Multi-Task Learning for Efficient Management of Beyond 5G Radio Access Network Architectures

ZorazeAli, Lorenza Giupponi, Marco Miozzo, PaoloDini

***Abstract*—Automation of Radio Access Network (RAN) operation is a fundamental feature to manage sustainable and efficient Beyond Fifth-generation wireless (5G) networks, in the context of the Next Generation Self-Organizing Network (NG-SON) vision. Machine Learning (ML) is already identified as the key ingredient of this vision, with new standardized and open architectures, like Open-RAN (O-RAN), taking momentum. In this paper, we propose models based on single-task and Multi-Task Learning (MTL) paradigms to address two RAN use cases, handover management and initial Modulation and Coding Scheme (MCS) selection. Traditional handover schemes have the drawback of taking into account the quality of the signals from the serving, and the target cell, before the handover. Also, initial MCS at the start of the session and after a handover usually is handled conservatively. The proposed ML solutions allow to address these drawbacks by 1) considering the expected Quality of Experience (QoE) resulting from the decision of a target cell to handover, as the driving principle of the handover decision and 2) using the experience extracted from network data to make smarter initial MCS allocations. In this line, we implement a realistic cellular simulation scenario by incorporating coverage holes to build an extensive database to train and test the proposed models. The results show that the ML-based models outperform the 3rd Generation Partnership Project (3GPP) standardized handover and initial MCS selection approaches by improving the QoE of users resulting from a handover and the throughput obtained upon establishing a new connection with a network. Besides that, using the obtained results, this paper extensively discusses the merits of leveraging the MTL model to address different, but related multiple RAN functions because it allows reusing a common learning architecture for multiple RAN use cases, which provides significant implementation advantages.**

***Index Terms*—LSTM, RNN, Next Generation Self-organizing networks, Deep learning, machine learning, Mobile Networks**

All authors are with the Mobile Networking Department at Centre Tecnológic de Telecomunicacions de Catalunya (CTTC/CERCA), Barcelona, Spain.
E-mails by the authorship order: zali@cttc.es, lgiupponi@cttc.es, mmiozzo@cttc.es, and pdini@cttc.ec.

This work was supported by Huawei Technologies, Sweden AB. The authors would like to specifically thank Stojan Denic, Stavroula Vassaki and Gunnar Peters, from Huawei Technologies, for the enriching discussions on the work.

ACRONYMS

3GPP	3rd Generation Partnership Project
5G	Fifth-generation wireless
6G	Sixth-generation wireless
AE	AutoEncoders
AI	Artificial Intelligence
ARPU	Average Revenue per User
CAPEX	Capital Expenditures
CQI	Channel Quality Indicator
CW	CodeWord
DB	DataBase
DL	Downlink
DNN	Deep Neural Network
ECDF	Empirical Cumulative Distribution Function
gNB	next-Generation Node B
HO	HandOver
KQI	Key Quality Indicator
LSTM	Long Short Term Memory
LTE	Long Term Evolution
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
MDT	Minimization of Drive Test
ML	Machine Learning
MLP	Multi-Layer Perceptron
mmWave	Millimeter Wave
MNO	Mobile Network Operator
MOS	Mean Opinion Score
MSE	Mean Square Error
MTL	Multi-Task Learning
NG-SON	Next Generation Self-Organizing Network
NOMA	Non-Orthogonal Multiple Access
O-RAN	Open-RAN
OPEX	Operational Expenditures
POMDP	Partially Observable Markov Decision Process
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RIC	Radio Intelligent Controller
RLF	Radio Link Failure
RNN	Recurrent Neural Network
RRC	Radio Resource Control
RSRP	Reference Signal Receive Power
RSRQ	Reference Signal Received Quality
SINR	Signal to Interference plus Noise Ratio
SLA	Service Level Agreement
SNR	Signal to Noise Ratio
SON	Self-Organizing Network
TCP	Transmission Control Protocol
TTI	Transmission Time Interval

UE User Equipment
UL Uplink
V2X Vehicular-to-everything

I. Introduction

Mobile communications have experienced during the last decades an incredible evolution. Since its inception, mobile device connections have surpassed the number of people in the world, making it the fastest growing technology ever [1]. Despite that, it is well known that the revenue generated by Mobile Network Operators (MNOs) per user (Average Revenue per User (ARPU)) has been steadily decreasing for the last decade. The Capital Expenditures (CAPEX) of Fifth-generation wireless (5G) networks are still not completely clear and include: 1) more spectrum, with expensive auction fees, 2) deployment of new antennas and equipment upgrade, 3) large scale small-cell deployments, to pursue the Millimeter Wave (mmWave) vision. Therefore, the reduction of Operational Expenditures (OPEX) is fundamental to the evolution of Beyond 5G mobile communication systems. Another interesting data is that 70 % of the total cost involving deployment, optimization, and operation of a network comes from the Radio Access Network (RAN) segment [2]. It is the reason that there is a significant interest in improving the efficiency of the RAN management, which consequently reduces its OPEX. Currently, there are two main trends to achieve these objectives.

On the one hand, automation and self-organization of the RAN have become two fundamental ingredients for optimal resource utilization and management. It has been almost a decade since when Self-Organizing Network (SON) was defined and introduced as a feature of Long Term Evolution (LTE), in 3rd Generation Partnership Project (3GPP) Release 8 [3]. Since then, it has been evolving through the releases and into the concept of Next Generation Self-Organizing Network (NG-SON) for 5G networks [4]. 5G cellular networks and beyond are characterized by highly complex, dense, and heterogeneous deployments to increase network coverage and capacity. Besides traditional sub-6 GHz and licensed bands, the access can span over a wide range of bandwidth, including mmWave and unlicensed spectrum. The high diversity of mobile devices and new applications further complicates the network architecture and its management. In this context, mobile networks generate a massive amount of measurements, control, and management information during their normal operation [5] [6], which can be efficiently used to address the 5G and beyond network management challenges.

On the other hand, the evolution towards Beyond 5G and Sixth-generation wireless (6G) networks calls for further architectural transformations required to support service heterogeneity, coordination of multi-connectivity, on-demand service deployment. In Feb. 2018, the Open-RAN (O-RAN) Alliance was founded by a group of mobile network operators to enhance RAN performance through virtualized network elements, openness, and intelligence. Openness aims to eliminate proprietary hardware and software implementations by establishing open standard interfaces, which will reduce operating costs. Intelligence has already become a necessity for the deployment, optimization, and operation of Beyond 5G networks. O-RAN introduces new Radio Intelligent Controller (RIC) modules and enables them with Machine Learning (ML)/Artificial Intelligence (AI) features to enhance traditional network functions with intelligence. In one of its defining white papers [7], different use cases are proposed, like traffic steering, Quality of Experience (QoE) optimization, Quality of Service (QoS) based resource optimization, RAN Slice Service Level Agreement (SLA) assurance, context based dynamic HandOver (HO) management for Vehicular-to-everything (V2X).

Both these trends and visions converge to the already widely agreed need of ML/AI as fundamental ingredients of Beyond 5G and 6G networks. Specifically, deep learning has been lately intensively applied in communication and networking literature to solve a wide range of problems [8]. RAN operation is extremely complex. It requires the ability to continuously adapt to the environment's ever-changing conditions in terms of propagation, users' needs, system load, high mobility, etc. The number of tasks that a Beyond 5G RAN has to execute is vast and includes all traditional SON use cases and more to come. These functions are built around adjusting operation parameters, which often affect the operation of other SON functions, leading to the need for the so-called self-coordination of multiple SON tasks.

In ML, we typically care about optimizing for a particular metric. To do this, we generally train a model to perform our desired task, or in our case, the RAN function. We then fine-tune these models until their performance no longer increases. By doing so, independently for all the different RAN use cases, we may be ignoring information that might help us do even better on the metric of interest. Specifically, this useful information may come from different and related tasks/RAN functions. By sharing representations between these tasks, for example, using a wide feature space, which is not reduced only to a specific problem to solve, we can enable our model to generalize better on our original task. This approach is called Multi-Task Learning (MTL) and has been used successfully across all applications of ML [9]. MTL is inspired by human learning, as integrating knowledge across tasks is an essential feature of human intelligence. In that, we believe that MTL reflects more accurately the human learning process than single-task learning. In this paper, our objective is to prove the potentiality of MTL to address RAN automation of multiple tasks, which have to function in parallel during the regular operation of the RAN. We propose different deep architectures to address a set of tasks through individual or shared models. Among them, we study the effectiveness of AutoEncoders (AE) [10] to reuse the compressed representation of the data for multiple heterogeneous use cases [11] [12]. This approach can significantly reduce the implementation and computational complexity of the learning architectures. To prove this concept, we propose to target, without loss of generality, two RAN use cases: 1) HO management and 2) the selection of the optimal initial Modulation and Coding Scheme (MCS).

We address both the use cases, first through *single-task individual* and then through *multi-task shared* models. To

address the use cases individually, we use a Long Short Term Memory (LSTM) Recurrent Neural Network (RNN) to take advantage of the temporal characteristic of the data extracted from several and extensive simulation campaigns using the latest 3GPP models of *ns-3* [13]. The LSTM is designed to solve a regression problem to estimate the QoE of the users. We obtain excellent prediction errors, and with these results, we can prove that the learning approach outperforms traditional HO solutions and conservative approaches to select the initial MCS. Successively, we build a different architecture where the two RAN tasks share and train a common AE, based on MTL principles. The same compressed data output of the AE is used as input to two different Multi-Layer Perceptrons (MLPs), implementing the regression of the particular parameter that we want to estimate for the two use cases (i.e., the HO management and initial MCS). Both MLPs offer excellent regression results similar to the one obtained using LSTM. It means, the AE successfully reduces the dimensionality of the data without losing meaningful information and network performance. Thus, it facilitates the sharing of knowledge between different tasks. Consequently, the same architecture can be used to implement multiple parallel RAN functions.

In addition to that, we go more deeply into the study by comparing two different ways of learning. In the first case, the *parallel MTL* case, we learn the shared model by building a shared database for all the use cases we plan to address. This type of database is viable when we know the needed use cases beforehand. However, RAN management problems can be more complex and continuously require adding new use cases and on-demand tasks to the design without retraining previous tasks from scratch, or compromising their performance. As a result, there is a need for the MTL shared model to be flexible and be able to gradually add more tasks to its knowledge without forgetting previously known tasks. For that, we also propose a second *incremental MTL* scheme, based on the continual learning paradigm [14], where the training database is not built beforehand, but a new task can be incorporated separately while previous task knowledge is preserved. This approach is much more flexible and adequate for real networks and provides clear implementation advantages [15].

A preliminary and partial version of this work was presented in [16], and [17]. In [16], we focused only on the single task HO management use case, and used an LSTM RNN to solve the regression problem and estimate the QoE of the users. The obtained results proved that the learning approach outperforms traditional HO solutions. This paper further extends our research horizon and includes the second use case of the initial MCS. Moreover, in our previous work [17], we focused on studying the advantages and disadvantages of a decentralized or a centralized ML solution, emphasising the energy aspect. Differently, in this work, we build another architecture where the two RAN tasks share and train a common AE, based on the MTL principles. Finally, the contributions of this paper are the following:

- Design of ML models based on single-task and multi-task paradigms to address two RAN use cases. In particular, we propose two models based on two different multi-task techniques, i.e., parallel and incremental learning.
- To encourage the reproducibility of the proposed models and results, we provide in-depth details of the simulation scenario and steps to create the databases using an open-source simulator *ns-3*.
- Performance evaluation of the proposed solutions by comparing the results with 3GPP standardized HO and initial MCS selection schemes.

Last but not least, building upon the above contributions, we are advancing the state-of-the-art and related work in the following aspects:

- We present a holistic solution to handover based on download time that is not limited to adjusting typical HO parameters but considers previous experience to select a target next-Generation Node B (gNB) to improve users' QoE.
- Extending our previous work in [16], we introduce an additional RAN use case, i.e., initial MCS selection. Compared to the studies in the literature that lean towards treating these two use cases separately, in this paper, we study the solutions that allow learning concurrently these RAN tasks based on the MTL learning paradigm.
- We study two possible solutions for MTL in the context of solving RAN problems, one based on parallel learning and another based on incremental MTL, which increases the learning pace and minimizes the delay to deploy RAN solutions.
- We study the performance based on the proposed solutions in comparison to 3GPP standard aligned baselines.

The rest of the paper is organized as follows. In Section II we discuss the related work. Section III introduces the system overview and the scenario that we use for synthetic data generation. Section IV, presents the procedure we adopted to generate the synthetic data. Section V proposes the RNN models for single-task and multi-task learning. Section VI discusses the training of the proposed architectures and the system level performance results. Finally, Section VII concludes the paper.

II. Related Work

Recent surveys on the application of ML learning in mobile networks [18] and their self-organization [19] show that the ML based solutions would play an essential role in the management of 5G and beyond networks. In the context of HO management, we identify three high-level potential ways to optimize its functionality. The first approach to optimize the HO process is to use model-based solutions based on Markov Decision Process (MDP). The objective, in this case, is to find the probability distribution of taking optimal HO decisions given the input state, which corresponds to a User Equipment (UE) state before the HO. In [20], the authors proposed a Viterbi algorithm to find an optimal HO policy to maximize the UE average capacity. The algorithm works under the assumption that the position of the eNBs, the UE trajectory, and the channel characteristics are known a priori. The work in [21] proposes a cell selection procedure based on the Partially Observable Markov Decision Process (POMDP). Specifically, the POMDP predicts the neighboring

cells' loading information to optimize the HO rate while maintaining the system throughput. Authors in [22], similarly to [20], proposed a context-aware HO policy, which optimizes the Time-to-Trigger parameter for HO by assuming the knowledge about the UE trajectory. These proposed solutions are based on the assumptions of having strong knowledge about network dynamics, which in turn are hard to capture in real networks. Therefore, model-free solutions which optimize the HO process without this previous and complete information are worth investigating.

The second approach considers then model-free solutions for HO parameter tuning. The idea is to adaptively fine tune the HO parameters defined in the standard to identify the strongest target cell, e.g., Hysteresis, Time-to-Trigger, HO Margin, and Cell individual Offset, by employing ML algorithms. In [23], a Q-learning approach is proposed that aims to optimize the HO parameters. In particular, the model finds the optimal values of Hysteresis and Time-to-Trigger parameters to reduce the radio link failure and ping pong effects. In [24], a method to adaptively select a Hysteresis value to reduce the number of unnecessary HO is proposed. Specifically, it uses a predefined threshold value of Reference Signal Received Quality (RSRQ) to adapt the Hysteresis value as per the UE measurements. Authors in [25], proposed a fuzzy logic controller, which finds an optimal value of the HO Margin parameter to reduce the signaling cost caused by HO.

These approaches that aim to select the strongest cell, based on the optimal tuning of HO parameters, have the shortcoming of considering the strongest signal for target cell selection before the HO. Furthermore, these schemes do not consider a long-term vision of performance indicators in the decision, in terms of, e.g., the perceived QoE, after the HO. For example, in urban scenarios where the HO to the strongest neighbour cell is successful, but shortly after the transmission is deeply affected by the presence of an outage, these HO approaches could fail to provide a satisfactory solution. Thus, they are likely to severely degrade QoE performance, due to the unpredicted cell outage [26].

As a result, the third approach to HO management, which is also the one considered in this paper and our previous work [16], is a data-driven approach. It aims at using experience extracted from network data to include the vision of long-term optimization in the HO management decision. In [27], the authors proposed a hybrid HO controller based on deep reinforcement learning to minimize the HO rate while maintaining a certain level of system throughput. In particular, the work uses a Deep Neural Network (DNN), composed of LSTM units, which are trained following the supervised learning approach to predict the probability of selecting a target cell. It uses a dataset consisting of RSRQ measurements by simulating a standard compliant HO algorithm before executing the reinforcement learning approach. Similarly, in [28], a DNN is trained to solve the multiclass classification problem. In particular, it uses the Reference Signal Receive Power (RSRP) measurements reported by the UEs to their serving gNB. Then, using the softmax activation function for the output layer of the trained model, it computes the probability for a neighbouring gNB to become the next serving gNB. The smart HO approach to select the next serving gNB presented in our paper is different from what is proposed in [27], [28]. Specifically, it uses the regression to predict the perceived QoE (i.e., file download time) for each potential target gNB, and it HOs to the one, which could provide a better QoE. Moreover, the inputs to our model, thanks to deep ML architecture, include not only RSRP and RSRQ, but also many other measurements from the whole protocol stack, as it will be discussed in Section IV.

On the other hand, to the best of our knowledge, related work for the initial MCS selection is limited to the usage of reinforcement learning. For example, [29] proposed a solution, that learns the best MCS given the Signal to Noise Ratio (SNR) at a specific channel state. Differently from [29], the authors in [30] present a solution that uses Channel Quality Indicator (CQI) as a metric for the state representation. In this case, the authors argue that fine discretization of SNR with discrete MDP leads to a higher state space, which increases the convergence and exploration time. The authors in [31] proposed a deep reinforcement learning approach to overcome the issue of a large state space highlighted by [30]. They used SNR, Signal to Interference plus Noise Ratio (SINR), the previous action, and its immediate reward for the state representation.

In this paper, differently from the above presented solutions in [23]–[25] and [20]–[22], [27]–[31], we aim to demonstrate that different RAN functions, such as HO and initial MCS selection, can be considered as related tasks. Therefore, these tasks can be jointly trained through shared models for each task can benefit from other auxiliary tasks. Such an approach offers multiple implementations and learning advantages like reduced training effort, improved data efficiency, reduced overfitting through shared representations, and fast learning by leveraging auxiliary information. MTL has already been recently considered in mobile communications literature. In [32], multi-task Sparse Bayesian Learning (SBL) is applied for learning time-varying sparse channels in the uplink for multi-user massive MIMO systems. Results show that it is possible to considerably reduce the complexity and the required time for the convergence with a negligible sacrifice of the estimation accuracy. In [33], MTL is used to train a shared model for both traffic classification and prediction at the edge of the network. Classification accuracy and prediction error benefit from the shared model and return better performance with respect to single-task neural network architectures. In [34] multi-task DNN framework for Non-Orthogonal Multiple Access (NOMA), namely DeepNOMA, has been proposed to treat non-orthogonal transmissions as multiple distinctive but correlated tasks. To the best of the authors' knowledge, this work is the first one in literature deeply discussing and proving the concept of MTL for the efficient automation of the RAN in future mobile networks, improving so the ability to make connections between facts, observations, patterns, and other tasks from which they learn.

III. System Overview

This section first describes the implemented RAN functions using deep learning solutions. Then, we introduce the target

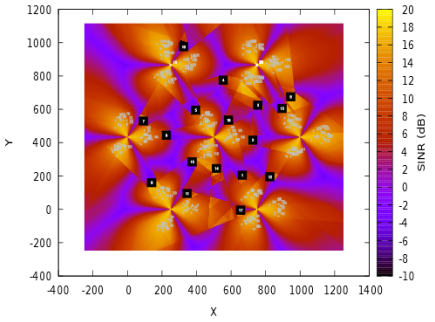


Fig. 1: Simulation scenario with coverage holes (black squares).

simulation scenario, which is depicted in Fig. 1 [16].

A. Target RAN functions and use cases

Keeping the high-level objectives in mind, we have selected two RAN functions to be addressed using a supervised learning approach: 1) HO management, 2) initial MCS selection.

- *HO management*: We propose a HO management approach, which allows HO to the cell suggested by a supervised learning algorithm capable of predicting a QoE indicator through a regression procedure. The supervised learning algorithm exploits the experience extracted by data already available in the network (e.g., the Minimization Drive Test database [35]). Based on this, it detects a most appropriate cell to HO, as a function of the future expected QoE perceived by the user, instead of the RSRP or the RSRQ as the standard suggests. We model the problem as a regression problem, where we aim to estimate the necessary time to download a file transmitted over a Transmission Control Protocol (TCP) transport, while the users move around in a realistic multi-cell scenario challenged by deep outage zones. Related to the selection of time to download as a QoE metric, it is motivated by the fact that it is one of the standardized Key Quality Indicator (KQI) for the file transfer service in mobile networks [36] [37]. In the literature, some models are based on subjective Mean Opinion Score (MOS) to derive the QoE; however, all of them depend on end-to-end throughput perceived by the users [38]. On the other hand, following a similar methodology taken in [39], we take a more generalized approach, i.e., instead of using a specific MOS model for file transfer service, time to download has been used as an indicator of the QoE perceived by the users. Finally, it is to be noted that this solution has to be considered a component of a more sophisticated HO algorithm that also includes other aspects, e.g., load balancing, QoS requirement of a UE, etc. However, we think that including these additional components is out of the scope of this paper.

TABLE I: Simulation network parameters.

Parameter	Value
System bandwidth	5 MHz
Inter-site distance	500 m
Handover algorithm	A2-RSRP
Adaptive Modulation & Coding Scheme	Vienna [13]
SINR computation for DL CQI	Control method [13]
gNBs antenna type	Parabolic
gNBs antenna Beamwidth	70 degrees
gNBs antenna max attenuation	20 dB
Number of macro gNBs	21 (7 cells)
gNBs Tx Power	46 dBm
Numerology	0
Distance between the center points of the UEs cluster and the cell	100 m
UEs Cluster diameter	50 m
Number of UEs in the system	210 (30 per sector)
Mobility model	RandomWalk2dMobilityModel Mode: Time, Speed: 10 m/s Time: 40 sec, Distance: 4000 m
Path loss model	Cost231
gNB Antenna height	30 m
Obstacle height	35 m
Traffic	TCP Bulk File Transfer
File size	1.5 MB
Simulation time	40 sec

- *Intial MCS selection*: A standard approach in cellular networks, is that when the UE first switches to the CONNECTED Radio Resource Control (RRC) state, the initial selected MCS follows a conservative approach that guarantees that initial transmissions go through. As a result, usually, the lower MCS is selected (i.e., 0). We propose using knowledge from data reported by the users to choose an initial MCS in an optimal and non-conservative way to avoid wasting radio resources in the initial data exchange.

The selection of the use cases is done without loss of generality and is instrumental to the high level purpose of the work.

B. Simulation Scenario

We implement a realistic simulation scenario through *ns-3* LENA LTE - EPC (Evolved Packet Core) simulator [40]. A macro cell outdoor scenario has been considered with a network consisting of three-sectorial gNBs. A cluster of UEs is placed in each sector at a fixed distance from the center of a cell, in which the UEs are dropped at random positions. Since, in this scenario, we use TCP as the transport protocol, such deployment of the UEs guarantees to establish a TCP connection between the remote host and the UEs. The UEs start moving after receiving the first packet, following a mobility pattern resulting by tuning the parameters of the *RandomWalk2dMobilityModel* in *ns-3*. In particular, for every simulation run, a UE picks a random starting position in the cluster and a random angle in the range of [0° to 360°] to move away from the source gNB following a straight line. To increase the communication challenges in the scenario and to generate more random coverage patterns, we introduce obstacles in the scenario, which create multiple coverage holes,

TABLE II: List of input and output features used to create the training and testing dataset.

Input feature			
Layer	Measurements		
APP	1. Throughput UL 4. Throughput DL	2. Avg. number of rcvd packets UL 5. Avg. number of rcvd packets DL	3. Avg. number of rcvd bytes UL 6. Avg. number of rcvd bytes DL
RRC	7. Cell ID of serving cell 10. Cell ID of neighbour 1 . . 31. Cell ID of neighbour 8 34. Total number of radio link failures	8. RSRP from serving cell 11. RSRP from neighbour 1 . . 32. RSRP from neighbour 8 35. Total number of handovers	9. RSRQ from serving cell 12. RSRQ from neighbour 1 . . 33. RSRQ from neighbour 8 36. First target cell ID to handover
PDCP	37. Total number of txed PDCP PDUs DL 40. Avg. PDCP PDU delay DL 43. Min. PDCP PDU size DL 46. Total number of rcvd PDCP PDUs UL 49. Min. value of the PDCP PDU delay UL 52. Max. PDCP PDU size UL	38. Total number of rcvd PDCP PDUs DL 41. Min. value of the PDCP PDU delay DL 44. Max. PDCP PDU size DL 47. Total bytes txed UL 50. Max. value of the PDCP PDU delay UL	39. Total bytes txed DL 42. Max. value of the PDCP PDU delay DL 45. Total number of txed PDCP PDUs UL 48. Avg. PDCP PDU delay UL 51. Min. PDCP PDU size UL
RLC	53. Total number of txed RLC PDUs DL 56. Total number of bytes rcvd DL 59. Max. value of the RLC PDU delay DL 62. Total number of txed RLC PDUs UL 65. Total bytes rcvd RLC PDUs UL 68. Max. value of the RLC PDU delay UL	54. Total number of rcvd RLC PDUs DL 57. Avg. RLC PDU delay DL 60. Min. RLC PDU size DL 63. Total number of rcvd RLC PDUs UL 66. Avg. RLC PDU delay UL 69. Minimum RLC PDU size UL	55. Total number of bytes txed DL 58. Min. value of the RLC PDU delay DL 61. Max. RLC PDU size DL 64. Total bytes txed RLC PDUs UL 67. Min. value of the RLC PDU delay UL 70. Maximum RLC PDU size UL
MAC	71. Initial MCS 74. Avg. MCS UL 77. Avg. RB occupied DL 80. UL CQI	72 Avg. TB size UL 75. Avg. MCS DL 78. DL CQI inband	73. Avg. TB size DL 76. Avg. RB occupied UL 79. DL CQI wideband
PHY	81. Avg. SINR DL 84. Avg. number of UL HARQ NACKs	82. AVG. SINR UL	83. Avg. number of DL HARQ NACKs
Output feature			
APP	1. File download time [sec]	2. Initial DL throughput over 100 msec when a new RRC connection is established after the second handover	

as shown in Fig. 1 [16]. Each UE performs a TCP file transfer to a remote host in Downlink (DL) and Uplink (UL) direction. The complete set of simulation parameters are described in Table I [16]. The above simulation scenario is then used to conduct three extensive simulation campaigns, two for the single-task approaches, i.e., the HO management and the initial MCS, and one for the multi-task approach jointly targeting both the use cases. Each of them is repeated a specific number of times, which depend on the values of the parameters, i.e., the number of independent simulation runs, the maximum number of neighbours to HO, and the number of initial MCS values evaluated. The data obtained from these campaigns for each UE are stored in the form of a dataset, according to the format described in the next section (Section IV). We will explain in detail the use of this simulation scenario to build the databases for single and multi-task learning, targeting the two use cases.

IV. Data Generation

This section describes the characteristics of the collected dataset that we use as input to our proposed deep learning solutions. We constructed this dataset by conducting extensive simulation campaigns in the scenario presented in Subsection III-B. As mentioned in Section I, we model the HO management and initial MCS problems as regression problems, where we need to estimate, respectively, the QoE expected from performing HO to a certain target cell, and the initial throughput obtained by the UEs over a certain window. In general, when working with supervised learning, such as in our case, one has to build a DataBase (DB) with enough data to train, test, and evaluate the model. This dataset consists of input and output features stored in rows and columns. For this purpose, we have identified features at the multiple layers of the simulator protocol stack. These features can bring information to address not only the targeted RAN function, but also other RAN use cases that could be later considered. In particular, we have organized these features per layer of the 3GPP protocol stack, and presented them in Table II [16]. 3GPP already contemplates uploading a part of these measurements, e.g., UE measurements, under the Minimization of Drive Test (MDT) functionality [35]. All of these measurements are gathered in the simulator, by leveraging the *ns-3* "tracing system", which enables us to write them in text files as an output of the simulation program. Successively, we run multiple independent runs of the simulation scenario and then post-process all the generated text files to build a unique DB in *csv* format. The rest of this section describes the procedures to construct the DB for training and testing.

A. Procedure to build the database

For the purpose of evaluating and comparing single-task versus multi-task learning performances, we build four databases, two for the single-task approaches, i.e., targeting the two use cases individually, and two for the multi-task approach, considering the parallel and incremental MTL possibilities. In the following, we explain the pseudocode procedure to generate these databases.

1) Single-task HO management database (DB1): In a real-world scenario, a UE served by a gNB could HO to different

potential neighbor gNBs. It depends on the HO criterion. Examples of such criteria are the signal strength before the HO, reported using UE measurements, as traditionally proposed in standards, or the QoE after the HO, as proposed in this paper. This decision is usually affected by the UE's mobility pattern. However, it may also happen that different mobility patterns lead to the selection of the same target neighbour, because it is in all the cases identified as the most suitable neighbour to HO to. A QoE oriented HO algorithm must take these aspects into account. Therefore, the simulation campaigns to build the first DB (**DB1**) consists of several deterministic HOs to learn the QoE, i.e., a file download time for each UE, for the possible mobility patterns. The procedure to generate **DB1** is illustrated with the help of Pseudocode 1. Specifically, to consider both the aspects discussed above, the number of deterministic HOs to be performed by a UE of a gNB would depend on the number of independent runs used to generate different mobility patterns of this UE for each HO (first "for" loop of Pseudocode 1), and on the maximum number of neighbours this UE manages to see (last "for" loop of Pseudocode 1). Then, these deterministic HOs have to be simulated for every gNB (second "for" loop of Pseudocode 1) and every UE attached to a gNB (third "for" loop of Pseudocode 1) in our simulation scenario. The measurements resulting from these HOs will assist the proposed architecture in learning the most reasonable neighbour to HO. For this DB, we collect the data focusing only on the HO management use case.

It is also worth mentioning that we engineered this deterministic HO procedure to collect a synthetic DB in a reasonable time. However, in a real network, it would be possible to collect real online measurements based on the realistic mobility of the UEs during their lifetime while the network is normally operating.

2) Single-task initial MCS database (DB2): The second DB (**DB2**), targets the initial MCS (i.e., DL) use case. The logic to construct **DB2** is somewhat similar to **DB1**. The steps to generate this DB are presented in Pseudocode 2. In particular, in this case, we have to evaluate all, or a set of initial MCS values a gNB could use for a newly connected UE. Moreover, since the MCS depends on the SINR, which depends on the mobility of a UE, this DB should also consider this aspect. Therefore, a simulation for each MCS value (second "for" loop of Pseudocode 2) should be repeated for a number of independent runs (first "for" loop of Pseudocode 2) to record the QoE resulting from different mobility patterns of a UE. Similar to **DB1**, this has to be simulated for all the gNBs and their UEs in our simulation scenario (see, third and fourth "for" loop of Pseudocode 2). For this use case, the selected QoE indicator is the throughput achieved over a certain time window after a successful RRC connection establishment. This window's duration should be smaller than the configured DL CQI reporting interval of a UE, which is typically 200 ms, after which a gNB adapts the MCS based on the reported CQI.

3) Parallel MTL database (DB3): The procedure to generate the third DB (**DB3**) is the combination of the HO use case (see Pseudocode to generate **DB1**), which is then extended to repeat for all the potential initial MCSs for each gNB. This DB is generated using Pseudocode 3.

TABLE III: Database parameters.

Parameter	Value
Total number of input features	84
Maximum neighbours	8
UE measurement periodicity	200 ms
MCS values considered	0 (QPSK), 14 (16 QAM), 28 (64 QAM)
Total simulation runs	HO use case : 20 Initial MCS use case : 63 Multi-Task use case : 8

4) Incremental MTL database (DB4): For the evaluation of MTL, we also consider a fourth alternative DB (**DB4**). This DB is built incrementally based on the previous availability of **DB1** and **DB2**. In particular, it starts from **DB1**, and it incrementally adds data from **DB2**. With this DB we aim to evaluate the capability of the proposed architecture to incrementally learn a new task, once it has already been trained for other tasks. It would allow scalability in the RAN management, since new RAN tasks could be incrementally added to the architecture without additional implementation costs. All these databases could be intuitively expressed in the form of a dataset, as detailed in the next subsection.

B. Resulting database

A DB generated using any of the aforementioned pseudocode can be expressed as a matrix $\overline{\mathbf{X}}$.

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

where the feature vector of size 84 (i.e., the total number of input feature considered in this paper) is $\overline{x}_{i,j} \in \overline{\mathbf{X}}$, $1 \leq i \leq n$, and $1 \leq j \leq m$.

The parameter m defines the duration of the time series to be analyzed (i.e., the number of samples in the total simulation time, sampled with UE measurement periodicity), which corresponds to the number of time steps that the LSTM processes to perform the prediction. This number of time steps is the same for all the databases since we used the same periodicity to collect the measurements. On the other hand, the upper limit of n is different for all the 4 databases, and can be computed by multiplying the total number of UEs with the maximum neighbor BSs and/or the initial MCS to explore, and the total number of simulation runs. During the simulations, it may happen that some of the data are not available or is not valid. For example, in the simulations used to build **DB1** UEs might experience a Radio Link Failure (RLF) when forced to HO to a BS with poor channel conditions. When this happens, we do not have data since the user is not connected. On the other hand, for **DB2** it might happen that the initial throughput is not available due to the fact that the download is concluded before the measurement could be taken (see Section VI-A1 for more details). In these cases, after removing the affected entries from the databases for the overall simulation scenario,

Pseudocode 1: Pseudocode to generate **DB1**

initialization
$numGnbs \leftarrow 21$; $numUesPerGnb \leftarrow 10$
$numNeighboursPerUe \leftarrow 8$; $numRuns \leftarrow 20$
for $r \leftarrow 1$ **to** $numRuns$ **do**
  for $e \leftarrow 1$ **to** $numGnbs$ **do**
    for $u \leftarrow 1$ **to** $numUesPerGnb$ **do**
      for $n \leftarrow 1$ **to** $numNeighboursPerUe$ **do**
        Start moving away from the serving gNB, based on a random direction.
        HO to neighbour cell n as per event A2.
        Run for simulation time and collect stats with a configured measurement periodicity.

Pseudocode 2: Pseudocode to generate **DB2**

initialization
$numGnbs \leftarrow 21$; $numUesPerGnb \leftarrow 10$
$numMcsValues \leftarrow 3$; $numRuns \leftarrow 63$
for $r \leftarrow 1$ **to** $numRuns$ **do**
  for $m \leftarrow 1$ **to** $numMcsValues$ **do**
    for $e \leftarrow 1$ **to** $numGnbs$ **do**
      for $u \leftarrow 1$ **to** $numUesPerGnb$ **do**
        Fix the Initial MCS of all the UE to m.
        Start moving away from the serving gNB, based on a random direction.
        Run for 40 seconds and collect stats every 200 msec.

Pseudocode 3: Pseudocode to generate **DB3**

initialization
$numGnbs \leftarrow 21$; $numUesPerGnb \leftarrow 10$
$numMcsValues \leftarrow 3$; $numRuns \leftarrow 8$
$numNeighboursPerUe \leftarrow 8$;
for $r \leftarrow 1$ **to** $numRuns$ **do**
  for $m \leftarrow 1$ **to** $numMcsValues$ **do**
    for $e \leftarrow 1$ **to** $numGnbs$ **do**
      for $u \leftarrow 1$ **to** $numUesPerGnb$ **do**
        for $n \leftarrow 1$ **to** $numNeighboursPerUe$ **do**
          Fix the Initial MCS of all the UE to m.
          Start moving away from the serving gNB, based on a random direction.
          HO to neighbour cell n as per event A2.
          Run for 40 seconds and collect statistics every 200 msec.

the total number of entries, i.e., the parameter n, are: 33,500 for **DB1**, 29,648 for **DB2**, 33,662 for **DB3**, and 31856 for **DB4**. Without loss of generality, the parameters, which dimension these databases are listed in Table III.

Each simulation, whereby one simulation we mean the individual run considered for 1 HO to a deterministic target cell and 1 initial MCS, lasts approximately 4,5 hours on an Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz platform. We have parallel processing capabilities with 40 cores. The raw ns-3 traces occupy 250 MB per simulation. ns-3 traces have been merged in a unique file, where 1 row corresponds to 1 UE data, and the columns are the features previously introduced. Each simulation, once post-processed, occupies 11 MB.

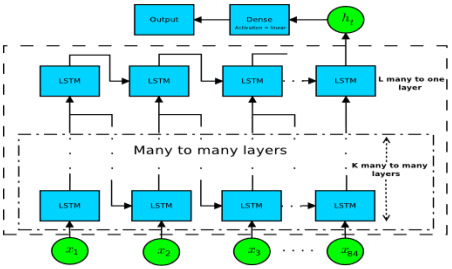


Fig. 2: Single-Task: Many to one LSTM architecture.

V. RNN MODELS FOR SINGLE-TASK AND MULTI-TASK LEARNING

In this section, we discuss the RNN models to solve the proposed RAN functions through individual and joined deep learning models, following a traditional single-task learning or an MTL approach. A RAN efficient management involves several RAN functions, which are usually handled by ML independent control loops. It means that a separate model is optimized for each task, which results in several task-specific models. However, the single-task approach presents many limitations in terms of self-coordination of the different tasks, negatively interfering among them, and is challenging from the implementation and computational perspectives. Specifically, we need models to perform multiple tasks in parallel without significantly compromising each tasks' performance. When it comes to learning multiple tasks under a single model, MTL techniques have been proposed in the literature as the solutions.

As mentioned in Section IV, the dataset consists of the measurements and traces extracted with a certain periodicity from each layer of the 3GPP protocol stack, which generates a time series of multivariate features. We believe that, by exploiting the temporal characteristic of this data one could understand the impact of HO decisions or select an appropriate initial MCS. Therefore, we propose different architectures, employing RNN with LSTM units [41]. LSTM is a special kind of RNN, which outperforms other ML approaches for time series analysis [42] [43], and solves the problem of long-term dependency issue found in vanilla RNN [44]. Specifically, we propose to model both the target use cases as regression problems using LSTM-based architectures where we aim to estimate, respectively, the time to download the file, and the initial throughput over a certain time window.

TABLE IV: Proposed approaches and RNN models.

RNN Modeks	Approaches			
Multi-layer many-to-one-LSTM	Single-Task			
	HO use case using DB1		Initial MCS use case using DB2	
Auto Encoder + MLP neural network	Multi-Task			
	Parallel MTL		Incremental MTL	
	HO use case using DB3	Initial MCS use case using DB3	HO use case using DB4	Initial MCS use-case using DB4

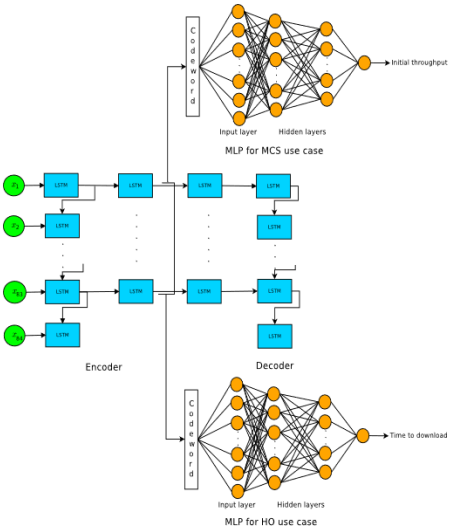


Fig. 3: Multi-Task: Joined HO management and initial MCS architecture consist of AE + MLP neural network.

A. *Single-Task Learning*

Fig. 2, shows the proposed multi-layer many-to-one LSTM architecture for single-task solutions, which is individually designed and trained to address the two selected use cases for study [16]. This model takes all the 84 features as input to infer the time to download for the HO management and the throughput for the initial MCS selection use cases. It processes them in a lag of 16800 data (i.e., 84 features x 200 time steps) samples with multiple batches of fixed size. In our previous work [16], we already discussed the effectiveness of this single-task architecture, in comparison to other options, for the HO management use case. Therefore, in this study, we leverage the same model for the initial MCS selection use case but after fine-tuning its hyperparameters, as discussed in the next Section VI-A.

B. *Multi-task Learning*

In MTL, multiple tasks, each of which can be a general learning task, i.e., supervised, unsupervised, semi-supervised, or reinforcement learning tasks, are simultaneously learned through a shared model. It is found that jointly learning these tasks can lead to performance and/or computational improvement compared to learning them individually. MTL is inspired by human learning activities where people often apply the knowledge learned from previous tasks to help learn a new task. It helps to alleviate well-known weaknesses of deep learning, like the large-scale data requirements and computational demand. We believe that it also brings an added value to the design of an intelligent RAN, where multiple correlated tasks have to be executed concurrently. The setting of multi-task learning is similar to transfer learning. The main difference, though, lies in the fact that in MTL, there is no distinction among different tasks, and the objective is to improve the performance of all the tasks, or reduce the computational component of all the joined tasks together. On the other hand, in transfer learning, the objective is to improve a target task with the support of source tasks. Learning separately multiple tasks brings difficulties that are not present in multi-task learning. It may happen that different tasks have conflicting needs. This may easily happen during the optimization of the RAN, where different tasks may intervene over the same parameters to optimize their functions independently. When the increasing performance of a model of one task hurts the performance of another task with different needs, we talk about *negative transfer*. There are many different factors to consider when creating a shared architecture, such as the portion of the model's parameters that will be shared between tasks. Many of the proposed architectures for MTL play a balancing game with the degree of information sharing between tasks: Too much sharing will lead to negative transfer and can cause the worse performance of the multi-task than the single-task model. At the same time, too little sharing does not allow the model to leverage information between tasks effectively. One commonly used multi-task architecture in computer vision follows the general vision of a global feature extractor made of convolutional layers shared by all tasks, followed by an individual output branch for each task. This architectural approach is usually referred to as *shared trunk*. Other architectures can follow alternative methods, for example, based on having a separate network for each task, with information flows between parallel layers in the different task networks. In the rest of this section, we discuss the architecture and the different options for learning that we propose to implement the MTL vision for efficient RAN management. In particular, we propose an architecture that follows a *shared trunk* architecture with hard parameter sharing [45].

The architecture is based on a multi-layer LSTM AE [46], in charge of performing the shared feature extraction in conjunction with a MultiLayer Perceptron (MLP) neural network, as shown in Fig. 3. An AE is an unsupervised ML algorithm, which learns a function to approximate an output

identical to the input. Since it is based on the *encoder-decoder* paradigm, the input is transformed into a lower-dimensional space, also known as CodeWord (CW), to more efficiently model highly non-linear dependencies in the inputs. The compression operation manages to extract more general and useful features, which retain essential aspects of a dataset [47]. Our goal is to smartly reduce the data to be used for inferring the time to download and the initial throughput. We use the same AE following a hard parameter sharing architecture, but independent MLPs to estimate the specific QoE indicator of interest for each RAN function. We opt for this LSTM based architecture, for the same reason already discussed for the single-task case, which is to take the best advantage of the temporal characteristic of the collected data.

In this line, using the model shown in Fig. 3 we propose the following two different methods for MTL learning:

- *Parallel MTL*: We explore learning behaviour when no task is given priority with respect to the others, but all tasks are concurrently learned. The reference database for training, in this case, is **DB3**.
- *Incremental MTL*: In this case, we analyze the learning behaviour when the learning is inherently incremental, meaning we first learn for one task and then incorporate information from new tasks. The advantage of this approach is that once we have trained the shared trunk architecture, we can progressively introduce more tasks to the design of the intelligent RAN, without further implementation costs. The risk, on the other hand, is that of the *catastrophic forgetting* [48], while we aim to incorporate information from new tasks without forgetting the previously learned. The reference databases to train in this case is **DB4**.

To summarize, in this work, to address two RAN use cases, i.e., HO management and initial MCS selection, we design and evaluate architectures employing different RNN models for single-task and multi-task approaches, as shown in Table IV.

VI. Performance evaluation

This section discusses training and system-level performance evaluation using the proposed models for the single, and the two multi-task approaches.

A. Training the proposed architectures

The implementation of the models is done in Python, using Keras and Tensorflow as backend. In particular, to speed up the training, testing, and evaluation of these models, we use fast LSTM implementation with Nvidia CUDA Deep Neural Network (CuDNN) library for GPUs [49]. The DBs for each of the proposed approaches 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 models using the training and validation sets to minimize the reconstruction error over 200 epochs, in case of the AE, or prediction error, in case of the single-task LSTM and MLP. The loss function used to train the models is the Mean Square Error (MSE), and the *RMSProp* algorithm is used to optimize the learning process. Moreover, a linear activation function is used for the output layer of the LSTM (see Fig. 2) and MLP, while the *Leaky ReLU* activation function is used for the hidden layers of MLP. We discuss in the following the details of the training and selected architecture for the single task and multi-task architectures.

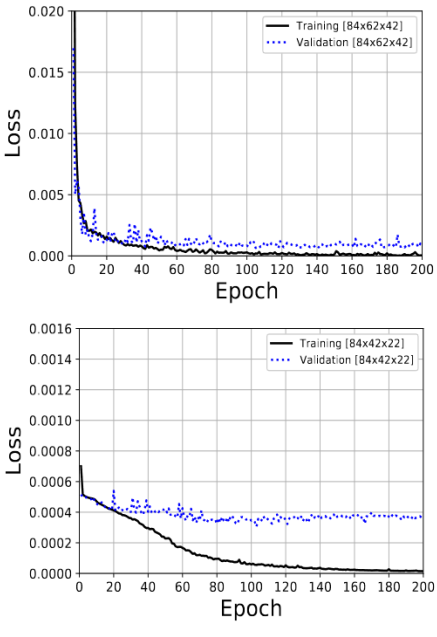


Fig. 4: Single-task training MSE per epoch. (a) HO management use case LSTM 84×62×42 (b) Initial MCS use case LSTM 84×42×22.

1) Single-task architectures: We have trained the LSTM architecture shown in Fig. 2 for the single task learning based on **DB1**. To select the hyperparameters of this model, i.e., the number of layers and the number of LSTM units (blue LSTM blocks in Fig. 2) in each hidden layer, we have tested nine different combinations. Then, we have selected the hyperparameters resulting in the lowest average MSE (over 200 epochs). Fig. 4.(a) shows the MSE per epoch of the single-task model trained to address the HO management task, using 2 and 3 layers of LSTM nodes, where the numbers separated by "x" in the legend represent the number of hidden LSTM units in each layer. We observe that, after 140 epochs, this model is able to achieve and maintain very low testing loss independently from the number of layers and cells per layer. Based on the results in Fig. 4.(a), the architecture we stick with is [84x62x42].

We follow a similar approach to train another model based on the single-task architecture in Fig. 2, to tackle the initial

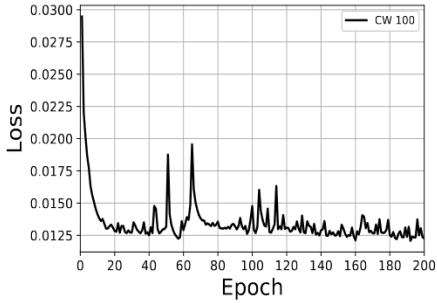
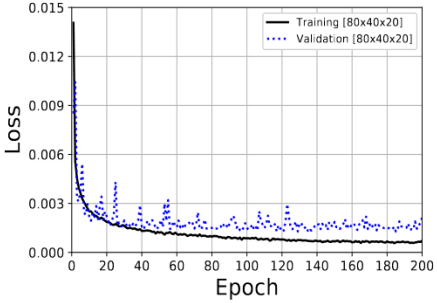
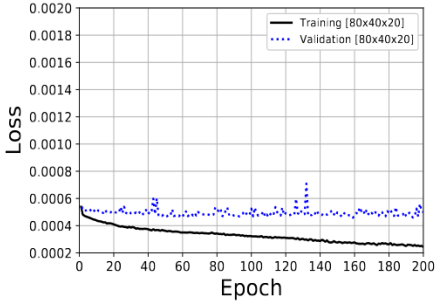


Fig. 5: Multi-task training (parallel MTL): MSE between original data and decoder for the AE trained with dB3 for codeword length 100.



(a)



(b)

Fig. 6: Multi-task parallel MTL training MSE per epoch. (a) HO management use case AE + MLP 80×40×20 (b) Initial MCS use case AE + MLP 80×40×20.

MCS use case. As shown in Table III, we focus on three MCSs (0, 14, and 28), representative of the three main available modulations (e.g., QPSK, 16QAM, 64QAM). The output to be estimated in the regression problem is the initial throughput computed over a window of 100 ms when a new RRC connection is established. The initial throughput at the beginning of the session cannot be considered because, during the initial window of a TCP connection, we are only able to capture messages from its initial handshake, which results in the same initial throughput for all UEs. As a result, we focus on the initial throughput after the second HO. We avoid taking the measurement after the first HO because the first HO is deterministic for the HO management use case. Consequently, the output of the initial throughput would be influenced by the imposed decision. Fig. 4.(b) shows the regression loss, measured as MSE, obtained with the selected LSTM architecture, i.e., [84x42x22].

2) Multi-task architecture trained with parallel MTL: For the multi-task architectures, we use a similar approach to the single-task one. The architecture now is based on a shared trunk approach, which first considers a multi-layer LSTM AE in charge of performing the shared feature extraction, and then a per task MLP is used to perform the regression. To train and validate these models, we used **DB3**, obtained to target both initial MCS and HO management use cases. It is also worth mentioning that we consider 8 runs to build **DB3**, in order to maintain a similar dimension for **DB1** and **DB2**. In this training process, first we select the CW length of the AE, among five different CW lengths of 50, 100, 200, and 300. In particular, we select a CW equal to 100 to take into the account the tradeoff between the length of the CW and the MSE of the decoder. Fig. 5 shows the AE reconstruction error, i.e., the MSE between original data and the one after decoding, common to the two use cases, using CW length of 100. Then, using this selected CW as an input to the MLP neural network, one set of hyperparameters, among 7 (based of the lowest average MSE), is chosen for the two MLPs (see Fig. 3). Similarly, Fig. 6.(a)-(b) show the regression loss of the AE plus MLP structure for the chosen MLP structure of three layers [80x40x20], to estimate the time to download and the initial throughput, for the HO use case, and for the initial MCS use case, respectively.

3) Multi-task architecture trained with incremental MTL: Fig. 7, shows the loss of the AE, considering incrementally increasing databases for training (**DB4**). We start from the AE trained with **DB1**, which is indicated in the figure with "0 runs"Then, we progressively add runs from **DB2**, and observe the behavior of the loss of the AE. We consider that the loss is comparable in all cases. Furthermore, Fig. 8.(a)-(b) show the loss of the regressors to estimate respectively the time to download the file for the HO use case, and the initial throughput for the initial MCS use case, as a function of the different number of runs. Different behaviours can be observed in the loss of the two regressors. It depends on whether the architecture is first trained for one task, or another task is incrementally learned after the first one, by adding training data to the training DB. In the HO use case, for which the architecture is initially individually trained, we observe that the loss increases when 8 runs are introduced from **DB2**. It is because the architecture has to suddenly adapt to new data coming from a DB built for a different purpose. However, as we add more runs from **DB2**, the loss trend is to get reduced.

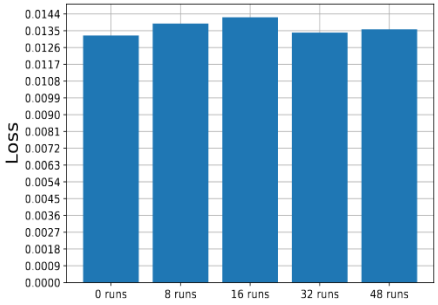


Fig. 7: Multi-task training (incremental MTL): Average MSE over 200 epochs between original data and decoder as a function of the incremental runs for the AE of codeword length 100.

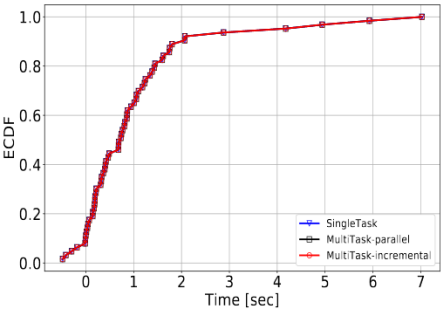
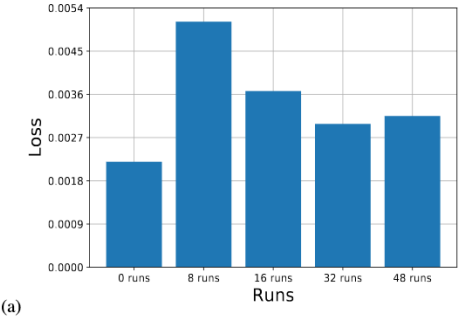
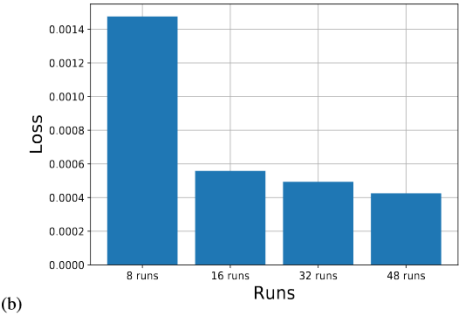


Fig. 9: ECDF of the difference of download time obtained by the benchmark A2-based HO benchmark and the ML based architectures.



(a)



(b)

Fig. 8: Multi-task incremental MTL training average MSE over 200 epochs. (a) HO management use case AE + MLP 64×32 (b) Initial MCS use case AE + MLP 64×32.

On the other hand, the loss in the estimation of the initial throughput for the initial MCS use case, which is the new use case we aim to learn by adding the new data, linearly decreases with the number of runs that we add from **DB2**. This behaviour of the loss is reasonable, since the architecture gradually improves its learning performance, as we add more information related to the MCS use case. We select the combination of AE CW length of 100 and the MLP of [64x32], which provided us the lowest average MSE for all the tested run values.

B. System level performance

The performance evaluation of these models is performed in an offline manner. In particular, to perform this evaluation, we consider another dataset generated with two extra simulation campaigns using a Run value which was not used to build the training dataset (i.e., Run 21 for the HO use case and Run 65 for the initial MCS use case). It allows us to evaluate the models in a common new data set which is different from the data sets used to learn. In the following, we analyse the results obtained using the trained models for the two use cases.

1) HO use case: For the HO management, 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 HO algorithm. The first campaign aims at gathering the file download time using the benchmark HO algorithm (e.g., A2-RSRP). The second simulation campaign is conducted in a similar way as the one to build the training dataset, i.e., it consists of 8 deterministic HOs. Following this approach, we construct 8 input strings for each neighbour of a UE, which consists of 1 row and 16800 columns (i.e, 84 features x 200 time steps). These strings are used individually as their input to obtain a predicted time to download for all the architectures, i.e., single task and multi-task (parallel and incremental learning). Finally, for each UE, we select the gNB with the minimum predicted time to download for the HO. We compare results of the ML based and the benchmark approaches for the UEs that successfully finalize the download. In particular, we compare the number of UEs completing the download and the time needed to download the file.

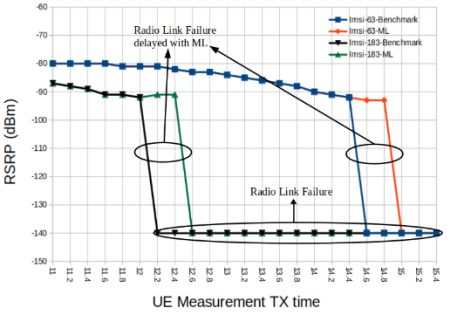


Fig. 10: Example of reduction in the duration of radio link failure with proposed ML models.

We first compare the performance of the benchmark HO algorithm with the single-task and the parallel MTL architectures. Fig. 9 shows the Empirical Cumulative Distribution Function (ECDF) of the difference between the download time observed by these UEs using the benchmark and the proposed models. The results obtained using the benchmark HO algorithm show that there are 63 (i.e., 30%) UEs out of 210, which are able to finalize the download. On the other hand, 77 (i.e., ≈37%) UEs are able to download the file successfully using the single-task and parallel MTL models. It means that the ML approach manages to increase by 18% the number of UEs able to finalize the download during the simulation time. Moreover, there are 62 common UEs, which were always able to download the file, irrespective of the tested HO solution, i.e., benchmark or ML based.

Out of these 62 UEs, the ECDF trend in Fig. 9 on the positive x-axis shows that we can reduce the file download time for 56 UEs compared to the benchmark case using the single-task or the parallel MTL architectures. However, there are 6 UEs that experience marginally higher download time than the benchmark (see the trend on –ve x-axis). We believe that their performance can be improved by increasing the size of the database used to train the models and by further fine tuning their hyper-parameters. Moreover, this evaluation shows that the MLP, fed with the AE CW of 100 performs similarly to the LSTM. It proves that the AE has efficiently transformed the inputs into a lower-dimensional space without losing the meaningful information of the dataset for the use case of the HO.

We now evaluate the capabilities of incremental MTL offered by the AE based architecture. In particular, we want to prove that an AE that is trained for a specific use case (e.g., the HO) can be reused for another use case. As mentioned in Section VI-A3, we first consider the AE and MLP model trained with **DB1**. In this DB we removed the entries where the initial throughput is not available for the reasons described earlier, e.g., when the file download finishes before the second HO. We observe that the HO performance based on this architecture is similar to the one obtained with the single-task learning. It is reasonable since with "0 runs" the DB is still purely built to handle the HO use case only. However, using the other four incrementally trained models, we notice that the performance of the HO algorithm is the same for all of them. The reason is that, even while observing some difference, the regression losses of these models are comparable and low enough to provide comparable system performance. This result is further validated when compared to the results achieved using single-task and parallel MTL, as shown in Fig. 9. In this figure, to simplify the representation, we only present the results using the incremental MTL model trained with "8 runs" from **DB2**.

TABLE V: Summary of HO use case results.

Approach	% of UEs finalizing the download	% of UEs decreasing the time to download
A2 based benchmark	30% (out of 210)	-
ML based	37% (out of 210)	90% (out of 62)

The offline evaluation performance for the HO use case reaches exactly the same results for all the approaches. It allows us to conclude that incrementally introducing runs from a different database adding new information to the system, does not jeopardize the previously learned information, in our case, where the features of the two databases are the same. For our case and the nature of the database, we do not observe any phenomenon of catastrophic forgetting, i.e., the tendency of an artificial neural network to entirely and abruptly forget previously learned information upon learning new information. More research should be conducted to evaluate how different **DB1** and **DB2** can be to maintain the same conclusion that we reach here.

Furthermore, in Fig. 10, we present, as an example, 2 UEs out of 56 UEs for which ML reduced the time spent in RLF and, consequently, the time to download the file (there are more UEs in the scenario experiencing the same performance advantage when using the ML technique). We notice that these UEs experience an RLF just before the first HO irrespective of the scheme used, i.e., benchmark or ML. In fact, once the UE is inside a coverage hole generated by an obstacle, all gNBs are unable to offer any service, and there is no coverage from any of the surrounding gNBs. As a result, an obstacle impairs the coverage of all gNBs equally. However, in those challenging situations, with the help of ML models, we can reduce the RLF duration for these UEs by 400 ms (i.e., 400 Transmission Time Intervals (TTIs)), which also improves their time to download. The reason is that ML models, thanks to their capability of learning from past experience, can identify a more appropriate neighbour gNB to HO to provide more extended service than the benchmark, and doing so reduces the RLF duration. Finally, Table V summarizes the comparative results between the different approaches for the HO use case.

2) Initial MCS use case: We evaluate the initial MCS performance, following an offline strategy, as we did previously for the HO use case, using an extra run, "Run 65". First we consider traces for the three evaluated MCSs, which provide three different input strings for each UE to get the predicted

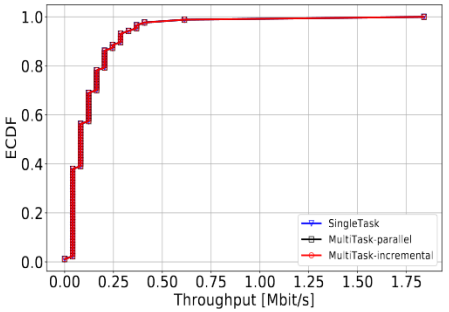


Fig. 11: ECDF of the difference between initial throughput obtained with benchmark MCS selection and the ML based architectures.

initial throughput for the selected MCS values. Then, for each UE, we choose the MCS, which results in higher initial throughput. At this point, we further filter out some UEs for those cases when the proposed ML models select MCS 0. In particular, the throughput achieved by the UEs using the benchmark scheme, which always selects MCS 0, is the same that we get when the ML based solutions also consider the same MCS. Therefore, we consider only those UEs for the performance evaluation for which the ML based models select MCS values different 0, i.e., 14 and 28.

Similar to what was observed for the HO management, it is also observed for the initial MCS use case. In particular, the gain in the performance is the same when using any of the proposed ML based models. In total, we obtain 86 UEs out of 210 for which the initial throughput takes a valid value, and the proposed approaches select a different MCS from 0. Out of these 86 UEs, 44 (i.e., 51.16%) select MCS 28 and 42 (i.e., 48.83%) select MCS 14. From the analysis of the results in Fig. 11, we conclude that all the 86 UEs that obtained a valid initial throughput get better initial throughput than the benchmark, when considering the ML based approaches. The average initial throughout per UE considering the benchmark with MCS 0 is 0,051 Mbit/s, while the average initial throughput attained using the ML based models is 0,1944 Mbps. Thus, on average, we obtain a 73.35% increment of initial throughput per UE. Moreover, for the incremental MTL approach we observe no difference in the performance of the selection of the appropriate initial MCS, when using 8, 16, 32, or 48 runs from **DB2**, to train the AE and MLP. In this case, we also believe that the average MSE using the traces only from 8 runs is already low enough (see Fig. 8.(b)) to provide the performance similar to the one using the single-task or parallel MTL approaches. Therefore, in Fig. 11 we plot only the results obtained using the incrementally trained model using 8 runs.

It is also worth mentioning that in the case of incremental MTL, we are able to obtain already acceptable results for both use cases by using the joined DB, which has a similar

TABLE VI: Summary of initial MCS use case results.

Approach	% of UEs selecting MCS 28	% of UEs selecting MCS 14	Initial throughout increase per UE
ML based	51.16%	48.83%	73.75%

dimension as of the individual **DB1** or **DB2**. On the other hand, to target the use cases with single-task approaches, we should train two independent architectures with a database of a dimension twice as big as the one we need with the incremental MTL use case. The advantage that we get with the incremental MTL, with respect to the parallel MTL or the separated single-task approaches, is at the implementation level since, at any moment, we are able to add a new function to our learning architecture by incrementally training the model. It guarantees scalability concerning all the RAN functions we wish to add to the design. Finally, Table VI summarizes the gains obtained using ML approaches over the benchmark scheme for the initial MCS use case.

VII. Conclusions

In this paper, we have presented an MTL approach based on deep architectures to provide models that optimize the operational efficiency of managing multiple RAN functions executed concurrently. To prove our MTL concept in the efficient RAN management domain, we selected two RAN use cases, 1) the HO management and 2) the selection of an initial MCS when UEs establish a new connection with a gNB. We trained our proposed models following single-task and MTL paradigms. In particular, to exploit the network data's temporal characteristics, we proposed an RNN based on LSTM for single-task learning, and an LSTM AE along with an MLP for the MTL approach.

As for the single task approaches, the results proved that the proposed ML models outperform the A2 event-based benchmark HO algorithm in terms of the number of successful downloads and time to download statistics. Besides this, our proposed models also provided the gain in terms of the increased initial throughput by selecting a better MCS than a benchmark scheme, which always selects MCS 0 upon establishing a new RRC connection. Furthermore, the results show that the models based on AE, used for the MTL parallel and incremental learning, perform similarly to the single-task model using only the LSTM. It is proven that the AE could efficiently compress the inputs into a lower-dimensional space without losing the dataset's meaningful information. The MTL solution, which allows sharing training models among RAN tasks, provides then a series of advantages at implementation, coordination, and training levels. Additionally, the model trained by employing the incremental learning approach did not suffer from the phenomenon of *catastrophic forgetting*.

References

[1] "GSMA," https://www.gsma.com/, accessed: 2021-08-01.
[2] J. Wang, H. Roy, and C. Kelly, "OpenRAN: The next generation of radio access networks,," https://telecominfraproject.com/openran/, Nov 2019.
[3] 3GPP TS 32.500, "Self-Organising Networks (SON): Concepts and requirements," Version 0.5.1, Release 8.

[4] Huawei, "White Paper: Next Generation SON for 5G," https://www.huawei.com/en/industry-insights/outlook/mobile-broadband/insights-reports/next-generation-son-for-5g.

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

[6] N. Baldo, L. Giupponi, and J. Mangues, "Big Data Empowered Self Organized Networks," in *Proc. IEEE 20th European Wireless Conference*, Barcelona, Spain, May 2014, pp. 1–8.

[7] O-RAN Alliance, "O-RAN Use Cases and Deployment Scenarios," https://www.o-ran.org/resources.

[8] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, Mar 2019.

[9] R. Caruana, "Multitask Learning: A Knowledge-Based Source of Inductive Bias," in *Proc. Tenth International Conference on Machine Learning*, Amherst MA, USA, Jul 1993, pp. 41–48.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1464–1468, Jun 2013.

[11] Y. Ju, J. Guo, and S. Lui, "A Deep Learning Method Combined Sparse Autoencoder with SVM," in *Proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Xi'an, China, Sept 2015, pp. 257–260.

[12] M. Ghifari, W. Kleijn, M. Zhang, and D. Balduzzi, "Domain Generalization for Object Recognition with Multi-task Autoencoders," in *Proc. International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 2551–2559.

[13] ns 3. 2020, "Network Simulator," http://code.nsnam.org/ns-3-dev.

[14] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. early access, pp. 1–1, Feb 2021.

[15] M. Kanakis, D. Bruggemann, S. Saha, S. Georgoulis, A. Obukhov, and L. Van Gool, "Reparameterizing Convolutions for Incremental Multi-Task Learning Without Task Interference," in *Proc. European Conference on Computer Vision (ECCV)*, Glasgow, UK, 2020, pp. 689–707.

[16] 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. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, London, UK, Sept 2020, pp. 1–6.

[17] M. Miozzo, Z. Ali, L. Giupponi, and P. Dini, "Distributed and Multi-Task Learning at the Edge for Energy Efficient Radio Access Networks," *IEEE Access*, vol. 9, pp. 12 491–12 505, 2021.

[18] J. Wang, C. Jiang, H. Zhang, Y. Ren, K. C. Chen, and L. Hanzo, "Thirty Years of Machine Learning: The Road to Pareto-Optimal Wireless Networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[19] P. V. Klaine, M. A. Imran, O. Onireti, and R. D. Souza, "A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.

[20] I. Pappalardo, A. Zanella, and M. Zorzi, "Upper Bound Analysis of the Handover Performance in HetNets," *IEEE Communications Letters*, 2017.

[21] P. Tseng, K. Feng, and C. Huang, "POMDP-Based Cell Selection Schemes for Wireless Networks," *IEEE Communications Letters*, vol. 18, no. 5, pp. 797–800, 2014.

[22] F. Guidolin, I. Pappalardo, A. Zanella, and M. Zorzi, "Context-Aware Handover Policies in HetNets," *IEEE Transactions on Wireless Communications*, vol. 15, no. 3, pp. 1895–1906, 2016.

[23] S. S. Mwanje and A. Mitschele-Thiel, "Distributed cooperative Q-learning for mobility-sensitive handover optimization in LTE SON," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, Funchal, Portugal, Jun 2014, pp. 1–6.

[24] Z. Becvar and P. Mach, "Adaptive Hysteresis Margin for Handover in Femtocell Networks," in *Proc. 6th International Conference on Wireless and Mobile Communications*, Valencia, Spain, Nov 2010, pp. 256–261.

[25] J. Wu, J. Liu, Z. Huang, and S. Zheng, "Dynamic fuzzy Q-learning for handover parameters optimization in 5G multi-tier networks," in *Proc. International Conference on Wireless Communications Signal Processing (WCSP)*, Nanjing, China, Dec 2015, pp. 1–5.

[26] Z. Ali, N. Baldo, J. Mangues, and L. Giupponi, "Machine Learning Based Handover Management for Improved QoE in LTE," in *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Istanbul, Turkey, Apr 2016, pp. 794–798.

[27] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover Optimization via Asynchronous Multi-User Deep Reinforcement Learning," in *Proc. IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, Jul 2018, pp. 1–6.

[28] C. Lee, H. Cho, S. Song, and J. Chung, "Prediction-Based Conditional Handover for 5G mm-Wave Networks: A Deep-Learning Approach," *IEEE Vehicular Technology Magazine*, vol. 15, no. 1, pp. 54–62, 2020.

[29] J. P. Leite, P. H. P. de Carvalho, and R. D. Vieira, "A flexible framework based on reinforcement learning for adaptive modulation and coding in OFDM wireless systems," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Paris, France, Jun 2012, pp. 809–814.

[30] R. Bruno, A. Masaracchia, and A. Passarella, "Robust Adaptive Modulation and Coding (AMC) Selection in LTE Systems Using Reinforcement Learning," in *Proc. IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, Vancouver, BC, Canada, Dec 2014, pp. 1–6.

[31] L. Zhang, J. Tan, Y. Liang, G. Feng, and D. Niyato, "Deep Reinforcement Learning-Based Modulation and Coding Scheme Selection in Cognitive Heterogeneous Networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3281–3294, Apr 2019.

[32] A. Shahmansoori, "Sparse Bayesian Multi-Task Learning of Time-Varying Massive MIMO Channels With Dynamic Filtering," *IEEE Wireless Communications Letters*, vol. 9, no. 6, pp. 871–874, Feb 2020.

[33] 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 Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 362–10 374, Jun 2020.

[34] 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 Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2208–2225, Jan 2020.

[35] 3GPP TS 36.331, "Radio measurement collection for Minimization of Drive Tests (MDT); Overall description," version 10.4.0, Release 10.

[36] 3GPP TR 32.862, "Study on Key Quality Indicators (KQIs) for service experience," Version 14.0.0, Release 14.

[37] ITU-T Recommendation, "Estimating end-to-end performance in IP networks for data applications," Series G, G.1030.

[38] M. Khan and U. Toseef, "User utility function as quality of experience (QoE)," in *Proc. of the Tenth International Conference on Networks*, The Netherlands, Jan 2011, pp. 99–104.

[39] J. Mendoza, I. de-la Bandera, D. Palacios, and R. Barco, "Qoe optimization in a live cellular network through rlc parameter tuning," *Sensors*, vol. 21, no. 16, 2021.

[40] 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 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, Miami, Florida, USA, Oct 2011, pp. 293–298.

[41] S. Hochreiter and J. A. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov 1997.

[42] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE INFOCOM - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[43] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. IEEE 29th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Bologna, Italy, Sept 2018, pp. 1827–1832.

[44] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Oct 2000.

[45] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *ArXiv*, vol. abs/2009.09796, Sept 2020.

[46] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Proc. Advances in Neural Information Processing Systems 28*, 2015, pp. 3079–3087.

[47] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. 21st International Conference on Artificial Neural Networks*, Espoo, Finland, Jun 2011, pp. 52–59.

[48] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, Apr 1999.

[49] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

Zoraze Ali received his MSc degree in Radio Communication from Blekinge Institute of Technology, Karlskrona Sweden. He joined Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) in September 2014 as a research assistant to pursue his PhD degree on Self Organizing Networks (SON), Big Data and Machine Learning with particular focus on LTE/LTE-A and 5G networks at Telematics Engineering Department of Universitat Politècnica de Catalunya (UPC), Barcelona. His research interests include wireless communications and machine learning.

Lorenza Giupponi received her Ph.D. degree from UPC, Barcelona, Spain, in 2007. She joined the Radio Communications Group of UPC in 2003 with a grant of the Spanish Ministry of Education. During 2006 and 2007 she was assistant professor in UPC. In September 2007 she joined the CTTC where she is currently a Research Director in the Mobile Networks Department of the Communication Networks Division. Since 2007 she is also a member of the Executive Committee of CTTC, where she acts as the Director of Institutional Relations. She is the co-recipient of the IEEE CCNC 2010, IEEE 3rd Int. workshop on Indoor and Outdoor Femto Cells 2011, and IEEE WCNC 2018 best paper awards. Since 2015 she is a member of the Executive Committee of ns-3 consortium.

Marco Miozzo received his M.Sc. degree in Telecommunication Engineering from the University of Ferrara (Italy) in 2005 and the Ph.D. from the Technical University of Catalonia (UPC) in 2018. After graduated, he worked as a Research Engineer in wireless networking for the Consorzio Ferrara Ricerche (CFR), where he collaborated with the Department of Information Engineering (DEI) of the University of Padova, both in 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. Currently he is collaborating with the EU founded H2020 SCAVENGE (MSCA ETN). He participated in several R&D 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 are: sustainable mobile networks, green wireless networking, energy harvesting, multi-agent systems, machine learning, green AI, energy ethical consumerism, transparent and explainable AI.

Paolo Dini received MSc and PhD from Università di Roma La Sapienza, in 2001 and 2005, respectively. He served as a Post-Doc Researcher at the Research Centre on Software Technology (RCOST) - Università del Sannio, and contracted Professor at Università di Roma La Sapienza in 2005. He is now with the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) as a Senior Researcher. 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 the 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.