Received December 23, 2020, accepted January 5, 2021, date of publication January 11, 2021, date of current version January 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3050841

Distributed and Multi-Task Learning at the Edge for Energy Efficient Radio Access Networks

MARCO MIOZZO, ZORAZE ALI, LORENZA GIUPPONI, (Senior Member, IEEE), AND PAOLO DINI
CTTC/CERCA, 08860 Barcelona, Spain

Corresponding author: Marco Miozzo (marco.miozzo@cttc.es)

The research leading to these results received funding by the Spanish Government under project TEC2017-88373-R (5G-REFINE) and Generalitat de Catalunya under grant 2017 SGR 1195. It was also supported by Huawei Technologies, Sweden AB.

ABSTRACT The big data availability of Radio Access Network (RAN) statistics suggests using it for improving the network management through machine learning based Self Organized Network (SON) functionalities. However, this may increase the already high energy consumption of mobile networks. Multi-access Edge Computing can mitigate this problem; however, the machine learning solutions have to be properly designed for efficiently working in a distributed fashion. In this work, we propose distributed architectures for two RAN SON functionalities based on multi-task and gossip learning. We evaluate their accuracy and consumed energy in realistic scenarios. Results show that the proposed distributed implementations have the same performance but save energy with respect to their correspondent centralized versions and benchmark solutions. We conclude the paper discussing open research issues for this interesting emerging field.

INDEX TERMS Distributed learning, edge intelligence, energy efficiency, Green AI, mobile networks.

I. INTRODUCTION

Mobile communications have become part of our daily lives. However, the fifth generation (5G) of mobile technology is expected to introduce a new revolution by bringing enhanced broadband services everywhere, smart vehicles and transportation, and complex human machine interactions (e.g., extended reality) [1], [2]. Such a connected society will generate a considerable amount of data: CISCO estimates that the internet traffic will increase up to 805 ZB by 2021 [1], with an annual growth rate of 20.6 ZB in 2021. Big data availability and processing will be the key driver for the booming of the Artificial Intelligence (AI), whose application is expected to significantly enrich people's lifestyle, improve human productivity and enhance social efficiency.

Nevertheless, various reports indicate that mobile networks already have a huge carbon footprint and the situation is worsening: their energy consumption is expected to reach the 51% of the worldwide electricity generation by 2030 [3]. Consequently, an energy sustainable design of next generation mobile networks architecture and algorithms represents one of the key requirements of 5G and beyond in order to ensure cost effectiveness and reduce the negative impact on the environment.

As a key driver that boosts 5G performance, Self-Organizing Networks (SON) functionalities represent an important building block enabling an automatic network management. By learning from the experience and adapting to the changing environment, SON functionalities are able to maximize the efficiency of the network, while at the same time reducing the operational costs. 5G cellular networks are characterized by extremely dense and heterogeneous deployments, such that coverage and capacity are increased. In addition, the high diversity of mobile devices and applications, further complicates the network architecture and its management. In this context, current and next generation networks generate a massive amount of measurements, control and management information [4], [5]. This huge amount of information could be efficiently utilized to address the 5G network management challenges. Recently, the evolution in computational capabilities, has allowed to take advantage of Machine Learning (ML) and novel Deep Learning (DL) solutions to tackle multiple problems in different disciplines. In mobile networks and its evolution toward the 6G wireless

The associate editor coordinating the review of this manuscript and approving it for publication was Haruna Chiroma.

communications paradigm, the possibilities now available for ML implementations are infinite and pave the way to an evolved vision of Next Generation SON to be able to address intelligent end-to-end solutions [6].

There is a complexity downside, though, that normally is not taken into account when picturing this intelligent, autonomous and DL-based Next Generation SON. In fact, DL is based on Artificial Neural Networks (ANNs) of many neurons and layers that need a big amount of data in order to learn the huge number of parameters they are composed of. This represents an extremely high computational complexity, which requires an equally high energy consumption. For example, training a single Natural Language Processing (NLP) ML model is equivalent to 284 tons of carbon dioxide emission, i.e., five times the lifetime emissions of an average car [7]. Therefore, both industry and academia need to consider the trade-off between the improvement in the performance and the energy consumed for a sustainable design of AI solutions, which is the aim on the emerging Green AI [8]. From an ethical perspective, the environmental costs have been included among the nine main requirements and practical specifications for a transparent development and implementation of AI [9]. In particular, the energy required from an AI solution, as well as its model sensitivity to hyperparameters, should be investigated in order to provide a comprehensive evaluation.

To respond to such energy needs, whilst meeting the rate and latency requirements of the underlying services, the next cornerstone in the mobile network domain is represented by the integration with Multi-access Edge Computing (MEC), which amounts to performing computation right at the network edge [10]. By empowering Base Stations (BSs) with processing capabilities, MEC will avoid unnecessary communication latency, enabling faster responses and higher privacy for end users. Moreover, MEC solutions allow consuming 25% less energy than conventional data centers [11] by reducing the need of communications and the dimension of the cooling system. However, adding MEC processes to Mobile Network Operators (MNOs) will further exacerbate their electrical power consumption, which is already responsible of a major part of their operational expenditures [12]. Hence, the integration of mobile network and edge computing domains needs a proper sustainable design. To do so, we advocate an energy-aware design of the radio access segment and the computing infrastructure of the mobile network based on the *edge intelligence* (EI) [13], [14], which enables distributed computing of ML models at the network edge. EI will facilitate the management of data coming to the edge, like Internet of Things (IoT), human and machine based. In particular, the scope of EI is to distributively train ML models (training) and run ML models (inference) [13]. Several solutions have been already proposed to this respect [13], mainly with the focus of parallelizing the models. In addition, EI has been already identified as one of the key missing elements in 5G networks and it is expected to become a key enabling factor for future 6G networks [15].

Consequently, we consider that an accurate study of the different distributed learning solutions in terms of accuracy and energy consumption, is of paramount importance for their efficient and energy-aware application.

Another paradigm which is attracting interest in the community is represented by the Multi-Task Learning (MTL) [16]. MTL aims to leverage useful information contained in multiple related tasks to improve the generalization process of all the tasks and to boost the performance. Thanks to the online, parallel and distributed principles, in conjunction with dimensionality reduction, MTL models can be used to speed-up the learning process, by sharing training models of high dimensional spaces, among multiple tasks.

In this work, we focus on studying the energy and accuracy performance of EI solutions for SON functionalities. In detail, we investigate distributed ML paradigms that can rely on the data acquired by the network elements, collected at the edge, and distributively perform the training. We compare them against centralized implementations, where data are transferred from the edge to a central entity performing the training and finally distributing actions to be taken at the edge. Without loss of generality, we consider two Radio Access Network (RAN) use cases: handover decision and the selection of the initial Modulation and Coding Scheme (MCS). We believe that these use cases can benefit from the introduction of sophisticated DL solutions, taking into account all the historical information available at all the layers of the protocol stack, which brings added value in terms of users' Quality of Experience (QoE). To do this, we set up a realistic cellular scenario using a high fidelity, full protocol stack, end-to-end network simulator, *ns-3*, and we extract statistics from all the layers of the RAN protocol stack. In order to exploit the temporal characteristic of the extracted data, we adopt ML models based on Recurrent Neural Network (RNN). In particular, we consider different Long-Short-Term-Memory (LSTM) architectures for implementing predictors based both on single-task and multi-task paradigms. To parallelize the training of the two uses cases, we adopt MTL, which has been implemented through Autoencoders (AEs). For the distribution of training, among various options, we propose Gossip Learning (GL) paradigm [17], [18], which allows to perform training by peer-to-peer information exchange, thus enabling a full asynchronization and total decentralization. As a result, the contributions of the paper are summarized in the following list:

- Design of distributed learning architectures using GL and MTL.
- Design of predictors for testing two RAN use cases with both single-task and multi-task paradigm, and in distributed and centralized version.
- Performance evaluation in realistic scenarios of the proposed distributed solutions both in terms of accuracy and consumed energy and comparison with respect to their correspondent single-task and centralized versions.

- Discussion of the main open issues to be addressed for enabling an effective and energy sustainable development of EI techniques in mobile networks.

We present a set of results, which demonstrate that prediction can be successfully performed in distributed fashion. Moreover, we prove that the approach based on QoE outperforms traditional solutions from network performance perspective for both use cases compared with state-of-the-art algorithms (i.e., considering instantaneous signal strength for the handover and taking the most conservative option for the MCS). In addition to that, regarding the energy consumption, we show that the distribution of training process based on GL, allows to save energy, without compromising the prediction performance. Similarly, MTL also results beneficial in terms of used energy when multiple tasks have to implemented. These results are novel, and valid for two different RAN use cases that we have considered, and to the best of the authors' knowledge still have not been discussed in the literature.

The rest of the paper is organized as follows. In Section II, we discuss the background. In Section III, we introduce the methodology considered in this paper, including the problem statement and an overview of the different ML solutions used. In Section IV, we describe the generation of the database. In Section V, we analyze the achieved performance in terms of error behavior during the training, QoE and energy consumption. In Section VI, we present open research challenges concerning the integration of the edge computing into future mobile networks. Finally, in Section VII, we draw our conclusions.

II. BACKGROUND

The architectural availability of edge computing resources to execute AI directly at the edge, lately has been attracting significance attention, also due to Ultra-Reliable Low Latency Communications (URLLC) scenarios, like factory automation, autonomous driving, remote surgery, and augmented/virtual reality. In fact, locating the data processing in proximity of its origin, e.g., at the edge, offers manifolds benefits, that can be classified in these four Key Performance Indicators (KPIs):

- *Computation*: the algorithms work with local information, which implies a reduced amount of data and, thus, the use of less demanding hardware, both in terms of computational and memory requirements. On the other hand, it will require the use of specific techniques for distributing the training and inference models.
- *Communication*: proximity allows to reduce the transmission hops of the data, and hence network congestion.
- *Privacy*: distributed data permits to keep it safer locally, which prevents leakage [19].
- *Energy*: the advantages in computation and communication enable to reduce the energy consumption of the whole system.

In order to efficiently exploit data on the edge, the *edge intelligence* paradigm, also called *edge AI* [13], [14], aims at evaluating distributed solutions to run ML models (the *inference* phase) and to train ML models (the *training* phase).

For what concerns the inference at the edge, the main problem is the limited resources of the devices. In this case, the solutions aim to relax the computational requirements of the model during the inference phase. In *model compression*, some of the weights can be pruned according to a specific policy, e.g., their magnitude [20], the energy [21]. In *model early-exit*, the inference is performed only with a subset of the network, according to the latency requirements. Whereas, *partition* [22] and *input filtering* [23] represent interesting solutions for reducing the computational complexity on the device model, which relay on pre-processing the data on the device and perform the inference on the edge.

With respect to the training phase, the main problem of a distributed solution is the convergence of a consensus, i.e., how fast and whether the training can be considered finalized. This problem is related to how the gradient is synchronized and updated. The most popular solution for distributed training is represented by the Federated Learning (FL) [24]. In this solution, the server is in charge of combining the results of the training of a shared model with specific Stochastic Gradient Descent (SGD) methods, such as the Selective SGD [25]. However, the SGD methods are not optimized for working with unbalanced and non independent and identical distribution (iid) data. The frequency of the updates of the model at the central server is also an open issue. Too frequent updates allow to relax the hardware constraints of the edge, but result in a higher risk of unreliable network communications. Another interesting solution is the Knowledge Transfer Learning (KTL) [26], where a teacher network is trained with general data and then student networks are retrained on a more specific local dataset. This allows to reduce the resource demand at the edge devices. Similarly, GL [17], [18] allows for a total decentralized paradigm and is based on randomized gossip algorithms. GL works by finding the convergence towards a consensus among nodes, by exchanging information in a peer-to-peer fashion, thus removing the requirement on centralized nodes or variables, as for FL and KTL. Moreover, the full decentralized and asynchronous nature of GL makes it a valuable approach for RAN SON functionalities, since it allows to limit the coordination among the nodes and to reduce the complexity of the management architecture on interfaces like X2/Xn, which are usually characterized by high latency and low bandwidth.

Regarding the possible applications, AI will play an important role also for providing solution to the resource management problem in edge computing, the so called Intelligence-enabled Edge Computing (IEC), which is complementary to the problems presented above, where the issue is how to carry out the ML process on the edge, or *AI on the Edge* (AIE). Typical examples of IEC are Radio Resource Management in wireless networking, computation offloading strategies and services placement and caching. In this case, the challenges are on the model definition, which often has to be defined as a tractable Markov decision process, on the algorithm deployment, since it has to work on-line. Consequently, a trade-off between optimality and efficiency has to be found. In case of Radio Resource Management (RRM), individual rule-based algorithms can be replaced by a general-purpose learning framework capable of autonomously generating complex algorithms specialized for each RRM functionality. This type of framework, when based on DL applied to the raw network data, can be able to improve the feature-engineered solutions based on standard ML designed to solve specific tasks in a restricted environment and, possibly, solve multiple tasks [27].

Up to now, the work has been mainly concentrated on IEC and, in particular, on RRM optimization solutions for improving the performance of the system in edge-computing architectures, mainly on IoT scenarios. In this case, ML has been applied to improve the resource allocation and the spectrum management for offloading computational-intensive tasks from resource-limited machine-type devices to powerful edge servers [28], [29].

Regarding AIE, some work has been already published in the area of wireless communications, especially on FL for IEC [30]. FL has been applied in [31] to learn the statistical properties of vehicular users for ultra-reliable low-latency vehicular communications. Authors in [32] applied FL to learn the locations and orientations of the users in wireless virtual reality networks for minimizing the breaks in presence. A device scheduling problem for heterogeneous resources in MEC based on FL has been proposed in [33]. In [34], a FL based approach has been used for mobile packet classification, which allows mobile devices to collaborate and to train a global model, without sharing users' sensitive information. The authors demonstrate its effectiveness in terms of classification performance using three real-world datasets. Energy-efficient radio resource allocation for enabling FL in an edge scenario has been investigated in [35] by adapting the communication to devices' channel states and computation capacities so as to reduce their energy consumption while guaranteeing learning performance.

Another ML paradigm that can help in reducing the implementation complexity without sacrificing ANN's universal function approximation property is represented by MTL [36]. As KTL, MTL is animated by the human learning principle of transferring the acquired knowledge: often people apply some ability, learned from previous tasks, to help learn a new task. In MTL, all the tasks are treated with the same priority and the objective is to improve the performance of all the tasks. Whereas, in KTL, the target is to improve the performance of a specific task with the help of a source task. An example of usage of MTL in mobile communications is represented by [37], where multi-task Sparse Bayesian Learning (SBL) has been applied for learning time-varying sparse channels in the uplink for multi-user massive MIMO systems. Results showed that it is possible to considerably reduce the complexity and the required time for the convergence with negligible sacrifice of the estimation accuracy. In [38], multi-task deep ANN framework for non-orthogonal multiple access (NOMA), namely DeepNOMA, has been proposed for treating non-orthogonal transmissions as multiple distinctive but correlated tasks.

In this work we would like to evaluate jointly the two AI MEC paradigms by investigating the performance of AIE solutions with a specific IEC application (i.e., the two RAN SON functionalities). In particular, for AIE we consider GL as a distributed training model together with MTL and we study their performance both in terms of prediction accuracy and wasted energy, which have not been done till now in these scenarios to the best of our knowledge.

III. METHODOLOGY

A. PROBLEM STATEMENT

In this work we consider next-Generation Node B (gNB) architecture [39], where each BS will be composed of a Central Unit (CU) and one or more Distributed Units (DUs), which have computation capabilities to store the different algorithms of the protocol stack as defined in the Open RAN (O-RAN) paradigm [40]. In detail, we rely on the concept of decoupling the Control-Plane (CP) from the User-Plane (UP) into RAN for bringing in the CP embedded intelligence by introducing the RAN Intelligent Controller (RIC). Without loss of generality, in this paper we focus on two RAN SON use cases for the RIC: Handover Decision (HD) and Initial MCS Selection (IMS). We selected them in such a way that they are sufficiently different tasks of the RAN, and they are traditionally handled by different layers of the protocol stack. In particular, handover management is normally handled by Radio Resource Control (RRC) functions, whereas the selection of the MCS is a Medium Access Control (MAC) layer problem. We do that to derive conclusions as general as possible from the solutions proposed for these use cases. We consider that both use cases will benefit from an ML oriented approach, as we discuss in the following.

Regarding the HD, in standards and literature, mobility algorithms are traditionally based on standard events defined by 3GPP specifications [41], e.g., the A3 or A2 event, and are mainly focused on the optimization of event trigger parameters, e.g., Hysteresis, Time-to-Trigger and Cell individual Offset [42]. In this respect, many ML solutions have been proposed to dynamically adjust online these parameters [4]. However, all the solutions present the same limitation: they consider only some representation of the signal power for evaluating which decision to make, but not the actual perceived QoE after the decision. A typical problem of HD in urban scenarios is the presence of many obstacles, that may cause that the handover to the strongest neighbor cell is successful but, a while after, the transmission is deeply affected. In such cases, traditional handover approaches based on instantaneous signal strength are not able to provide a satisfactory solution, since they cannot take advantage of available data to gain experience and make smarter decisions. Those approaches are likely to severely affect the QoE of the users, due to the unpredicted cell outage [43].

Similarly, for what concerns the IMS, the initial MCS adopted by a device when starting a connection is commonly selected in a very conservative fashion, e.g., by taking the MCS with the lowest spectral efficiency. In this way, the first communication will be delivered with the lowest bit error probability independently from the device's position, which is usually unknown at the beginning. However, for those UEs that are in good coverage, this solution penalizes their initial performance.

We propose different ANN architectures to estimate the QoE that the users are perceiving, based on the data that are generated by the BS during its normal operation. We extract data from the complete protocol stack, from the transmission related parameters of the PHY layer (e.g., transmission power, Reference Signal Received Power, hybrid automatic repeat request feedback) to the ones of MAC (e.g., MCS, resource blocks usage, retransmissions, protocol data unit size and delay), up to application layer (e.g., inter packet delay, instantaneous throughput). In particular, our ANNs are designed to predict the QoE parameters of the two use cases: i) the time to finalize the download for HD and ii) the throughput perceived in the initial window for IMS.

The ML models that we propose are based on LSTM, so as to exploit the temporal characteristic of the data extracted from the mobile network. These solutions have been implemented in both centralized and distributed architectures. In the rest of the paper we will focus on the following architectures:

- *centralized*: where all the local data are collected in a common server to perform the training of the LSTM-based solutions.
- *distributed*: where the data are maintained at the BS premises for being processed by the local MEC server. Only the model parameters are exchanged by the different BSs after performing the local training.
- *multi-task*: adopted for enhancing the efficiency of the training phase of the two use cases. MTL relies on a shared AE based on LSTM for learning the most relevant features and predictors implemented with Multi-Layer Perceptrons (MLPs).

As benchmark, we consider the single-task architecture, where two dedicated LSTM predictors are used to estimate the time to finalize the download and the initial throughput perceived, respectively.

An example of the reference scenario architecture is provided in Fig. 1. The centralized solutions, placed in the central cloud server of Fig. 1, are: single-task LSTM predictor for HD (ST-HD), single-task LSTM predictor for IMS (ST-IMS), multi-task LSTM-AE with MLP predictor for HD and IMS (MT-HD and MT-IMS, respectively). The corresponding distributed extensions based on GL, deployed on the edge directly at the BS premises in left part of Fig. 1, are: single-task LSTM predictor for HD (GL-ST-HD), single-task LSTM predictor for IMS (GL-ST-IMS), multi-task LSTM-AE with MLP predictor for HD and IMS

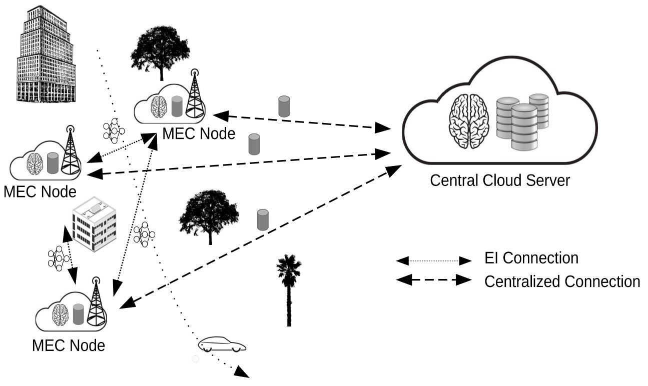

FIGURE 1. Reference scenario architecture.

(GL-MT-HD and GL-MT-IMS, respectively). Summarizing, in this work we have considered the implementations reported in Table 1.

TABLE 1. Implemented solutions.

Use case	Centralized		Distributed	
	Single-task	Multi-task	Single-task	Multi-task
HD	ST-HD	MT-HD	GL-ST-HD	GL-MT-HD
IMS	ST-IMS	MT-IMS	GL-ST-IMS	GL-MT-IMS

Our focus in this paper is on adapting and evaluating the distributed framework specifically in an EI scenario for RAN management SON functionalities. This includes the evaluation of two sensitive aspects: i) the behavior of the ML models, when considering GL and MTL paradigms, and ii) the assessment of the used energy with each architecture.

In what follows, we present the GL, MTL and LSTM based architectures together with their background information.

B. GOSSIP LEARNING

GL has been designed to handle the special cases of peer-to-peer data processing of distributed data for managing sensitive content that is better to process locally. The main idea behind GL is to avoid any synchronization among the nodes, so there is no need of any centralized entity in charge of managing the training phase. This allows to have a more robust algorithm, since it prevents the single point of failure problem and do not requires the synchronization among all the nodes and the central entity at each distributed training step. Moreover, GL guarantees a low communication overhead, since it needs a reduced number of messages to be exchanged and of a reduced size, i.e., only the model parameters have to be exchanged instead of the local database.

The generic skeleton of GL involves three main components: an implementation of random walk, an online learning algorithm and ensemble learning [17]. In this paper we consider a specific implementation of the GL, where the online learning method is SGD for all the ML models.

Algorithm 1 provides the generic skeleton of the GL framework. The same algorithm is run at each node in the network.

Algorithm 1 Skeleton of the Gossip Learning Algorithm

```
1: procedure Main
2:     currentModel ← InitModel()
3:     loop
4:         wait (Δ)
5:         p ← SelectRandomPeer()
6:         send currentModel to p
7:     end loop
8: end procedure
9: procedure OnReceiveModel(m)
10:    currentModel ←Merge(m, currentModel)
11: end procedure
```

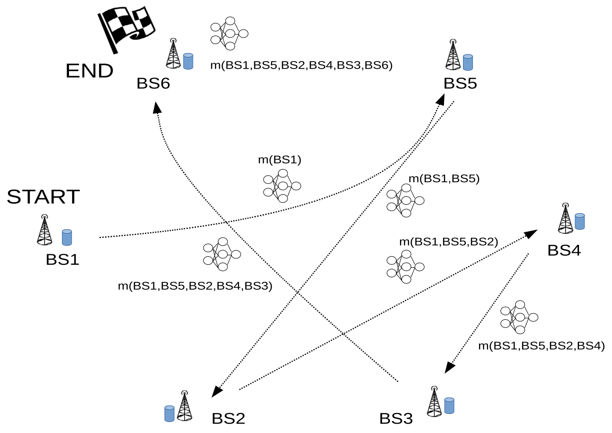

FIGURE 2. Example of GL organization of compute nodes.

The algorithm consists of an active loop of periodic activity, and a method to handle the reception of incoming models. We assume that the length of the period of the loop Δ is the same at all nodes. The original GL model is designed for working with fully distributed data, where each node is assumed to own a single, private data point, as for peer-to-peer application, e.g., recommender systems. In this case, nodes perform only one step of the learning process at each loop cycle, according to the available data. However, this may not be the case in many scenarios, where a single node might have multiple useful data points, such as image classification and our case. To this respect, the study in [44] shows that training on multiple data points provides a clear advantage over the original protocol, as models see more data in the same number of iterations, and thus converge faster. Therefore, we consider the extension proposed in [44], which propose to call multiple times the SGD on different data points, i.e., implementing at each node multiple training steps sequentially every loop cycle. In particular, we consider to perform the local training on the whole dataset generated by one BS in one loop cycle, thus the maximum number of steps of the algorithm will be the number of BSs. For simplicity, in the implemented version, nodes have been randomly ordered during the initialization, instead of distributively generating the random sequence during the algorithm. In this way, nodes are aware of when they receive a new model and to which node send the updated version. The active loop is initiated at the same time in all the nodes, and the stopping criteria is when the nodes reach a consensus. The consensus is usually dependent on the specific problem to be solved, e.g., the requested prediction loss. In our case, we consider as consensus the finalization of the training phase from all BSs. This allows us to compare the distributed approach against the centralized, since they have been trained with the same amount of data.

An example of the execution of the GL algorithm in a cell with 6 BSs is provided in Fig. 2. During the initialization, we pick up a random sequence for the distributed training, i.e., $\{1, 5, 2, 4, 3, 6\}$. This sequence will be used by the nodes to send their updated local model, as in Algorithm 1. In this case, the algorithm initiates in BS1, which starts training the model through SGD with its entire local dataset according the specific ML model: the LSTM predictor for GT-ST-HO and GT-ST-IMS, and the LSTM AE and the MLP for GT-MT-HD and GT-MT-IMS. After this, BS1 randomly sends the parameters of the model to another node, in our case BS5. BS5 initializes its internal local model with the parameters received by BS1 and performs the training with its local dataset. The trained parameters are then randomly passed to the next node, BS2. This procedure is repeated until the last BS in the sequence has performed its local training, i.e., BS6.

C. MULTI-TASK LEARNING

The general motivation of the MLT is to obtain some underlying functions for prediction, which can define what good predictors should be like. The critical issue is to obtain such functions by simultaneously taking various prediction problems into consideration. Traditional MTL methods assume that all the tasks are related and their dependencies can be modeled by a set of latent variables. However, in many real-world applications not all tasks are related, and enforcing erroneous (or non-existent) dependencies may lead to negative knowledge transfer/sharing.

In this work, we consider the Multi-Task Autoencoder Model (MTAM) [45]. MTAM extracts multiple features from the data thanks to the AE for the prediction of the two RAN tasks. The prediction is based on a MLP model, which is a standard class of feedforward ANN. The MTL paradigm has been proposed with the aim to enhance parameters estimation; however, in our study, we are more interested in its computational efficiency, since it allows to share the AE training phase among the different tasks.

Autoencoders are used in representation learning to learn a representation of the input in a feature space in unsupervised manner. We consider a *sequence-to-sequence* autoencoder [46], since our dataset consists of time-series sequences. The objective is to reconstruct the data samples using an *encoded representation* of the input sequence. An autoencoder is made of an encoder and a decoder. Let $\mathcal{X} = R^D$ be the input space and $\mathcal{F}$ be the feature space.

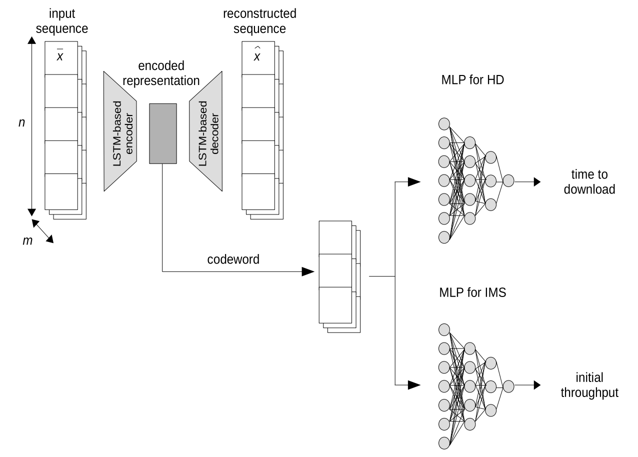

FIGURE 3. **MTL architecture diagram.**

An *encoder* is a function $\phi : \mathcal{X} \rightarrow \mathcal{F}$ that has to learn the prominent characteristics and generate an encoded version of the sample in the feature space $\mathcal{F}$. Alternatively, the *decoder* is a function $\psi : \mathcal{F} \rightarrow \mathcal{X}$ that aims to reconstruct the input using the internal representation. Formally, given a sample sequence $\mathbf{x}(n)$, the autoencoder is a function $\Phi_{AE} : \phi \circ \psi$ that outputs $\hat{\mathbf{x}}(n)$

$$\Phi_{AE}(\mathbf{x}(n)) = \hat{\mathbf{x}}(n). \quad (1)$$

When trained with sufficient samples, the architecture is able to learn the reconstruction of the normal samples with a low reconstruction error.

For the implementation of both the encoder and the decoder, we adopt LSTM cells, according to their capability of extracting the temporal dependencies from one instance to another. In detail, the encoder and the decoder architectures have been chosen through simulation trials and evaluation, and each one consists of two layers of LSTM cells, respectively. The first layer of the encoder ENC_1 has $N_{ENC_1} = 84$ cells and the second layer ENC_2 has $N_{ENC_2} = 100$ cells, which represents the dimension of the learned representation, or codeword, of the AE. The decoder is symmetrical: DEC_1 has N_{DEC_1} 100 cells and the second layer DEC_2 has $N_{DEC_2} = 84$ cells. Both encoder and decoder use *tanh* activation function, *sigmoid* recurrent activation function and *Adam* optimizer with mean square error minimization function for the training. The MLP used for the prediction has two fully connected layers. The first layer MLP_1 contains $N_{MLP_1} = 84$ neurons and the second layer MLP_2 has $N_{MLP_2} = 42$ neurons. *Leaky ReLU* activation function is used for hidden layers while linear activation is applied at the output layer. In addition, *RMSProp* optimizer with mean square error minimization function is used for the training. The resulting number of trainable parameters of each model based on MTL according to the hyperpameters of above are 235,620 for the AE and 6,774 for the MLPs. The input sequence will be described in Section IV. The diagram of the predictors based on the MTL architecture is depicted in Fig. 3.

D. LSTM SINGLE-TASK PREDICTOR

When used for performing a prediction, the LSTM algorithm receives a traffic sequence of length W at time n, $\mathbf{x}(n) = [\boldsymbol{x}(n), \boldsymbol{x}(n+1), .., \boldsymbol{x}(n+W-1)]$, and tries to predict the traffic sample at time $n+W$, $\hat{\boldsymbol{x}}(n+W)$

$$\Phi_{PRED}(\mathbf{x}(n)) = \hat{\boldsymbol{x}}(n+W). \quad (2)$$

We consider a stacked architecture that includes multiple LSTM layers. The number of concatenated cells in the first layer indicates the number of observations of the data, which in our case corresponds to the window length W. By doing so, LSTM are capable of learning long-term dependencies from the input time series, while solving the vanishing-gradient problem that affects standard RNN [47]. This capability is due to the structure of the basic LSTM cells (or units) that includes gates to regulate the learning process.

Through simulation trials and evaluation, we have selected four stacked layers combining three LSTM layers and a final fully connected output layer. The LSTM layers (respectively $LSTM_1$, $LSMT_2$ and $LSMT_3$) have $N_{LSTM_1} = 84$, $N_{LSMT_2} = 42$ and $N_{LSMT_3} = 21$ cells. The final fully connected layer uses *linear* activation function and its output consists of the prediction. *RMSProp* optimizer with mean square error minimization function is used for the training. According to these hyperparameters configuration, the number of trainable parameters of each model based on the LSTM predictor are: 83,346 for ST-HD and GL-ST-HD, 84,106 for ST-IMS and GL-ST-IMS. The input sequence will be described in Section IV.

IV. DATABASE DESCRIPTION

In order to test the proposed solutions, we use a synthetic database generated with a simulator. In detail, we implemented a realistic simulation scenario through the ns-3 LENA LTE (Long Term Evolution) - EPC (Evolved Packet Core) simulator [48]. A macro cell outdoor scenario has been considered with a network consisting of three-sectorial BSs. Each sector has a cluster of User Equipments (UEs) located at random positions and moving following a predefined mobility pattern. In order to mimic the communication challenges of an urban scenario, and generate random coverage patterns, we introduce obstacles in the scenario, thus generating multiple coverage holes, as shown in Fig. 4. Each UE performs a Transmission Control Protocol (TCP) file transfer to a remote host in downlink and uplink directions. Different databases have been generated in order to test the different ML architectures: DB-ST-HD and DB-ST-IMS for the single-task HD and IMS solutions, respectively, and DB-MT for the MLT. For the cases of single-task LSTM prediction of HD use case (ST-HD) the database is created for collecting info on this specific task. Therefore, the simulations are designed for performing deterministic handovers to different potential neighbors. In this scenario, we observe that the maximum number of neighbors a UE can perceive is 8; therefore, each run is repeated 8 times to measure the file download time when the handover is performed to one of the possible

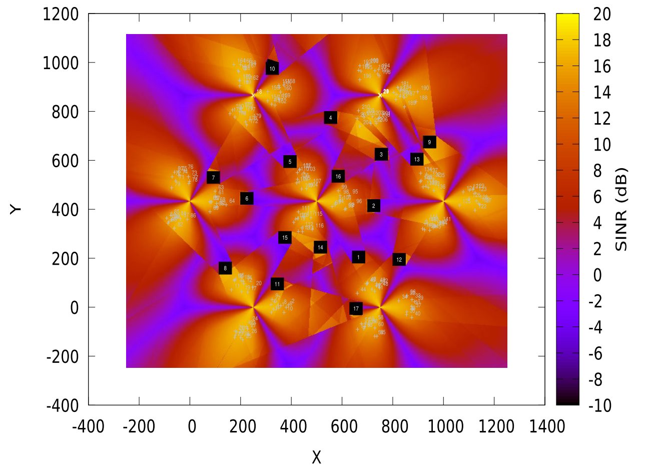

FIGURE 4. **Simulation scenario.**

neighbor BSs. For every simulation run, a UE picks a random starting position in the cluster and a random angle in the range of [0°, 360°] to move away from the source BS following a straight line. Regarding DB-ST-IMS, the simulations are designed for retrieving the throughput in the firsts 20 ms window when using the initial MCS. Without loss of generality, we analyze three MCS (0, 14 and 28) for generating this database, since they substantially differ in spectral efficiency and allow us to highlight their behavior according to the position of the UEs with respect to the BS. Finally, for generating DB-MT, the simulations are a combination of the two cases of above. In fact, the database DB-MT is common for the two tasks; therefore, each run is repeated 24 times for each user: 8 times to measure the download time and 3 times to measure the initial throughput. The data obtained from these simulation campaigns for each UE are stored in the form of a dataset, according to the format described in what follows.

We design the solutions of the two RAN use cases as a regression problem, where we need to estimate the QoE expected for performing handover to a specific target cell and the initial throughput when starting a new connection. Therefore, the algorithms are adopting the supervised learning paradigm, which implies that a labeled dataset is needed. In fact, in supervised learning, each entry is a pair consisting of an input object and the correspondent desired output value, e.g., the label. In our case, the inputs of these datasets consist of the configuration parameters and the set of features, and the outputs are the QoE parameters perceived. For what concerns the features, we extracted 84 measurements from all the layers of the LTE protocol stack. We gather the measurements using some logs/traces already available in ns-3, e.g., those available for RLC (Radio Link Control) and PDCP (Packet Data Convergence Protocol), and other new custom trace sources at RRC (Radio Resource Control), MAC and PHY, obtained by leveraging the tracing system of *ns-3*. The input features, for our dataset, are extracted with the periodicity of 200 ms in order to be consistent with the approximate periodicity with which UE measurements are reported from UEs at the RRC level. This dataset can be expressed as a 3D matrix $\overline{\mathbf{X}}$:

$$\overline{\mathbf{X}} = \begin{bmatrix} \overline{x}_{1,1} & \overline{x}_{1,2} & \cdots & \overline{x}_{1,m} \\ \overline{x}_{2,1} & \ddots & \cdots & \overline{x}_{2,m} \\ \vdots & \vdots & \overline{x}_{i,j} & \vdots \\ \overline{x}_{n,1} & \overline{x}_{n,2} & \cdots & \overline{x}_{n,m} \end{bmatrix}$$

where the feature vector of size 84 is $\overline{x}_{i,j} \in \overline{\mathbf{X}}$, $1 \leq i \leq n$, and $1 \leq j \leq m$. The upper limit of n can be computed by multiplying the total number of UEs with the maximum neighbor BSs to handover and/or the initial MCS to explore, and the total number of simulation runs. Whereas m defines the duration of the time series to be analyzed (i.e., the number of samples in the total simulation time, 40 sec, when sampling each 200 ms), which corresponds to number of time-steps that the LSTM processes to perform the prediction.

Regarding the simulation scenario, we consider 7 three-sectorial BSs, which corresponds to 21 sectors, and 10 UEs per sector, which results in a total of 210 UEs in the whole simulation field. For an exhaustive description of the simulation scenario and parameters, the reader can refer to [49]. During the simulations, it may happen that some of the data are not available, because UEs might experience a Radio Link Failure (RLF) when forced to handover to a BS with poor channel conditions. In those cases, we do not have data since the user is not connected, and consequently, we removed the affected entries. According to this, the number of entries of the databases for the overall simulation scenario, i.e., the parameter n, are: 33,500 for DB-ST-HD, 29,648 for DB-ST-IMS and 33,662 for DB-MT. These values correspond to the databases used for the centralized solutions, since they aggregate the data of all BSs. On the other hand, in the distributed versions, each node processes only the entries generated by the corresponding BS sector, which implies that the local databases dimension is smaller, i.e., approximately the 21st part of the dimensions for correspondent centralized databases.

V. RESULTS

The implementation of the proposed ML architectures relies on Keras and Tensorflow as backend. In particular, we adopted the fast LSTM implementation by Nvidia CUDA Deep Neural Network (CuDNN) library for Graphics Processing Units (GPUs) [50]. The used server has the following specs: 4 GPUs GeForce RTX 2080TI (4.352 cores 11 GB), 2 Central Processing Units (CPUs) Intel Xeon 6230 (20 cores) 2,1 GHz, 192 GB DDR4 2933 MHz of memory, 2 disk of 2TB SATA3 6GB/s.

A. TRAINING PHASE

The datasets have been randomly divided into training and validation sets, using a split ratio of 0.75 and 0.25, respectively. We train and validate the algorithms using the training and validation sets to minimize the reconstruction error over 200 epochs, in the case of the AE, or prediction error,

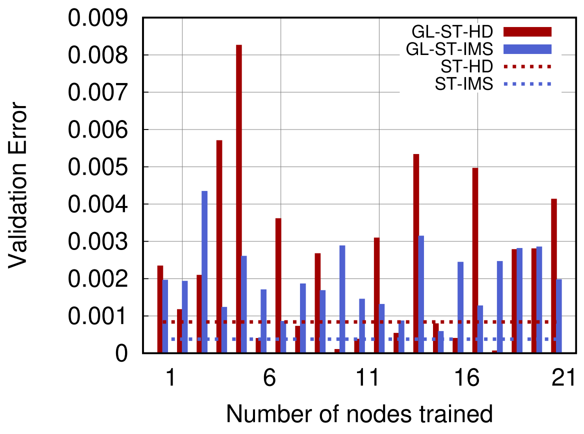

FIGURE 5. Validation losses for single-task architectures.

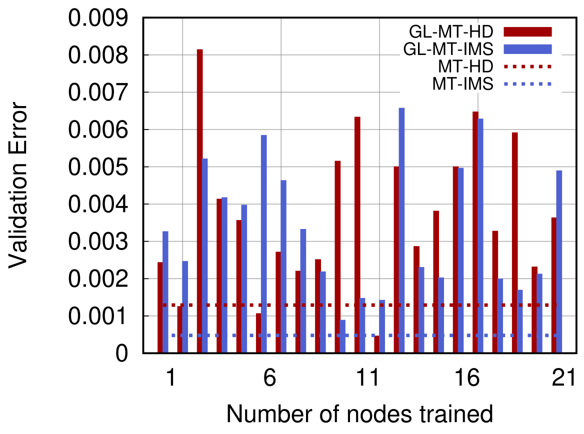

FIGURE 6. Validation losses for multi-task architectures.

in the case of the predictors. The loss function used to train the algorithm is the mean square error (MSE) and the *RMSProp* algorithm is used to optimize the learning process. The GL based solutions have been analyzed during its learning process among the nodes, i.e., analyzing the error of the ML model when different number of nodes have been visited and have performed their local training.

We start analyzing the single-task architectures and we present the validation errors over the number of visited nodes in Fig. 5. The centralized versions are presented as a horizontal line, since they are trained on the aggregated databases. The validation errors of both GL-ST-HD and GL-ST-IMS have an oscillating behavior when varying the number of visited nodes. The specific value change according to the GL training order; but the range and the oscillations are similar regardless the different random sequence that can be used. Therefore, we report the results only for one sequence during its learning process. GL-ST-HD and GL-ST-IMS have in general worst validation errors with respect to the correspondent centralized solutions. In detail, for what concerns HD use case, the GL-ST-HD solution presents both higher and lower values of validation error with respect to ST-HD. The GL-ST-HD has many values of lower validation error with respect to ST-HD in the middle of the training phase, i.e., when the number of nodes that have performed the training is between 7 and 18. Similarly, for the IMS use case, the values of validation error of the GL-ST-IMS solution are varying with the number of visited nodes and the values of high validation errors are concentrated at the beginning and at the end of the training phase. The higher validation errors for the distributed solutions at the beginning (i.e., when the number of visited nodes is low) is expected, since the model has been trained with a reduced number of samples. However, when the number of visited nodes is increasing, the distributed solutions still present higher validation errors in many steps. Moreover, the final validation errors of GL-ST-HD and GL-ST-IMS are higher with respect to the correspondent centralized versions. This phenomenon depends on the quality of the datasets, i.e., whether data are iid. If the data distribution changes while learning, the new data will interfere with already acquired knowledge. In this case, the model might experience performance degradation at previously learned concepts (i.e., the relation between the inputs and outputs of the model) when trained sequentially on learning new concepts, the so called *Catastrophic Forgetting* (CF) [51]. This is due the fact that SGD is sequentially applied to the local datasets; whereas, in the centralized version, SGD is executed on the randomized aggregated dataset, which has higher iid properties.

Considering the multi-task architectures, Fig. 6 reports the validation errors with respect to the number of visited nodes. In this case, the oscillations of the validation error are higher with respect to the single-task, especially after 7 nodes, where both GL-MT-HD and GL-MT-IMS experience only one value of validation error below the correspondent centralized solutions, MT-HD and MT-IMS, respectively. The same considerations done for the single-task architectures apply also in this case. In addition, comparing Fig. 5 with Fig. 6, we can see that there is no correlation among the peaks of high validation error between the single-task and the multi-task architectures. This implies that the data generated by each BS and the variation of validation error during the GL training are not correlated.

In this work we limit our study to the demonstration that a distributed learning implementation using GL is returning similar network performance compared to a centralized solution, as presented in Section V-B. Therefore, we do not investigate CF issue in more depth. However, we consider that CF has to be carefully investigated when applying distributed solution; thus, in Section VI-A we discuss some open issues on training when data cannot be assumed iid.

B. NETWORK PERFORMANCE

The network performance evaluation of these models is performed in a supervised offline fashion similar to [52], and detailed in what follows. We compare the network performance of benchmark solutions against the ones obtained with the predictions based on the centralized architectures and

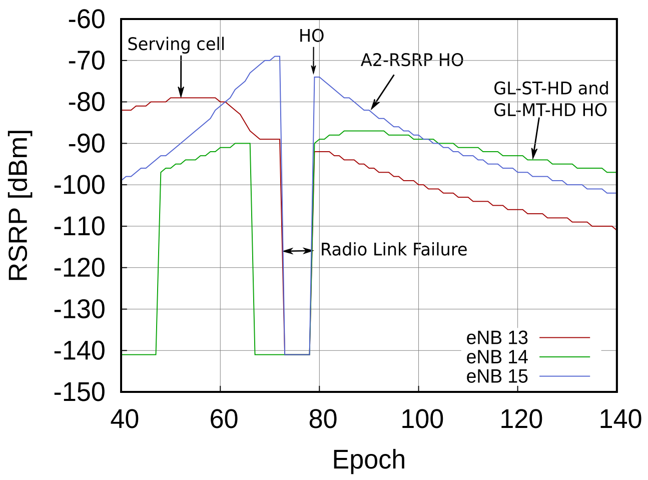

FIGURE 7. Example of handover behavior with A2-RSRP, GL-ST-HD and GL-MT-HD solutions.

the correspondent distributed. To do this, we consider the distributed version when the training is finalized, i.e., the GL algorithm has visited all the nodes, in order to have trained all the models with the same amount of data. The results based on the centralized architecture are equal to the distributed, so we can claim that the difference in validation error does not affect the prediction accuracy for the considered use cases. According to this, we only report the results of the distributed approaches for sake of brevity.

For the HD use case, we compare the real time to download for each UE, obtained after selecting the target cell providing the lowest predicted time to download, to the one achieved by using a benchmark approach, i.e., A2-RSRP based handover algorithm. In particular, to perform this evaluation we consider a test dataset generated with two extra simulation runs obtained using a seed value for the random number generator which was not used to build the training and validation datasets. For IMS use case, we evaluate the performance of the initial MCS, following an offline strategy, as we did for HD use case, considering a new testing dataset. We evaluate three different MCSs for each UE to get the correspondent predicted initial throughput. Then, for each UE we select the MCS, which results in the higher initial throughput and we compare it to the benchmark approach (i.e., select always MCS 0).

TABLE 2. Offline evaluation HD use case.

	no. of UEs finalizing the download	% of UEs decreasing the download time
A2-RSRP	78	-
GL-ST-HD	88	92
GL-MT-HD	90	81

In Table 2, we report the results on the HD use case obtained with the A2-RSRP, the GL-ST-HD and the GL-MT-HD. As we can see, both GL-ST-HD and GL-MT-HD outperform the benchmark solution in terms of number of UEs that are able to finalize the download and, among these, the download time is also reduced in most cases. We have analyzed the behavior of the UEs that have a better download time from signal strength perspective. In particular, we have evaluated the behavior of the handover algorithm in the different cases from Reference Signal Received Power (RSRP) perspective, which is stored in the database as it is collected by the UEs for all the neighbors independently from the handover solutions adopted. In Figure 7 we plot the RSRP as a function of the epoch time perceived by one of the UEs that have resulted with lower download time. The UE started attached to sector 13 until epoch 73 where it reaches an obstacle and it looses the connection, i.e., a Radio Link Failure (RLF) occurs. At epoch 80, the UE has already passed the coverage hole, and the A2-RSRP algorithm detects that sector 15 has the best RSRP and perform a handover to sector 15. Differently, GL-ST-HD and GL-MT-HD exploit the experience extracted by the data to select as target sector for the handover the cell that provide the best long term QoE, i.e., sector 14.

TABLE 3. Offline evaluation IMS use case.

	% of UEs with improved performances	% of increment of initial throughput
GL-ST-IMS	100	73.35
GL-MT-IMS	81.9	74.47

The results on the IMS use case are presented in Table 3. Results show that GL-ST-IMS and GL-MT-IMS outperform the baseline. In this case, GL-MT-IMS has a lower number of UEs with improved performance with respect to the GL-ST-IMS. Since this phenomenon happens also in the centralized version, the reason can be a low degree of relation among the two tasks, which jeopardizes the prediction performance of the IMS use case. In fact, in this case, the two tasks are learned simultaneously, thus CF does not represent an issue. It is to be noted that, even all the algorithms present lower validation error for the IMS use case with respect to HD, the specific MCS selection problem might require a higher precision to properly work.

C. ENERGY AND COMMUNICATION KPIs ASSESSMENT

In what follows, we investigate on the consumed energy by the different architectures for performing the training. In particular, we are interested in evaluating the energy figures of the distributed solutions with respect to the centralized ones. To do so, we use the Machine Learning Emission Calculator (MLEC) [53] and Green Algorithms (GA) [54] tools, which can provide the used energy according to the type of used hardware, the amount of used memory and the execution time. MLEC takes as input the details regarding the training of an ML model (i.e., the type of GPU, and the training time) and gives as output the approximate amount of Wh. GA is designed to have a wider application and removes the restrictions on the hardware and applications of MLEC. To do this, GA considers the running time, the number, type and

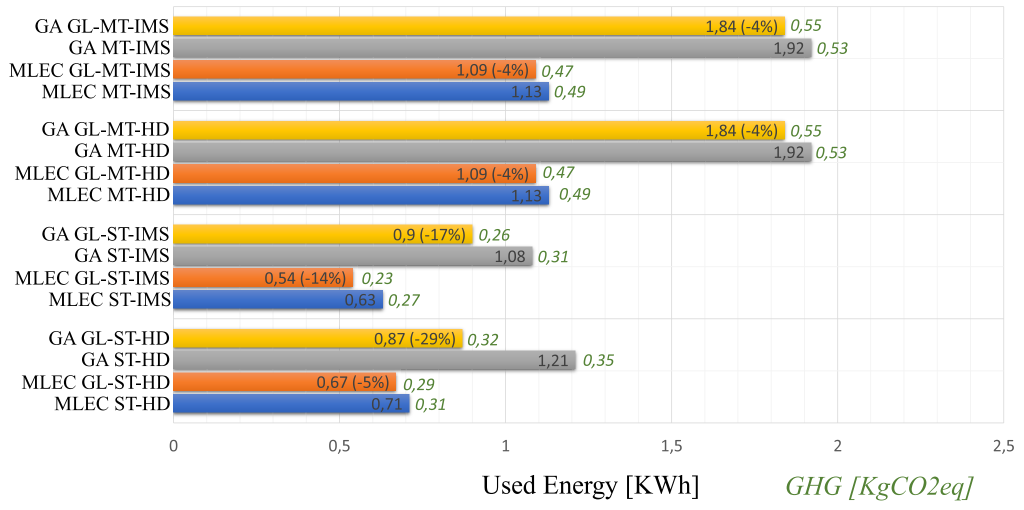

FIGURE 8. Energy (in black) and GHG emissions (in italic and green) assessment, the percentages in brackets refer to the savings when using the distributed solution with respect to the correspondent centralized.

process time of computing cores (CPU or GPU), the amount of used memory and the power draw of these resources. In detail, the energy consumption E (in KWh) is calculated as:

$$E = t \times (n_c \times P_c \times u_c \times n_m \times P_m) \times 0.001, \tag{3}$$

where t is the running time (hours), n_c the number of cores and n_m the size of memory available (gigabytes), u_c is the core usage factor (between 0 and 1), P_c is the power draw of a computing core and P_m the power draw of the memory (Watt). By providing the region where the training is performed, MLEC and GA can also estimate the approximate amount of Carbon Dioxide Equivalent (CO2e) produced. In this work we adopted a carbon efficiency of 0.432 kg/kWh, which is the 2014 yearly average value according to the Organization for Economic Co-operation and Development (OECD). The results are presented in Fig. 8 and provide the KWh used by each architecture (in black) and the correspondent Greenhouse Gas (GHG) emissions in terms of CO2e (in green and italic). We can see that the GL based solutions always reduce the energy consumption, especially in the single-task paradigm, where GL-ST-HD can reach up to 29% of energy savings according to GA. The lower energy savings of MTL can be motivated by the higher complexity of its architecture (242,394 parameters) with respect to the single-task (83,346 parameters). However, the different implementations of the two solutions affect the amount of work performed with their parameters. A more in-depth investigation on the relation between the consumed energy and their parameters for each proposed solution might help in clarifying this phenomenon; however, we consider it is out of the scope of this evaluation and we left it for future work.

In addition, we would like to observe that, the MTL architectures facilitate a more energy-efficient paradigm since the training of the AE is common for all tasks. In fact, we calculate that the energy consumed by the MLP is around 5% of the whole MTL solution: therefore, adding a new task in MTL would imply a marginal increment, i.e., the 5% of GL-MT-HD or GL-MT-IMS. Alternatively, for the single-task based architectures, it implies to train another LSTM model from scratch. Thus, MTL favors scalability with respect to number of SON use cases that can be handled in parallel by the RAN. For instance, considering MLEC values of GL architectures, the sum of the used energy by GL-ST-HD and GL-ST-IMS is 1.21 KWh, whereas with MTL the two tasks would consume 1.15 KWh (i.e., 1.09 KWh + 5%). In case we added a third task with energy consumption equal to the average of the HD and IMS tasks, the single-task paradigm would require approximately 1.82 KWh, whereas MTL only 1.2 KWh. Thus, the single-task presents a linear increment in energy expenditure with respect to the number of tasks, whereas MTL increments of only 5%.

Finally, it is also worth mentioning that the amount of data involved in the communication has been dramatically reduced. In the centralized version, 40 GB have to be transmitted to the central cloud, whereas in the GL versions the total amount of data exchanged when passing the model parameters is: 7 MB for GL-ST-HD and GL-ST-IMS, 21 MB for the AE and 1.5 MB for the MLPs of GT-MT-HD and GT-MT-IMS. This translates in further energy savings, since less data have to be transmitted and less demanding storage hardware is needed at the edge with respect to a centralized cloud solution. According to [11], these energy savings in the communication and storage are of the 25%.

VI. OPEN ISSUES AND FUTURE DIRECTIONS

In this work, we presented a comparison between centralized and distributed architectures for implementing two RAN SON functionalities. The results highlight that the distributed solutions are viable and cost-effective. However, several issues are still open on this field and need a proper investigation, both theoretical and applied, in order to efficiently use ML models in a distributed manner. In the following, a few fundamental research problems are presented and discussed.

A. SEQUENTIAL TRAINING

As seen in Section V-A, the GL presents good performance in terms of prediction accuracy for the scenario considered. However, the variation of the error during the training phase suggests that the sequentially learning might suffer from the problem of non-iid data sources. In this case, an ANN experiences CF and tends to lose information from previously learned data as information relevant to the new data are incorporated. Continual Learning (CL) [55] is a branch of ML aiming at handling this type of situation. CL investigates the ability of ML models to learn consecutive tasks without forgetting how to perform previously trained tasks. Therefore, CL can apply not only to GL but also to KTL and MTL, when the learning phase of the different tasks is performed at different steps. The main idea of CL is remembering only essential concepts and identifying the potential source of interference in the data. In order to do this, CL implements different memorization approaches that incorporate new knowledge and protect them from modification, in detail:

- Dynamic Architecture: the ANNs create new weights automatically that will learn new tasks, whereas trained weights are frozen to protect memories.
- Rehearsal: this class of algorithms identifies a subset of training data as memory, which maintains knowledge from past learning experiences.
- Generative Replay: where the goal is to learn generative models for generating artificial samples as memory of past learning experiences.
- Regularization: the loss is defined to constrain weight updates in order to retain knowledge from previous tasks.

In the GL algorithm, CF can be addressed also through an early-stop solution, which allows to interrupt the GL training phase as soon as the desired level of error has been reached. This may also increase the energy efficiency of the system. To do so, the trade-off between the original GL algorithm, where a single SGD is performed at each round, and the evaluated solution, where the entire local database is processed at each round, should be evaluated in order to study the training behavior of the different solution as function of the accuracy, energy and communication performance.

The incremental paradigm of the sequential training phase can be used also for managing more complex scenarios, i.e., with higher number of BSs and/or with data distribution modifying in time (e.g., changes in scenario as new obstacles or deployment of new BSs). In fact, CL based solutions aim at finding working algorithms for agents which learn from an evolving environment and that need to learn continually to adapt to unseen situations and remember already learned solutions to known situations.

Finally, also the local databases require a more in-depth investigation of their iid properties so as to evaluate the GL training phase in terms of the energy and accuracy. This will help to discover noisy data and filter them accordingly for improving the final accuracy or implementing the rehearsal CL memorization solution.

B. DISTRIBUTED ML SOLUTIONS

As presented in Section II, several ML solutions are available for distributing the training phase: FL is attracting the attention, but KTL can be also an interesting field of research. On this matter, it is of paramount importance to investigate on the different performance of the distributed ML solutions in terms of accuracy, latency, energy and communication KPIs in order to be able to choose the proper solution for the different applications. FL can help in having a stricter control on the distributed training process thanks to its centralized management of the model, thus facilitating the optimization of accuracy and latency. However, the information exchanged between nodes and the central entity is higher with respect to GL and need more strict requirements in terms of latency and reliability.

On the other hand, KTL can improve the energy efficiency of the system by exploiting the transfer of the already acquired knowledge. This is not always possible, as in many cases it is impracticable to identify a source model since the data stored at the nodes are of equal importance, or the hierarchical distribution among the nodes is not known a-priori.

Finally, all distributed ML solutions that implement a sequential learning have to deal with the CF problem. Consequently, all the challenges presented in Section VI-A have to be properly evaluated for each solution.

C. DEEP REINFORCEMENT LEARNING SOLUTIONS

Reinforcement Learning (RL) based solutions also represent a viable approach to investigate, especially considering their extension through DL. In particular, Deep ANN can be used as approximation function to solve the RL problem when a huge number of variables has to be considered, as in our case. The main problem of DRL-based solutions is represented by the training phase, in which the algorithm needs to interact with the environment during its learning process. To do this, the off-line paradigm implements the training in a model of the environment, e.g., with simulation tools. However, the model of the environment has to be carefully designed for having a limited gap with respect to the real environment in order to be able to perform a valuable training. On the other hand, in the on-line training, the direct interactions with the environment can jeopardize the system operations and consequently, the exploration has to be carefully guided, such as with Upper Confidence Bound (UCB) [56] and Exponential weight algorithm for Exploration and Exploitation (EXP3) [57] solutions.

D. GREEN AI

The field of Green AI is very recent and still there is not a common agreement among the researchers on how to perform the energy assessment of the ML solutions, as presented in [8]. MLEC and GA represent two valuable tools, but provide energy figures which are hardware dependent, and, thus, they do not allow for a fair comparison between different models as well as to decouple the model contributions from hardware improvements. Another useful metric is represented by the Floating-Point Operations (FPO). FPO directly computes the amount of work done by a machine and is agnostic to the specific hardware, but it does not consider the implementation of the model and its memory consumption, which may often lead to additional energy and monetary costs [58].

The number of parameters of the model (i.e., the internal variables of the model whose values can be estimated from data) is an important metric. In fact, it is independent from the hardware and is highly correlated with the memory consumption. However, different algorithms make a different use of their parameters (e.g., deeper ANN and wider ANN), which implies that a similar number of parameters might correspond to a different amount of work. Finally, another aspect to be considered is the model sensitivity to hyperparameters, i.e., all the parameters related to the configuration that are external to the model and whose value cannot be estimated from data. An example of such analysis is the characterization of the model tuning time, which could reveal inconsistencies in time spent tuning baseline models compared to proposed contributions. This sensibility is especially important when proposing a model that has to be re-trained for its application, such as re-training on a new domain or fine-tuning on a new task.

VII. CONCLUSION

In this paper, we have presented distributed architectures for two RAN SON functionalities based on multi-task and gossip learning animated by the big data availability of BS statistics at the edge. We considered the handover decision and the initial MCS selection use cases. We evaluated the proposed solutions considering both their accuracy and consumed energy in realistic scenarios. Results prove that the proposed distributed implementations of the two RAN SON functionalities allow to increase the energy-efficiency of the system while maintaining the same network performance with respect to their correspondent centralized versions. Finally, we have discussed some open research issues that have been identified during this work and can be generalized to the emerging interesting field of edge intelligence.

REFERENCES

[1] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016 2021*, Cisco, San Jose, CA, USA, Feb. 2017. [Online]. Available: http://www.cisco.com/

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[3] A. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015.

[4] J. Moysen and L. Giupponi, "From 4G to 5G: Self-organized network management meets machine learning," *Comput. Commun.*, vol. 129, pp. 248–268, Sep. 2018.

[5] N. Baldo, L. Giupponi, and J. Mangues-Bafalluy, "Big data empowered self organized networks," in *Proc. 20th Eur. Wireless Conf.*, May 2014, pp. 1–8.

[6] S. Ali *et al.*, "6G white paper on machine learning in wireless communication networks," 2020, *arXiv:2004.13875*. [Online]. Available: http://arxiv.org/abs/2004.13875

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy, Jul. 2019, pp. 3645–3650. [Online]. Available: https://www.aclweb.org/anthology/P19-1355

[8] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," 2019, *arXiv:1907.10597*. [Online]. Available: http://arxiv.org/abs/1907.10597

[9] F. Woudstra. (2020). *Ethical Guidelines for Transparent Development and Implementation of AI—An Overview*. [Online]. Available: https://www.filosofieinactie.nl/blog/2020/4/9/ethical-guidelines-for-tr %ansparent-development-and-implementation-of-ai

[10] *Real-World Impact of Mobile Edge Computing*, Intel, Mountain View, CA, USA, 2016.

[11] E. Ahvar, A.-C. Orgerie, and A. Lebre, "Estimating energy consumption of cloud, fog and edge computing infrastructures," *IEEE Trans. Sustain. Comput.*, early access, Mar. 18, 2019, doi: 10.1109/TSUSC.2019.2905900.

[12] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The global footprint of mobile communications: The ecological and economic perspective," *IEEE Commun. Mag.*, vol. 49, no. 8, pp. 55–62, Aug. 2011.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[14] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," 2019, *arXiv:1909.00560*. [Online]. Available: http://arxiv.org/abs/1909.00560

[15] E. Peltonen, M. Bennis, M. Capobianco, M. Debbah, A. Ding, F. Gil-Castiñeira, M. Jurmu, T. Karvonen, M. Kelanti, A. Kliks, T. Leppänen, L. Lovén, T. Mikkonen, A. Rao, S. Samarakoon, K. Seppänen, P. Sroka, S. Tarkoma, and T. Yang, "6G white paper on edge intelligence," 2020, *arXiv:2004.14850*. [Online]. Available: http://arxiv.org/abs/2004.14850

[16] Y. Zhang and Q. Yang, "A survey on multi-task learning," 2017, *arXiv:1707.08114*. [Online]. Available: http://arxiv.org/abs/1707.08114

[17] R. Ormándi, I. Hegedås, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency Comput., Pract. Exp.*, vol. 25, no. 4, pp. 556–571, Feb. 2013.

[18] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," 2016, *arXiv:1611.09726*. [Online]. Available: http://arxiv.org/abs/1611.09726

[19] S. Parikh, D. Dave, R. Patel, and N. Doshi, "Security and privacy issues in cloud, fog and edge computing," *Procedia Comput. Sci.*, vol. 160, pp. 734–739, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050919317181

[20] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 1. Cambridge, MA, USA: MIT Press, 2015, pp. 1135–1143.

[21] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6071–6079.

[22] G. Li, L. Liu, X. Wang, X. Dong, P. Zhao, and X. Feng, "Auto-tuning neural network quantization framework for collaborative inference between the cloud and edge," 2018, *arXiv:1812.06426*. [Online]. Available: http://arxiv.org/abs/1812.06426

[23] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, "ReXCam: Resource-efficient, cross-camera video analytics at scale," 2018, *arXiv:1811.01268*. [Online]. Available: http://arxiv.org/abs/1811.01268

[24] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1040–1050.

[25] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, New York, NY, USA, Sep. 2015, pp. 1310–1321.
[26] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?" in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 42–49.
[27] F. D. Calabrese, L. Wang, E. Ghadimi, G. Peters, L. Hanzo, and P. Soldati, "Learning radio resource management in RANs: Framework, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 138–145, Sep. 2018.
[28] H. Liao, Z. Zhou, X. Zhao, L. Zhang, S. Mumtaz, A. Jolfaei, S. H. Ahmed, and A. K. Bashir, "Learning-based context-aware resource allocation for edge-computing-empowered industrial IoT," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4260–4277, May 2020.
[29] H. Liao, X. Chen, Z. Zhou, N. Liu, and B. Ai, "Licensed and unlicensed spectrum management for cognitive M2M: A context-aware learning approach," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 3, pp. 915–925, Sep. 2020.
[30] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities and challenges," 2019, *arXiv:1908.06847*. [Online]. Available: http://arxiv.org/abs/1908.06847
[31] 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.
[32] M. Chen, O. Semiari, W. Saad, X. Liu, and C. Yin, "Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 177–191, Jan. 2020.
[33] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.
[34] E. Bakopoulou, B. Tillman, and A. Markopoulou, "A federated learning approach for mobile packet classification," 2019, *arXiv:1907.13113*. [Online]. Available: http://arxiv.org/abs/1907.13113
[35] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient radio resource allocation for federated edge learning," in *Proc. IEEE Int. Conf. Commun. Workshops*, Jun. 2020, pp. 1–6.
[36] A. Rago, G. Piro, G. Boggia, and P. Dini, "Multi-task learning at the mobile edge: An effective way to combine traffic classification and prediction," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10362–10374, Sep. 2020.
[37] A. Shahmansoori, "Sparse Bayesian multi-task learning of time-varying massive MIMO channels with dynamic filtering," *IEEE Wireless Commun. Lett.*, vol. 9, no. 6, pp. 871–874, Jun. 2020.
[38] N. Ye, X. Li, H. Yu, L. Zhao, W. Liu, and X. Hou, "DeepNOMA: A unified framework for NOMA using deep multi-task learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2208–2225, Apr. 2020.
[39] *NG-RAN; Architecture description*, document TS 38.401 version 16.1.0, 3GPP, Mar. 2020.
[40] Open RAN Alliance, "O-RAN: Towards an open and smart RAN," White Paper, Oct. 2018. [Online]. Available: https://www.o-ran.org/s/O-RAN-WP-FInal-181017.pdf
[41] *Radio Measurement Collection for Minimization of Drive Tests (MDT); Overall Description*, document TS 36.331 version 10.4.0, 3GPP, 2010.
[42] S. S. Mwanje and A. Mitschele-Thiel, "Distributed cooperative Q-learning for mobility-sensitive handover optimization in LTE SON," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2014, pp. 1–6.
[43] Z. Ali, N. Baldo, J. Mangues-Bafalluy, and L. Giupponi, "Machine learning based handover management for improved QoE in LTE," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp.*, Istanbul, Turkey, Apr. 2016, pp. 794–798.
[44] L. Giaretta and S. Girdzijauskas, "Gossip learning: Off the beaten path," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2019, pp. 1117–1124.
[45] J. Yu, C. Hong, Y. Rui, and D. Tao, "Multitask autoencoder model for recovering human poses," *IEEE Trans. Ind. Electron.*, vol. 65, no. 6, pp. 5060–5068, Jun. 2018.
[46] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[47] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 06, no. 2, pp. 107–116, Apr. 1998.
[48] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented LTE network simulator based on NS-3," in *Proc. 14th ACM Int. Conf. Modeling, Anal. Simulation Wireless Mobile Syst.*, New York, NY, USA, 2011, pp. 293–298.
[49] Z. Ali, M. Miozzo, L. Giupponi, P. Dini, S. Denic, and S. Vassaki, "Recurrent neural networks for handover management in next-generation self-organized networks," in *Proc. PIMRC*, 2020, pp. 1–6.
[50] F. Chollet. (2015). *Keras*. [Online]. Available: https://keras.io
[51] R. French, "Catastrophic forgetting in connectionist networks," *Trends Cognit. Sci.*, vol. 3, no. 4, pp. 128–135, Apr. 1999.
[52] B. Bojovic, N. Baldo, J. Nin-Guerrero, and P. Dini, "A supervised learning approach to cognitive access point selection," in *Proc. IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2011, pp. 1100–1105.
[53] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," 2019, *arXiv:1910.09700*. [Online]. Available: http://arxiv.org/abs/1910.09700
[54] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: Quantifying the carbon footprint of computation," 2020, *arXiv:2007.07610*. [Online]. Available: http://arxiv.org/abs/2007.07610
[55] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. D. Rodríguez , "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Inf. Fusion*, vol. 58, pp. 52–68, Dec. 2020.
[56] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Nov. 2003.
[57] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Proc. IEEE 36th Annu. Found. Comput. Sci.*, 1995, pp. 322–331.
[58] M. N. Zhang, X. Zheng, and H. J. Sun, "Shufflenet v2: Practical guidelines for efficient CNN architecture design," in *Proc. 15th Eur. Conf. Comput. Vis., (ECCV)*, Sep. 2018, pp. 116–131.

MARCO MIOZZO received the M.Sc. degree in telecommunication engineering from the University of Ferrara, Italy, in 2005, and the Ph.D. degree from the Technical University of Catalonia (UPC), in 2018. After graduated, he worked as a Research Engineer in wireless networking with the Consorzio Ferrara Ricerche (CFR), Italy, where he collaborated with the Department of Information Engineering (DEI), University of Padova, Italy. In June 2008, he joined the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). In CTTC, he has been involved in the development of an LTE module for the network simulator 3 (ns-3) in the framework of the LENA project. He is currently collaborating with the EU founded H2020 SCAVENGE (MSCA ETN). He participated in several Research and Development projects, among them 5G-Crosshaul, Flex5Gware and SANSA, working on environmental sustainable mobile networks with energy harvesting capabilities through learning techniques. His main research interests include sustainable mobile networks, green wireless networking, energy harvesting, multi-agent systems, machine learning, Green AI, energy ethical consumerism, transparent, and explainable AI.

ZORAZE ALI received the M.Sc. degree in radio communication from the Blekinge Institute of Technology, Karlskrona, Sweden. He has worked as an Engineer with the Network and Operations Department, Telecard, Pakistan. Telecard is the country's first-ever payphone service provider and a pioneer for providing value-added services for mobile communication. In Telecard, he worked on LUCENT and ZTE technology based equipment offering CDMA cellular network communications, particularly CDMA2000 1X based WLL Network. In September 2014, he joined the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), as a Research Assistant to pursue his Ph.D. degree in self organizing networks (SON), big data and machine learning with particular focus on LTE/LTE-A and 5G networks with the Telematics Engineering Department, Universitat Politècnica de Catalunya (UPC), Barcelona.

LORENZA GIUPPONI (Senior Member, IEEE) received the Ph.D. degree from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 2007. In 2003, she joined the Radio Communications Group, UPC, with a grant of the Spanish Ministry of Education. From 2006 to 2007, she was an Assistant Professor with UPC. In September 2007, she joined the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), where she is currently a Senior Researcher with the Mobile Networks Department, Communication Networks Division. Since 2007, she has also been a member of the Executive Committee of CTTC, where she acts as the Director of Institutional Relations. She was a co-recipient of the IEEE CCNC 2010, IEEE 3rd International Workshop on Indoor and Outdoor Femto Cells 2011, and the IEEE WCNC 2018 Best Paper Awards. Since 2015, she has been a member of the Executive Committee of ns-3 Consortium.

PAOLO DINI received the M.Sc. and Ph.D. degrees from the Università di Roma La Sapienza, in 2001 and 2005, respectively. He served as a Postdoctoral Researcher with the Research Centre on Software Technology (RCOST), Università del Sannio, and a contracted Professor with the Università di Roma La Sapienza, in 2005. He is currently a Senior Researcher with the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). He received two awards from the Cisco Silicon Valley Foundation for his research on heterogeneous mobile networks, in 2008 and 2011, respectively. He has been involved in over 25 research projects related to network management, optimization, and energy efficiency. He is currently a coordinator of the EU H2020 MSCA SCAVENGE European Training Network on sustainable mobile networks with energy harvesting capabilities. His research interests include sustainable networking and computing, distributed optimization and optimal control, machine learning, and data analytics.