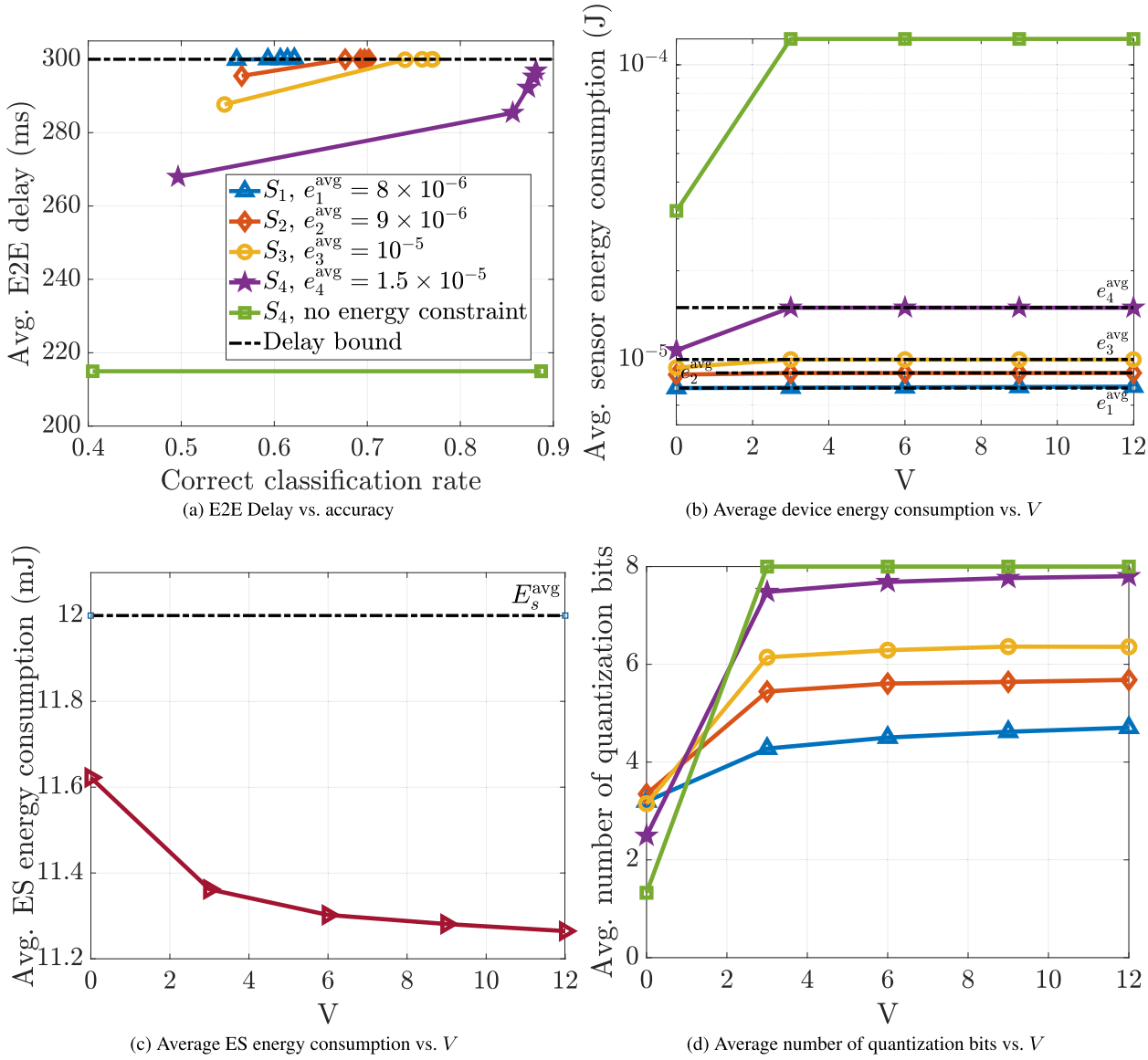
To summarize, the take-home message of Fig. 5 is twofold:

1) Our method is able to achieve the best accuracy while guaranteeing constraints on the device and ES energy consumption, and on the E2E delay;
2) By decreasing the maximum allowed energy consumption, as expected, a device has to sacrifice the learning performance to guarantee the same average E2E delay.

These results further show the power of Lyapunov optimization in exploring the energy-delay-accuracy trade-off, thus guaranteeing the best learning performance under E2E delay and energy constraints.

C. CLASSIFICATION WITH NEURAL NETWORKS

In this section, we customize our framework to a classification task implemented by a Neural Network. NNs have emerged as a fundamental tool for classification in pattern recognition, representing a valid and promising alternative to various conventional classification methods [59]. An NN for classification aims at learning a functional relationship between the group membership and the attributes of the object. In this sense, NNs represents powerful learning tools since: i) they are universal functional approximators, i.e., they can approximate any function with arbitrary accuracy; ii) they are data driven self-adaptive methods, i.e., they can adjust themselves to the data without any explicit specification of functional or distributional form for the underlying model. In the sequel, we assume a Multi-Layer Perceptron (MLP) structure composed of two layers. The hidden layer has 10 units with hyperbolic tangent sigmoid activation function. The output layer uses the softmax function, and the network is trained via scaled conjugate gradient backpropagation in order to minimize the cross-entropy loss function. The MLP weights are initialized by following the Nguyen-Widrow initialization algorithm, that is, by making sure that the active regions of the layer's neurons are distributed approximately



(a) E2E Delay vs. accuracy

(b) Average device energy consumption vs. V

(c) Average ES energy consumption vs. V

(d) Average number of quantization bits vs. V

FIGURE 6. Energy-delay-accuracy performance for NN edge classification (Hydraulic System Monitoring dataset).

evenly over the input space. We used the Hydraulic System Monitoring (HSM) dataset[6] [60]. In particular, this dataset considers physical measurements from several sensors (temperature, volume, pressure etc.) to infer the working condition of a hydraulic system. We used the pressure sensors as features. Then, 4 possible classes (conditions) are considered: i) optimal pressure; ii) slightly reduced pressure; iii) severely reduced pressure; iv) close to total failure.

NUMERICAL RESULTS: BEST ACCURACY STRATEGY FOR NN

Here, we present some numerical results for NN-based classification at the edge, using the strategy presented in Section IV, aimed at maximizing the accuracy under latency and energy constraints. In such a case, the ES runs an NN classification task on the data uploaded by the end devices.

[6] https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems

We consider the same scenario of the previous section, with $K = 5$ sensors generating data from the HSM dataset, with Poisson arrivals with parameter $A_k^{\text{avg}} = 1$. In our setting, HSM patterns have dimensionality $M_k = 36000$ features. At the ES, an NN classifies the data, which can be quantized with $n_k^q \in \mathcal{N}_k^q = \{1, 2, 3, 4, 5, 6, 7, 8\}$ bits. We assume that each of the 5 sensors has a different requirement on the energy spent for transmission, in order to compare different solutions in terms of final accuracy. The device average energy constraints are set to $e_k^{\text{avg}} = [0.8, 0.9, 1, 1.5] \times 10^{-5}$ J for the first 4 devices, while no energy constraint is imposed to the fourth one. The ES average energy constraint is $E_s^{\text{avg}} = 12$ mJ, while the average E2E delay constraint is $D_k^{\text{avg}} = 300$ ms for all devices. The slot duration is $\tau = 100$ ms, the carrier frequency is $f_0 = 6$ GHz, with a 10 MHz bandwidth equally shared among devices. The conversion factor J_k (cf. (8)) has been estimated to be equal to 3.54×10^{-7}.

The performance of the proposed strategy is illustrated in Fig. 6, in terms of trade-off between energy, delay, and accuracy. As for the SVM, Fig. 6a shows the average E2E delay vs. the obtained correct classification rate, whereas, Figs. 6b, 6c and 6d show the average device and ES energy consumption, and the number of quantization bits, respectively, all as a function of the Lyapunov parameter V. Fig. 6a is obtained by increasing V from left to right, with the values visible in Figs. 6b, 6c and 6d. Again, all devices meet the required energy and delay constraints. The device without energy constraint (green curve ■) shows the best accuracy, with the lowest average E2E delay. This is again paid with a much higher energy consumption than the other devices (cf. Fig. 6b). At the same time, the device with the lowest energy constraint (blue curve ▲), given the delay bound of 150 ms, shows the worst performance in terms of correct classification rate, due to the fact that it needs to lower the average number of quantization bits (see Fig. 6d) to meet the energy and delay constraints. All other devices show intermediate performance in terms of energy, delay and accuracy. Finally, from Fig. 6c, we can notice that the ES always meets its energy constraint. Thus, similar conclusions as for SVM edge classification can be drawn.

VI. CONCLUSION AND FUTURE DIRECTIONS

In this work, we have devised dynamic resource allocation strategies for executing machine learning tasks at the wireless network edge, hinging on the trade-off between energy, delay, and learning accuracy. The main novelty of our approach is to incorporate the learning accuracy into the search for the optimal balance between energy consumption and E2E service delay. The proposed methods are based on Lyapunov stochastic optimization, which enables the derivation of low-complexity algorithms able to work without a priori knowledge of a number of context features, such as task arrivals or channel state.

The first proposed strategy is aimed at minimizing the energy expenditure under E2E delay and learning accuracy constraints. We showed how to change the energy-delay trade-off curve, by acting on the learning accuracy requirement. Whenever it is possible to write closed form expressions for the learning performance, we derived a consequent model-based solution. In all cases where a model is not available, we have proposed a purely data-driven approach that measures performance in an online fashion. Both model-based and data-driven approaches have been customized and tested on a training task carried out using an LMS algorithm for estimation/prediction purposes, both on synthetic and real datasets.

The second proposed strategy aims at optimizing the learning accuracy under E2E delay and energy constraints. This strategy has then been applied to an inference task at the edge server, i.e., SVM and NN classification, under delay and energy constraints. Several numerical results on two real data illustrate the performance of the proposed approaches in terms of energy-delay-accuracy trade-off.

The proposed methods are very general and can be customized to several supervised, unsupervised, or semi-supervised learning tasks. Further developments are possible, in different directions. In this work, we used a simple source scalar encoder, but more sophisticated encoders can be used, like a vector quantizer, to achieve a better rate-distortion pair. Moreover, the stochastic optimization has been built on a drift-plus-penalty formulation, with a fixed parameter V, used to explore the energy-delay trade-off. Alternative approaches can be followed, adapting the value of the parameter V, depending on the behavior of the overall system.

APPENDIX

Here, we present the derivation of the upper bound of the Lyapunov drift-plus-penalty that leads to the per-slot optimization strategies of the different algorithms. First of all, note that, given a generic virtual queue $X(t)$ evolving as

$$X(t+1) = \max\left(0,\ X(t) + x(t+1) - \bar{x}\right),$$

and defining $\Delta_X = \frac{X^2(t+1) - X^2(t)}{2}$, we can always write [41]

$$\Delta_X \leq \frac{(x(t+1)-\bar{x})^2}{2} + X(t)x(t+1) - X(t)\bar{x}. \quad (44)$$

This inequality will be useful for all the upper bounds derived in this work.

A. MATHEMATICAL DERIVATIONS FOR SECTION III

We now present the mathematical derivations use to obtain the upper bound in (17). By applying (44) to the virtual queue $Z_k(t)$ defined in (13), we can write

$$\begin{aligned}\Delta_Z \leq\ & \frac{\left(Q_k^{\text{tot}}(t+1) - Q_k^{\text{avg}}\right)^2}{2} + Z_k(t)Q_k^{\text{tot}}(t+1) \\ & - Z_k(t)Q_k^{\text{avg}} = \frac{1}{2}\left(Q_k^{\text{tot}}(t+1)\right)^2 + \frac{1}{2}\left(Q_k^{\text{avg}}\right)^2 \\ & - Q_k^{\text{tot}}(t+1)Q_k^{\text{avg}} + Z_k(t)Q_k^{\text{tot}}(t+1) - Z_k(t)Q_k^{\text{avg}} \\ \leq\ & \left(Q_k^{l}(t+1)\right)^2 + \left(Q_k^{m}(t+1)\right)^2 - Z_k(t)Q_k^{\text{avg}} \\ & + Z_k(t)Q_k^{\text{tot}}(t+1) - Z_k(t)Q_k^{\text{avg}}, \end{aligned} \quad (45)$$

where we used the fact that $(x+y)^2 \leq 2x^2 + 2y^2$. Now, for $A, b \geq 0$ we have [41];

$$(\max(0, Q-b) + A)^2 \leq Q^2 + A^2 + b^2 + 2Q(A-b) \quad (46)$$

Then, recalling the evolution of the physical queues (6) and (8), we can write

$$\begin{aligned}\Delta_Z \leq\ & \left(Q_k^{l}(t)\right)^2 + \left(Q_k^{m}(t)\right)^2 + (A_{k,\max})^2 + (N_{k,\max}^{u})^2 \\ & + 2Q_k^{l}(t)\left(A_k(t) - N_k^{u}(t)\right) + (N_{k,\max}^{u})^2 \\ & + (N_{k,\max}^{c})^2 + 2Q_k^{m}(t)\left(N_k^{u}(t) - N_k^{c}(t)\right) \\ & + Z_k(t)\left(Q_k^{\text{tot}}(t+1) - Q_k^{\text{avg}}\right) \end{aligned} \quad (47)$$

where $N_{k,\max}^{u} = \frac{\tau R_{k,\max}}{M_k n_{k,\min}^{q}}$ (cf. (5)) and $N_{k,\max}^{c} = \tau f_{\max} J_k$ (cf. (7)). For the virtual queue $Y_k(t)$ (cf. (14)), by applying

(44), we can write:

$$\Delta_Y \leq \frac{\nu_k^2\left(G_k(t)-\mathcal{G}_k^{\text{avg}}\right)^2}{2}+\nu_k Y_k(t)\left(G_k(t)-\mathcal{G}_k^{\text{avg}}\right) \leq \frac{\nu_k^2\left(G_k^{\max}-\mathcal{G}_k^{\text{avg}}\right)^2}{2}+\nu_k Y_k(t)\left(G_k(t)-\mathcal{G}_k^{\text{avg}}\right), \quad (48)$$

where we used the fact that $G_k(t) \leq G_k^{\max}$. Then, using (47), (48), and noting the fact that $\min(N_k^u(t), Q_k^l(t)) \leq \min(N_{k,\max}^u(t), Q_k^l(t))$, we can write the following upper bound of $\Delta_p(\Theta(t))$ defined in (16):

$$\begin{aligned}\Delta_p(\Theta(t)) \leq \zeta + \mathbb{E}\Bigg\{\sum_{k=1}^{K}\Big[&\chi_k(t)-2Q_k^l(t)N_k^u(t)\\&+2Q_k^m(t)(N_k^u(t)-N_k^c(t))\\&+Z_k(t)\Big(\max\left(0,Q_k^l(t)-N_k^u(t)\right)\\&+\max(0,Q_k^m(t)-N_k^c(t))\Big)\\&+\nu_k Y_k(t)G_k(t)\Big]+VE_{\text{tot}}(t)\Big|\Theta(t)\Bigg\}\end{aligned}$$

where ζ is a positive constant given by

$$\zeta=\frac{1}{2}\sum_{k=1}^{K}\Big[2\left(A_k^{\max}\right)^2+4\left(N_{k,\max}^u\right)^2+2(N_{k,\max}^c)^2+(Q_k^{\text{avg}})^2+\nu_k^2\left(G_k^{\max}-\mathcal{G}_k^{\text{avg}}\right)^2\Big],$$

and $\chi_k(t)$ is a constant a time t, given by

$$\begin{aligned}\chi_k(t)=&(2Q_k^l(t)+Z_k(t))A_k(t)+(Q_k^l(t))^2+(Q_k^m(t))^2\\&+Z_k(t)\min(N_{k,\max}^u(t),Q_k^l(t))\\&-Z_k(t)Q_k^{\text{avg}}-\nu_k Y_k(t)\mathcal{G}_k^{\text{avg}}\end{aligned}$$

B. MATHEMATICAL DERIVATIONS FOR SECTION IV

Here, we present the mathematical manipulations used to obtain upper bound (31) in section IV. In particular, for the virtual queue $S_k(t)$ defined in (28), using (44), we can write

$$\Delta_S \leq \frac{\lambda_k^2\left(e_k(t)-e_k^{\text{avg}}\right)^2}{2}+\lambda_k S_k(t)\left(e_k(t)-e_k^{\text{avg}}\right) \leq \frac{\lambda_k^2\left(e_k^{\max}-e_k^{\text{avg}}\right)^2}{2}+\lambda_k S_k(t)\left(e_k(t)-e_k^{\text{avg}}\right) \quad (49)$$

where we used $e_k(t) \leq e_k^{\max}$. Similarly, for virtual queue $O(t)$ defined in (29), we can write

$$\Delta_O \leq \frac{\left(E_s(t)-E_s^{\text{avg}}\right)^2}{2}+O(t)\left(E_s(t)-E_s^{\text{avg}}\right) \leq \frac{\left(E_s^{\max}-E_s^{\text{avg}}\right)^2}{2}+O(t)\left(E_s(t)-E_s^{\text{avg}}\right) \quad (50)$$

Thus, using (47), (49), (50), and again noting the fact that $\min(N_k^u(t), Q_k^l(t)) \leq \min(N_{k,\max}^u(t), Q_k^l(t))$, we obtain the following upper bound of $\Delta_p(\Theta_2(t))$ in (31):

$$\begin{aligned}\Delta_p(\Theta_2(t)) \leq \zeta_2 + \mathbb{E}\Bigg\{\sum_{k=1}^{K}\Big[&\chi_{k,2}(t)-2Q_k^l(t)N_k^u(t)\\&+2Q_k^m(t)\left(N_k^u(t)-N_k^c(t)\right)\\&+Z_k(t)\Big(\max\left(0,Q_k^l(t)-N_k^u(t)\right)\\&+\max(0,Q_k^m(t)-N_k^c(t))\Big)\\&+\lambda_k S_k(t)e_k(t)-Vn_k^q(t)\Big]\\&+O(t)E_s(t)\Big|\Theta_2(t)\Bigg\}\end{aligned}$$

where ζ_2 is a positive constant given by

$$\zeta_2=\frac{1}{2}\sum_{k=1}^{K}\Big[2\left(A_k^{\max}\right)^2+4\left(N_{k,\max}^u\right)^2+2(N_{k,\max}^c)^2+(Q_k^{\text{avg}})^2+\lambda_k^2\left(e_k^{\max}-e_k^{\text{avg}}\right)^2\Big]+\frac{\left(E_s^{\max}-E_s^{\text{avg}}\right)^2}{2},$$

and $\chi_{k,2}(t)$ is a constant at time t, which reads as

$$\begin{aligned}\chi_{k,2}(t)=&(2Q_k^l(t)+Z_k(t))A_k(t)+\left(Q_k^l(t)\right)^2+\left(Q_k^m(t)\right)^2\\&+Z_k(t)\min(N_{k,\max}^u(t),Q_k^l(t))-Z_k(t)Q_k^{\text{avg}}\\&-\lambda_k S_k(t)e_k^{\text{avg}}-O(t)E_s^{\text{avg}}.\end{aligned}$$

ACKNOWLEDGMENT

The authors would like to thank Dr. Alessio Martino for his support in the section related to SVM and NN classification tasks.

REFERENCES

[1] E. C. Strinati, S. Barbarossa, J. L. Gonzalez-Jimenez, D. Ktenas, N. Cassiau, L. Maret, and C. Dehos, "6G: The next frontier: From holographic messaging to artificial intelligence using subterahertz and visible light communication," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 42–50, Sep. 2019.

[2] S. Ahmadi, *5G NR: Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*. Amsterdam, The Netherlands: Elsevier, 2019.

[3] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, and K. W. Wen, "MEC in 5G networks," ETSI, Sophia Antipolis, France, White Paper 28, 2018, pp. 1–28, vol. 28.

[4] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[5] J. Malmodin and D. Lundén, "The energy and carbon footprint of the global ICT and E&M sectors 2010–2015," *Sustainability*, vol. 10, no. 9, pp. 1–31, Aug. 2018.

[6] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[8] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[9] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018.

[10] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.

[11] M. Merluzzi, P. D. Lorenzo, and S. Barbarossa, "Dynamic joint resource allocation and user assignment in multi-access edge computing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 4759–4763.

[12] M. Merluzzi, P. D. Lorenzo, S. Barbarossa, and V. Frascolla, "Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 342–356, 2020.

[13] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2017, pp. 1–7.

[14] M. I. Ashraf, C.-F. Liu, M. Bennis, W. Saad, and C. S. Hong, "Dynamic resource allocation for optimized latency and reliability in vehicular networks," *IEEE Access*, vol. 6, pp. 63843–63858, 2018.

[15] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.

[16] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[17] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[18] Y. Mao, C. You, J. Zhang, K. Huang, and K. Letaief, "Mobile edge computing: Survey and research outlook," 2017, *arXiv:1701.01090v3*. [Online]. Available: https://arxiv.org/abs/1701.01090v3

[19] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[20] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proc. IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[21] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[22] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[23] H. Hellström, J. M. B. da Silva, Jr., V. Fodor, and C. Fischione, "Wireless for machine learning," 2020, *arXiv:2008.13492*. [Online]. Available: http://arxiv.org/abs/2008.13492

[24] N. Skatchkovsky and O. Simeone, "Optimizing pipelined computation and communication for latency-constrained edge learning," *IEEE Commun. Lett.*, vol. 23, no. 9, pp. 1542–1546, Sep. 2019.

[25] U. Mohammad and S. Sorour, "Adaptive task allocation for mobile edge learning," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshop (WCNCW)*, Apr. 2019, pp. 1–6.

[26] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1432–1436.

[27] A. Elgabli, J. Park, A. S. Bedi, M. Bennis, and V. Aggarwal, "Q-GADMM: Quantized group ADMM for communication efficient decentralized machine learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 8876–8880.

[28] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 63–71.

[29] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, "An energy efficient IoT data compression approach for edge machine learning," *Future Gener. Comput. Syst.*, vol. 96, pp. 168–175, Jul. 2019.

[30] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[31] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," 2016, *arXiv:1610.02527*. [Online]. Available: http://arxiv.org/abs/1610.02527

[32] P. H. Jin, Q. Yuan, F. Iandola, and K. Keutzer, "How to scale distributed deep learning?" 2016, *arXiv:1611.04581*. [Online]. Available: http://arxiv.org/abs/1611.04581

[33] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*. [Online]. Available: http://arxiv.org/abs/1812.06127

[34] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*. [Online]. Available: http://arxiv.org/abs/1602.05629

[35] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," 2019, *arXiv:1909.02362*. [Online]. Available: http://arxiv.org/abs/1909.02362

[36] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Feb. 2020.

[37] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, nos. 2–3, pp. 203–221, Aug./Sep. 1996.

[38] L. Brochard, V. Kamath, J. Corbalán, S. Holland, W. Mittelbach, and M. Ott, *Energy-Efficient Computing and Data Centers*. Hoboken, NJ, USA: Wiley, 2019.

[39] M. Merluzzi, P. D. Lorenzo, and S. Barbarossa, "Dynamic resource allocation for wireless edge machine learning with latency and accuracy guarantees," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 9036–9040.

[40] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, Jun. 1961.

[41] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.

[42] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[43] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[44] A. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends Mach. Learn.*, vol. 7, nos. 4–5, pp. 311–801, 2014.

[45] R. Abdolee and B. Champagne, "Diffusion LMS strategies in sensor networks with noisy input data," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 3–14, Feb. 2016.

[46] S. Sun, T. S. Rappaport, S. Rangan, T. A. Thomas, A. Ghosh, I. Z. Kovacs, I. Rodriguez, O. Koymen, A. Partyka, and J. Jarvelainen, "Propagation path loss models for 5G urban micro- and macro-cellular scenarios," in *Proc. IEEE 83rd Veh. Technol. Conf. (VTC Spring)*, May 2016, pp. 1–6.

[47] R. A. Wannamaker, S. P. Lipshitz, J. Vanderkooy, and J. N. Wright, "A theory of nonsubtractive dither," *IEEE Trans. Signal Process.*, vol. 48, no. 2, pp. 499–516, Feb. 2000.

[48] J. Burgués, J. M. Jiménez-Soto, and S. Marco, "Estimation of the limit of detection in semiconductor gas sensors through linearized calibration models," *Analytica Chim. Acta*, vol. 1013, pp. 13–25, Jul. 2018.

[49] J. Burgués and S. Marco, "Multivariate estimation of the limit of detection by orthogonal partial least squares in temperature-modulated MOX sensors," *Analytica Chim. Acta*, vol. 1019, pp. 49–64, Aug. 2018.

[50] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[51] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 144–152.

[52] J. Mercer, "Functions of positive and negative type, and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London A, Containing Papers Math. Phys. Character*, vol. 209, pp. 415–446, May 1909.

[53] U. H.-G. Kreßel, *Pairwise Classification and Support Vector Machines*. Cambridge, MA, USA: MIT Press, 1999, p. 255–268.

[54] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[55] V. Vapnik, *The Nature of Statistical Learning Theory*. London, U.K.: Springer-Verlag, 1995.

[56] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[57] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[58] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[59] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.
[60] N. Helwig, E. Pignanelli, and A. Schütze, "Condition monitoring of a complex hydraulic system using multivariate statistics," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I MTC)*, May 2015, pp. 210–215.

MATTIA MERLUZZI (Graduate Student Member, IEEE) received the M.S. degree in telecommunication engineering and the Ph.D. degree in information and communication technologies from Sapienza University of Rome, Italy, in 2017 and 2021, respectively. He held a visiting research appointment at CEA-Leti, Grenoble, France, in 2019. He is currently a Postdoctoral Researcher with the Department of Information Engineering, Electronics, and Telecommunications, Sapienza University of Rome. He has participated in the H2020 EU/Japan project 5G-Miedge. He is also participating in the H2020 EU/Taiwan project 5G CONNI and the MIUR funded PRIN Liquid Edge. His research interests include edge computing, 5G systems, stochastic optimization, and edge machine learning.

PAOLO DI LORENZO (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical engineering from Sapienza University of Rome, Rome, Italy, in 2008 and 2012, respectively. He held a visiting research appointment with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA, USA, in 2010. From May 2015 to February 2018, he was an Assistant Professor with the Department of Engineering, University of Perugia, Perugia, Italy. He is currently an Associate Professor with the Department of Information Engineering, Electronics, and Telecommunications, Sapienza University of Rome. He has participated in the FP7 European research projects FREEDOM, on femtocell networks; SIMTISYS, on moving target detection and imaging using a constellation of satellites; and TROPIC, on communication, computation, and storage over collaborative femtocells. He is a Principal Investigator of the research unit in the H2020 European project RISE 6G. His research interests include signal processing theory and methods, distributed optimization, mobile edge computing, machine learning, and graph signal processing. He is also an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. He was a recipient of the three best student paper awards, such as the IEEE SPAWC10, EURASIP EUSIPCO11, and the IEEE CAMSAP11, and the 2012 GTTI (Italian National Group on Telecommunications and Information Theory) Award for the Best Ph.D. thesis in communication engineering.

SERGIO BARBAROSSA (Fellow, IEEE) received the M.S. and Ph.D. degree in EE from Sapienza University of Rome. He has held visiting positions at the Environmental Research Institute of Michigan, in 1988, the University of Virginia, in 1995 and 1997, and the University of Minnesota, in 1999. He is currently a Full Professor and a Senior Research Fellow of the Sapienza School of Advanced Studies, Sapienza University of Rome. He has coauthored the papers that received the Best Student Paper Award at ICASSP 2006, SPAWC 2010, EUSIPCO 2011, and CAMSAP 2011. He has been the scientific coordinator of several EU projects on wireless sensor networks, small cell networks, distributed mobile cloud computing, and edge computing in 5G networks. He is also leading a national project on edge learning and he is involved in two H2020 European projects on 5G networks for Industry 4.0 and on reconfigurable intelligent surfaces. His current research interests include mobile edge computing and machine learning, graph signal processing, and distributed optimization. From 1997 to 2003, he was a member of the IEEE Technical Committee for Signal Processing in Communications. He is a EURASIP Fellow. He received the IEEE Best Paper Award from the IEEE Signal Processing Society, in 2000, 2014, and 2020, and the Technical Achievements Award from the EURASIP Society, in 2010. He has been the General Chairman of the IEEE Workshop on Signal Processing Advances in Wireless Communications (SPAWC), in 2003, and the Technical Co-Chair of SPAWC, in 2013. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, from 1998 to 2000 and from 2004 to 2006, the *IEEE Signal Processing Magazine*, and the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS. He has been the Guest Editor for Special Issues on the IEEE Journal on Selected Areas in Communications, the *EURASIP Journal of Applied Signal Processing*, the *EURASIP Journal on Wireless Communications and Networking*, the *IEEE Signal Processing Magazine*, and the IEEE SELECTED TOPICS ON SIGNAL PROCESSING. He has been an IEEE Distinguished Lecturer.