IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xxxx

Mobile Traffic Prediction at the Edge through Distributed and Deep Transfer Learning

ALFREDO PETRELLA, MARCO MIOZZO[1], PAOLO DINI[1]
[1]CTTC/CERCA, Av. Carl Friedrich Gauss, 7, 08860, Castelldefels, Barcelona, Spain (e-mails: petrellalfredo@gmail.com, marco.miozzo, paolo.dini@cttc.es

Corresponding author: Alfredo Petrella (e-mail: petrellalfredo@gmail.com).

This publication has been partially funded by the Spanish project PID2020-113832RB-C22(ORIGIN)/MCIN/AEI/10.13039/50110001103 and the grant CHIST-ERA-20-SICT-004 (SONATA) by PCI2021-122043-2A/AEI/10.13039/501100011033

ABSTRACT Traffic prediction represents one of the crucial tasks for smartly optimizing the mobile network. Recently, Artificial Intelligence (AI) has attracted attention to solve this problem thanks to its ability in cognizing the state of the mobile network and make intelligent decisions. Research on this topic has concentrated on making predictions in a centralized fashion, i.e., by collecting data from the different network elements and process them in a cloud center. This translates into inefficiencies due to the large amount of data transmissions and computations required, leading to high energy consumption.
In this work, we investigate a fully decentralized AI solution for mobile traffic prediction that allows data to be kept locally, reducing energy consumption through collaboration among the base station sites. To do so, we propose a novel prediction framework based on edge computing and Deep Transfer Learning (DTL) techniques, using datasets obtained at the edge through a large measurement campaign. Two main Deep Learning architectures are designed based on Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) and tested under different training conditions. hlSimulation results show that the CNN architectures outperform the RNNs in accuracy and consume less energy. In both scenarios, DTL contributes to an accuracy enhancement in 85% of the examined cases compared to their stand-alone counterparts. Additionally, DTL significantly reduces computational complexity and energy consumption during training, resulting in a reduction of the energy footprint by 60% for CNNs and 90% for RNNs. Finally, two cutting-edge eXplainable Artificial Intelligence techniques are employed to interpret the derived learning models.

INDEX TERMS Deep Transfer Learning, Edge Computing, Energy Efficiency, Machine Learning, Mobile Traffic Prediction.

I. INTRODUCTION

TRAFFIC analysis and prediction represent crucial tasks for mobile network operators to properly and efficiently manage the network, by implementing, e.g., network planning, Quality of Experience (QoE) management, anomaly detection algorithms, and cybersecurity frameworks. Historically, these tasks have been performed by mobile network operators through their data collected for billing, the Call Detail Records (CDRs). This solution relies on dedicated servers that retrieve the data from the base stations and then process it in the core part of the network. Therefore, data has to be transmitted from the edge to central servers, which implies possible privacy issues due to the transmission of sensitive information, higher network congestion probability, and higher energy consumption due to the massive volume of data to be exchanged.

The availability of such a huge amount of data, together with the evolution in computational capabilities, allows taking advantage of Deep Learning (DL) to tackle multiple problems in network management. DL has been identified as one of the pillars of the sixth generation (6G) mobile networks to address intelligent management solutions [1]. Nevertheless, there is a complexity downside that is typically not taken into account when envisioning these intelligent, autonomous, DL-based technologies. In fact, Artificial Neural Networks (ANNs) adopted in DL are made up of many neurons and layers identified by a high number of trainable parameters to execute identified tasks with high accuracy. This implies extremely high computational complexity, which requires an equally high energy consumption [2]. From a mobile operator perspective, energy accounts for 20-40% of its operational expenditures, pushing with urgency for cost reduction through

network optimization due to the low revenue growth environment [3]. To do so, AI is expected to be the main pillar thanks to its ability in cognizing the state of the mobile network and make intelligent decisions [4]. Therefore, it is of paramount importance to adopt sustainable design principles for AI [5] and advocate for efficiency as an evaluation criterion, alongside accuracy and other related measures.

Multi-access Edge Computing (MEC) represents an efficient solution to address data privacy, network congestion, and energy sustainability issues [6]. The main idea behind MEC is to process data directly at the edge, without sending it to central servers that may be managed by a third party. This approach solves both privacy and congestion concerns. Moreover, it has been demonstrated that MEC can save up to 25% of the network energy consumption [7]. These savings are mainly enabled by reduced communication costs (avoiding data transmission to the central server) and smaller and energy-efficient devices, which use less power for refrigeration compared to data centers.

In line with the above, we propose a distributed DL solution for traffic prediction implemented directly at the base stations. Deep Transfer Learning (DTL) [8] is leveraged to additionally reduce the complexity of the training phase and, consequently, the energy consumed. The main principle of DTL is to exploit the knowledge of a teacher network trained with general data and then retrain student networks on a more specific local dataset. This allows the reduction of the computational capability required by the edge devices [9]. In particular, we propose to exploit the learned knowledge for the other student base stations in a collaborative and distributed fashion.

In addition, we provide insights on the DL models built for traffic prediction through Visual-based eXplainable AI (v-XAI) [10]. These tools give explanations of black-box models to reveal their behavior and underlying decision-making mechanisms with a specific focus on how to visually represent their results for a general audience. The analysis performed allows for a better understanding of the output of both DL models and the DTL process.

In this work, data collected from several operative base stations located in Barcelona, Spain are used. Data is collected with OWL [11], a tool capable of decoding the unencrypted Physical Downlink Control Channel (PDCCH) of the Long Term Evolution (LTE) radio transmission technology. Among the control data contained in the PDCCH, the Downlink Control Information (DCI) messages can be retrieved, which contain radio-link level settings for the user communication, both for the downlink and uplink channels. This information includes, among others, the Radio Network Temporary Identifier (RNTI) (a temporary ID assigned to each UE), the Modulation and Coding Scheme (MCS), and the number of allocated Resource Blocks (RBs) per frame. Therefore, it is possible to infer the resources allocated to the users and the throughput of the base station without accessing the actual transmitted data, guaranteeing higher user privacy. The presented solution, based on control data and avoiding usual deep packet inspection techniques from the user plane, reduces the memory footprint for processing and includes an additional level of privacy since it does not need to access the user-generated content.

Although the literature contains papers using classical centralized learning in data center for mobile traffic prediction, only few works have explored a distributed paradigm. In particular, a fully decentralized framework, where no central servers are required, is still lacking at the time of this writing, as described in Section III. We investigate Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) both with standard stand-alone databases and with the DTL paradigm. Simulation results show that the predictions in the local datasets present good levels of accuracy. The models evaluated with DTL have improved accuracy performance with respect to their corresponding stand-alone versions. All the results obtained are in line with benchmark solutions based on Support Vector Regression (SVR) [12] applied to the stand-alone database. The analysis also details the difficulties in applying Transfer Learning (TL) to SVR. Moreover, we measure the complexity of the proposed models together with the energy savings obtained by the adoption of DTL, which can reach up to 60% for CNN and 90% for RNN. Finally, the learned models are interpreted through the XAI SmoothGrad [13] and Layer-wise Relevance Propagation (LRP) [14] algorithms. It is to be noted that the XAI analysis represents a novel contribution to this field, as it has never been applied to mobile traffic prediction models, to the best of our knowledge.

As a result, the original contributions of the paper are summarized in the following list.

- We design a novel and efficient MEC-based framework for collecting and processing mobile data obtained by passively sniffing LTE control channel from different base stations. An Exploratory Data Analysis (EDA) for evaluating the main characteristics of the different datasets is also provided.
- Two mobile traffic prediction models based on RNNs and CNNs are designed. The performance of the proposed models is assessed with both stand-alone and DTL paradigms. A comparison with SVR-based solutions as a benchmark is also presented.
- The complexity and the energy footprint of the proposed models are analyzed.
- The interpretation of the obtained models is discussed by applying two XAI algorithms (SmoothGrad and LRP).

The rest of the paper is organized as follows. The background on MEC and the architectural scenario for traffic profiling at the edge is introduced in Section II. The previous literature and the innovative aspects of this work are presented in Section III. The datasets used and the analysis of their main features are described in Section IV. Section V reports the description of the tackled tasks and the presented DL models and architectures. Section VI contains a dissertation about the models' training, accuracy and complexity performance

together with their energy consumption. In Section VII, the outcomes of the two XAI techniques are discussed. Finally, Section VIII concludes the paper with remarks and possible future directions.

II. BACKGROUND

Recently, the architectural availability of edge computing resources to execute AI directly at the edge has attracted significant attention. From the service side, it would help to support Ultra-Reliable Low Latency Communications (URLLC) scenarios, like factory automation, autonomous driving, remote surgery, and augmented/virtual reality [15]. However, locating the data processing in the proximity of its source enables many more benefits:

- *Computation*: the algorithms process only local data, which implies smaller datasets and, thus, the use of less demanding hardware, both in terms of computational and memory requirements.
- *Communication*: proximity allows for the reduction of data transmission across different elements, thereby alleviating network congestion.
- *Privacy*: distributed data avoids its passage through different network elements, which prevents leakage [16].
- *Energy*: the advantages in computation and communication enable a reduction in energy used.

Regarding the possible applications, AI will play an important role in providing solutions to the resource management problem in mobile communications [4]. Typical examples are designing and optimizing 6G architectures, protocols, and operations. In particular, the traffic load significantly affects the quality of experience (QoE) perceived by users, as their connections have to share the network resources. Based on this traffic load, an intelligent cognitive engine can help in reconfiguring the mobile network to avoid traffic congestion during high peaks, reducing energy consumption when traffic is low [4]. However, predicting the traffic load is not trivial since it is correlated to many parameters, e.g., channel usage, link conditions, users' mobility patterns, etc. In such cases, the challenges are on the model definition of complex 6G elements, which often has to be defined as a tractable Markov decision process, and the algorithm deployment, since it has to work online. Consequently, a trade-off between optimality and efficiency has to be found. In the 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. To correctly perform RRM, traffic prediction plays a crucial role. By providing such functionality directly at the edge, the aforementioned benefits can be leveraged, leading to more accurate traffic prediction algorithms that consume fewer resources both from communication and energy perspectives. In this respect, the current operational state of 5G is proving to be unsustainable due to the vast volume of data being generated, transmitted, processed, and stored [17]. Moreover, data will experience exponential growth in the future, and by 2035, it is expected to reach an annual total of 2000 zettabytes, constituting approximately 20% of the world's total energy consumption [18]. Thus, edge computing represents an important enabler toward an AI-native environmentally sustainable 6G [19].

However, distributing the computation requires the use of specific techniques. The *edge intelligence* paradigm, also called *edge AI* [20], [21], aims at evaluating distributed solutions to run ML models (the *inference* phase) and to train ML models (the *training* phase). With respect to the training phase, the main problem of a distributed solution is the convergence of a consensus and how to synchronize and update the local gradients. The most popular solution for distributed training is represented by Federated Learning (FL) [22]. In this solution, the server is in charge of combining the results of the local models' training. However, the local learning methods are based on Stochastic Gradient Descent (SGD) and thus they are not optimized for working with unbalanced and non-Independent and Identical Distribution (IID) data.

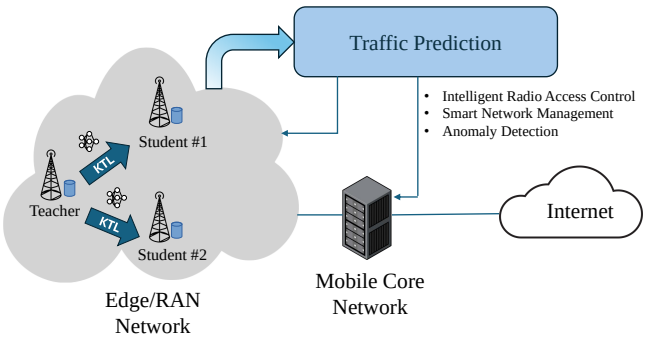


FIGURE 1: Example of the Edge AI scenario in mobile networks.

In this work, we are interested in evaluating collaborative solutions for training ML models for traffic prediction based on DTL. The aim is to avoid the synchronization problems of FL and be able to use the acquired knowledge from one BS site to others to save energy while maintaining similar accuracy with respect to benchmark centralized solutions. An example of the considered scenario is provided in Fig. 1. Thanks to the proposed solution operators can learn the traffic patterns of their users in a specific area covered by a base station and thus optimize the management of such area, for example in terms of resource allocation (i.e., intelligent radio access control and smart network/device management [23]) and anomaly detection (i.e., identify different types of anomalous events generated by flash crowds [24]). This can be done without moving data with privacy issues from the edge. Moreover, the DTL framework enables the usage of the knowledge acquired from one antenna site to be used in another site, thus saving energy thanks to the reduced computation and communication (i.e., only the ML model has to be exchanged among the sites). Furthermore, in this work data from the unencrypted control channel of an operative mobile network is exploited to properly characterize the mobile traffic patterns

at the base station site, thus maintaining a higher level of privacy for users' data and reducing storage and monitoring processing, as detailed in Section III.

III. RELATED WORK

Traffic prediction has been tackled for years due to its relevance for mobile network management. Traditional approaches involve SVR and AutoRegressive Integrated Moving Average (ARIMA) models [25] [26]. In the last few years, thanks to the evolution in computational capabilities, DL methods have overcome the traditional Machine Learning (ML) models. This is due to DL's structural ability to successfully recognize complex patterns and relationships present in mobile traffic time series [27], and in other networking-related tasks [28] [29]. Most of the literature regarding traffic prediction with DL is characterized by a centralized approach. In this category, good traffic prediction results have been obtained in [30] through densely connected two-dimensional CNNs, which are able to spot both the spatial and temporal dependence of cell traffic. In [31], an autoencoder-based deep model for spatial modeling and Long Short-Term Memory units (LSTMs) for temporal modeling was proposed and tested using a dataset provided by China Mobile. Similarly, in [32] LSTMs equipped with stochastic connectivity were proposed with the aim of reducing the considerable computing cost of the corresponding vanilla version. Combinations of the two architectures mentioned above were effective as well, as presented in [33] [34].

However, in centralized approaches data needs to be transmitted from the edge to dedicated servers. To deal with such issues, distributed approaches based on FL have been recently studied, where only local ML models are exchanged with a central server. In [35] CDR was used to train models for different neighborhoods of the city of Milan with a FL paradigm. In addition, Model-Agnostic Meta-Learning (MAML) has been used to train a sensitive initial model that can adapt to heterogeneous scenarios in different regions. It is to be noted that, the majority of the cited works use CDR, i.e., mobile network operators' data which is used for billing purposes. This approach clearly implies potential privacy leaks, as well as a large amount of data to be exchanged within the network, resulting in considerable energy consumption.

To address the privacy issue, the same datasets considered in this work have already been adopted in [36], where the authors present an FL-based solution to show its effectiveness considering the non-IID-ness of the data and different federated aggregators. Simulation results show that the proposed FL-based solutions reduce carbon emissions and energy consumption. However, the FL paradigm is based on a two-tier architecture where the MEC server has to coordinate during the learning phase with all the clients located in the proximity of the data sources. This might slow down the training process due to high network latency, unreliable links, or straggled clients. In addition, the central MEC server can be impacted by the single point of failure problem, i.e., if it becomes unreachable due to network issues or an attack, the training process is halted. Similarly, it may also become a bottleneck when the number of clients is very large.

Alternatively, this work aims to demonstrate the advantages of building one generic model to be trained directly on distributed sources of data (i.e., the base stations) to perform prediction and to apply DTL from the different sites.

Finally, to the best of our knowledge, XAI algorithms have never been applied to mobile traffic prediction models. Therefore, this work represents the first attempt to extract insights about the way models combine input variables to perform highly accurate predictions and to investigate how to improve model training when applying DTL. Similarly, the energy footprint assessment represents a unique contribution to this field. Table 1 provides a summary of the key points discussed in this section.

IV. DATA EXPLORATION

The datasets have been obtained leveraging the information collected at the base stations' level through OWL [11]. OWL is an open-source tool that allows decoding the DCI messages carried in the LTE PDCCH, which contains radio-link level settings for the user communication between the User Equipments (UEs) and the base station, both for the downlink and uplink channels. The data considered in this paper was collected from three different base stations located in the districts of Poble-sec (PS), El Born (EB), and Les Corts (LC), in the metropolitan area of Barcelona, Spain. The first one consists of approximately twenty-eight days of records, while the others have seven and twelve days of history, respectively.

The datasets are preliminary parsed, reorganized, and downsampled with a granularity of two minutes to mitigate the effect of missing values in the resulting dataset. In fact, the data collection phase is affected by decoding errors, resulting in missing values for the variables stored in the messages. Instances in which at least one variable is missing are removed, totaling roughly seventeen minutes from PS, six minutes from EB, and nine minutes from LC, which represent less than $2‰$ of the original datasets. Thus, downsampling allows a reduction in the number of instances where the value of at least one variable is missing for the entire time window and, in turn, enables the usage of DL schemes.

In this work, 5 variables are considered based on their importance with respect to the traffic prediction problem studied, namely:

- $RNTI_{count}$: the number of unique RNTIs; it can be interpreted as the number of users that communicate at least once with the base station within each time window.
- MCS_{down} and MCS_{up}: the average MCS assigned in downlink and uplink within each time window.
- RB_{down} and RB_{up}: the fraction of assigned RBs in downlink and uplink over the maximum number of available RBs within each time window; in the considered scenario this value ranges in $[0, 100] \cap \mathbb{N}$, which corresponds to the case of 20 MHz of channel bandwidth.

The resulting correlation among the considered variables is reported in Fig. 2 using Pearson coefficient. The matrix shows

TABLE 1: Comparison of Related Work approaches with the Proposed Solution.

	Centralized	Distributed with central server (federated)	Distributed server-less (de-centralized)	Privacy protection	XAI-evaluated	Energy footprint assessment
[30]	✓	✗	✗	✗	✗	✗
[31]	✓	✗	✗	✗	✗	✗
[32]	✓	✗	✗	✗	✗	✗
[33]	✓	✗	✗	✗	✗	✗
[34]	✓	✗	✗	✗	✗	✗
[35]	✗	✓	✗	✗	✗	✗
[36]	✗	✓	✗	✓	✗	✗
Proposed Solution	✗	✗	✓	✓	✓	✓

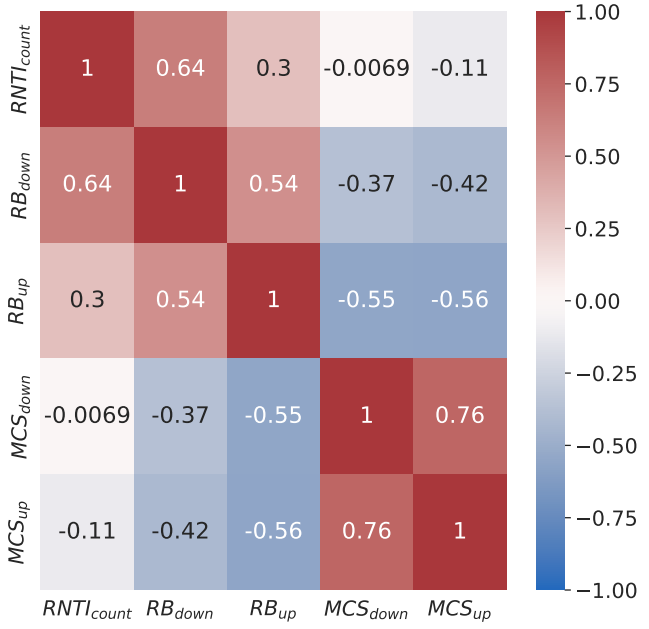


FIGURE 2: Average Pearson correlation matrix across the three datasets, each containing the same five variables. This matrix represents the average pairwise Pearson correlation coefficients calculated for the three datasets.

a poor correlation among the 5 data features. In particular, RB_{down} is more correlated to $RNTI_{count}$ than RB_{up}. This is due to the fact that downlink traffic is predominant in LTE with respect to uplink. Finally, it is interesting to note that the MCS columns are negatively correlated with the percentage of assigned RBs. This is because a low average MCS, whether in the downlink or uplink, indicates degrading channel quality, requiring more RBs to transmit the same amount of data.

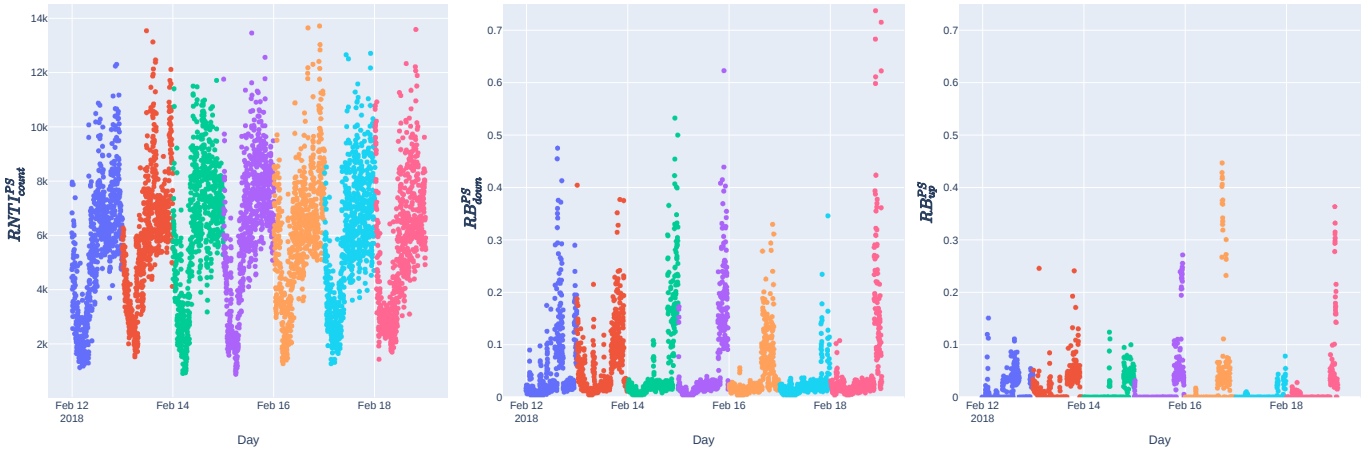
Fig. 3 displays a one-week snapshot of the dataset variables. In particular, the evolution in time for PS data is outlined for $RNTI_{count}$, RB_{down} and RB_{up}, respectively. The data is characterized by a clear daily periodicity. This daily pattern is also evident across all other datasets, with slight differences appearing only in EB, where higher peaks occur on weekend nights, reflecting the popularity of the El Born district as a nightlife center in the city.

Finally, the impact on the scale proportion and the dynamics between the downlink and uplink channels of the three selected datasets is analyzed. Here, human behavior in the neighborhoods where the data was collected is of key importance: Poble-sec is mainly a residential area, while El Born boasts a lively nightlife and Les Corts hosts the Camp Nou football stadium. For this analysis, the total traffic demand as a function of the actual data transmitted is evaluated. Driven by this aim, we define THR_{down} and THR_{up} as a-posteriori estimation of the cumulative Transport Block Size (TBS) in bits in downlink and uplink, respectively, evaluated according to 3GPP specifications [37]. These metrics are reported in Fig. 4 for the different datasets during a week, for the sake of readability. The patterns between the uplink and downlink channels are similar for the PS and EB datasets; however, the range of values significantly differs, being THR_{up} lower than THR_{down}. Alternatively, for LC, THR_{down} is much smaller with respect to the other databases, and THR_{up} shows peaks higher than THR_{down}. These peaks correspond to periods when football matches occur at the Camp Nou stadium, located near the base station. We refer to our previous papers [38] and [39] for a more comprehensive study on mobile traffic analysis during these events.

V. TRAFFIC PREDICTION MODELS

A. PROBLEM STATEMENT

Let $\mathcal{T}$ be the total measurement period of a given dataset. For every time $t \in \mathcal{T}$, $\boldsymbol{x}(t)$ is defined as the vector containing the m input features. Note that $\mathcal{T}$ changes for each dataset, and this formulation is intended to be general for all of them. Similarly, the time interval $[t, t+1]$ is 2 minutes for every dataset, as justified in Section IV. The prediction model is organized as follows. Let p be the number of input observations (samples), in terms of the number of dataset entries, constituting a single input instance, i.e., each input frame of the models will have shape $p \times m$: $\{\boldsymbol{x}(t), \boldsymbol{x}(t+1), ..., \boldsymbol{x}(t+p-1)\}$. Let Δn



(a) (b) (c)

FIGURE 3: Variables time series on a sample of 7 days of the PS dataset for $RNTI_{count}$ (a), RB_{down} (b) and RB_{up} (c). Each color corresponds to a day of the week, starting from Monday which is the first represented with blue color.

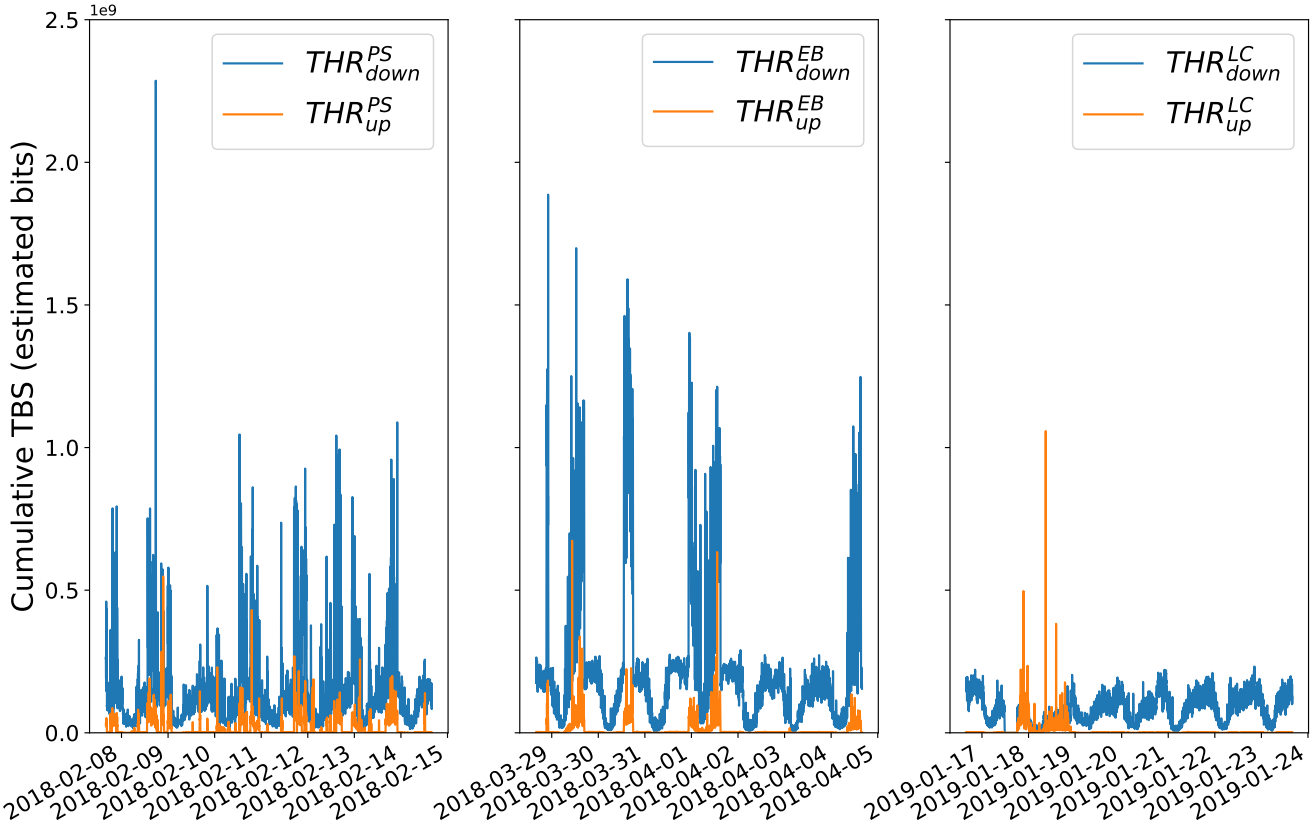


FIGURE 4: THR_{down} and THR_{up} time series by date for the three datasets on a sample of 7 days. The shared y-axis unit of measurement is billions of bits.

be the time of the output frame to be predicted in terms of the number of samples with respect to the input frame, i.e., for a specific input frame $\boldsymbol{x}(t)$ the prediction will be the vector of q variables $\boldsymbol{y}(t + \Delta n)$. The scenarios with $p \in \{10, 15, 20\}$ and $\Delta n \in \{0, 4, 9, 14\}$ have been studied; in other words, respectively, 20, 30 and 40 minutes of past data are exploited to predict samples expected to occur 2, 10, 20 and 30 minutes after.

Given the multiple variables of $\boldsymbol{x}(t)$, *data normalization* is performed for all the columns in each dataset in the interval $[-1, 1]$ to keep the data symmetric with respect to the activation functions of the DL models used (i.e., the tanh).

Moreover, *data windowing* is applied to the sequence $\boldsymbol{x}(t)$, which is split and grouped using a fixed-length window of p samples. The window is moved each time by one step. The value of p defines how many time lags are processed by the DL models. The multi-variate sequence can be expressed as $[\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)]$, where T is the cardinality of $\mathcal{T}$, i.e., $T = |\mathcal{T}|$. After the split, $N = T - p + 1$ sequences $\mathbf{x}(n)$, $n \in [1, N]$ are available:

$$\mathbf{x}(1) = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(p)]$$
$$\mathbf{x}(2) = [\boldsymbol{x}(2), \ldots, \boldsymbol{x}(p+1)]$$
$$\mathbf{x}(N) = [\boldsymbol{x}(N), \ldots, \boldsymbol{x}(T)]$$

A sequence $\mathbf{x}(n)$ has length p and each of its elements is m-dimensional. Hereafter, we refer to the sequence $\mathbf{x}$ as *samples*. Then, we define $\mathbf{X}$ as the three-dimensional matrix which contains N sequences of $\mathbf{x}$. The matrix $\mathbf{X}$ has dimension $N \times p \times m$ and serves as the input *tensor* to the DL algorithms.

The multivariate input and output of the prediction model have been designed based on the EDA in Section IV and driven by the physical meaning of the features. The resulting input features are the following $m = 5$ variables: $RNTI_{count}$, RB_{down}, RB_{up}, MCS_{down} and MCS_{up}. Regarding the output, $q = 5$ variables are exploited in the prediction models because of their valuable meaning for network operators, namely $RNTI_{count}$, RB_{down}, RB_{up}, THR_{down} and THR_{up}. The reason behind this choice is that these variables represent key performance indicators for network management, as detailed in Section II.

In what follows, different DL architectures for the mobile traffic prediction problem are tailored. In particular, the proposed models are based on RNN and CNN architectures to exploit the temporal characteristics of the datasets under study. The proposal accounts for both stand-alone models and the DTL paradigm.

B. RNN MODEL

Among the several RNN DL models, Gated Recurrent Units (GRUs) are adopted due to their high effectiveness and greater simplicity compared to other popular architectures, such as the LSTM cells [40].

After a comprehensive assessment of different configurations for the network structure, the architecture shown in Fig. 5 proved to be the best for the considered task. The intuition behind its effectiveness is based on the aim to create higher-level abstractions and capture more nonlinearity between the data through the stacked layers. Moreover, the intent is to enforce the data correlation with the previous inputs, which is provided by the structure of each GRU cell. In detail, after the input layer, a first recurrent layer is inserted, containing n_1 GRU cells with sigmoid activation for the input, forget and output gates, and hyperbolic tangent activation for the hidden and output states; dropout regularization with probability parameter D' is also implemented to prevent overfitting. The second and third layers are based on GRU with the same activation function of the first, and contain respectively n_2 and n_3, whereas the fourth layer contains n_4 such GRU cells and dropout regularization with probability parameter D''. At the end, the tensor is flattened by introducing the final dense layer containing q neurons, and a hyperbolic tangent activation function is added to obtain the prediction output. Note that the final activation is selected to meet the rescaling interval of the normalized output data ($[-1, 1]$), whereas the choice of adding dropout regularization to the first and last layers of the network is motivated by [41].

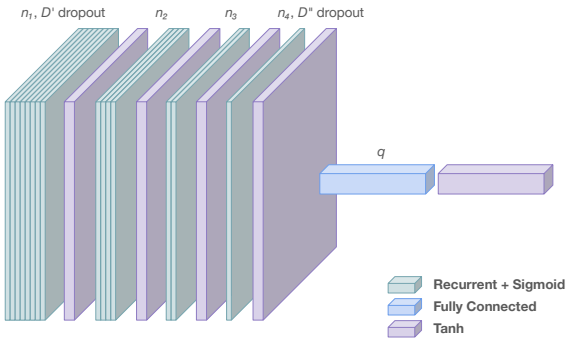


FIGURE 5: RNN model's structure graphical representation.

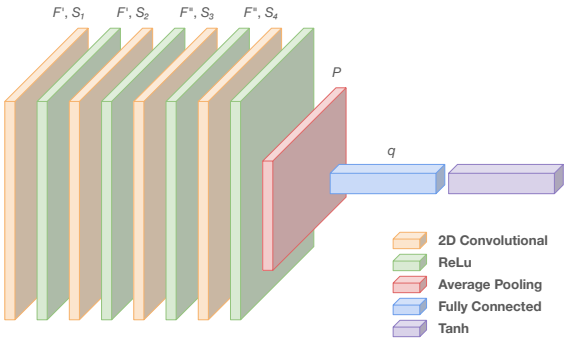


FIGURE 6: CNN model's structure graphical representation.

The best performing RNN hyperparameters are reported in Table 2a, as a result of a grid-search.

C. CNN MODEL

The second DL architecture is based on CNNs. As with RNNs, different network configurations have been tested, finally leading to the choice of the structure represented in Figure 6. To understand the strength of this architecture, each input feature can be thought of as an image, where each value corresponds to a pixel. As a processed input goes deeper in the network, higher-level abstractions of it are extracted and nonlinearly correlated through the activation functions. In this case, the advantages consist of a significantly lower number of parameters to be trained and in a natural mitigation of the vanishing gradient problem affecting the RNN architectures. In detail, the input layer is followed by four two-dimensional

convolutional layers, each accompanied by a Rectified Linear Unit (ReLU) activation layer, with a non-decreasing number of filters and different ad hoc kernel sizes. Leaky ReLU activation functions are chosen to solve the well-known problem of trained weights that permanently prevent a set of neurons from being activated, caused by ReLU functions' zero slope for negative x-axis values.

The first two convolutional layers contain F' filters, respectively of dimension S_1 and S_2, while the third and fourth convolutional layers contain F'' filters of dimension is S_3 and S_4. The filters' stride is equal to one in all the cases (one-step on the horizontal movements, and one-step row change on the vertical movements) since the dimensionality of the problem is quite small and no important information shall be lost. After that, a two-dimensional average pooling layer provides a valuable summary of the extracted information along the temporal axis, thanks to the pooling size $P = [2, 1]$. Finally, like in the RNNs, a flattening layer is inserted, followed by a final dense layer containing q neurons with hyperbolic tangent activation to obtain the prediction. Note that, in the CNNs' case, the order of the features is rearranged as follows: RB_{down}, RB_{up}, $RNTI_{count}$, MCS_{down}, and MCS_{up}. The reason is to allow the network to exploit the a-priori uncorrelation of $RNTI_{count}$ with all the other variables, given the local approach of kernels among the features in CNN models.

The best performing CNN hyperparameters resulting from a grid-search are reported in Table 2b. The resulting best value of $S_1 = [16, 3]$ implies that a correlation of a higher number of variables from the first hidden layer can help the model to perform better. In particular, given the filters' size and the unitary stride, the $RNTI_{count}$ variable is the only one to get directly involved with all the other features in the CNN convolutions since the layer closest to the input of the network. $S_2 = [3, 5]$, being the best choice for the second layer filters dimension, indicates that correlating all the five input variables three time steps at a time is the best trade-off to maximize the extracted information. Furthermore, $F'' = 32$, as the ideal number of filters for both the third and the fourth layers, suggests that too many filters are not needed when their size is chosen properly, with $S_3 = [8, 3]$ and $S_4 = [4, 3]$ in the described case. Finally, $P = [2, 1]$ confirms that the average pooling layer is useful to provide a summary of the convolutional layers' output along the temporal axis. However, it also reveals that it only needs to involve two rows to retrieve the relevant insights extracted from the internal layers and provide a reliable prediction.

D. DEEP TRANSFER LEARNING

The adopted DTL method is based on network-based deep transfer learning, which reuses part of the network that has been pre-trained in the source domain as part of the deep neural network used in the target domain [42]. In particular, the *frozen layers* approach is adopted, which consists of inhibiting the update of the weights of the neurons placed in different combinations of layers, and thus training the remaining part of the networks on the new data [8]. This approach is grounded on the assumption that, for both RNNs and CNNs, each layer extracts meaningful features from the output of the previous one. This implies that groups of neurons that are close (considering the adopted metric on the network graph) should be specialized in extracting similar types of information.

The idea is to perform new training only on the part of the network that is responsible for extracting the features devoted to describing the differences among the datasets. Thus, the common general information already obtained by the source model (and maintained by the frozen neurons) is preserved, while the model is adapting the prediction to the new input distribution. For example, in CNN architectures, the CNN layers closer to the input are devoted to extracting features from the given dataset and can be frozen in DTL. Alternatively, the fully connected layers before the output are responsible for classification and, thus, need to be trained with the target data.

A similar DTL solution is represented by *fine-tuning*, where a pre-trained model is trained with the target data. Despite its effectiveness in DTL methods for many tasks and datasets in various fields, it still represents a computationally expensive solution, since it needs to train the whole ML network with the student's data. Instead, the *frozen layers* approach limits such an issue by training only a subset of the ML network.

VI. EXPERIMENTS AND RESULTS

The following section details the characteristics of the experiments performed and the numerical results of the different models. For the sake of clarity, the section is divided into five subsections. In Section VI-A the model training setup is detailed. Section VI-B reports the performance of the stand-alone models on the PS, EB, and LC datasets, respectively. The usage of DTL and its performance is detailed in Section VI-C. Model complexity and the energy consumption of the studied models are discussed in Section VI-D. Finally, Section VI-E contains the comparison with the benchmark SVR models.

A. MODEL TRAINING SETUP

Each dataset has been split into a training $\mathcal{D}_{t,i}$ and a validation set $\mathcal{D}_{v,i}$, each of them, in turn, divided into two additional different proportions ($i \in [1, 2]$) to study the best threshold between the size of the training set $|\mathcal{D}_t|$ and the performance of each model. In particular, $\mathcal{D}^{\text{PS}}_{t,1}$ and $\mathcal{D}^{\text{PS}}_{t,2}$ contain respectively fourteen and twenty-one days of data in PS. $\mathcal{D}^{\text{EB}}_{t,1}$ and $\mathcal{D}^{\text{LC}}_{t,1}$ contain, instead, five days of data, whereas $\mathcal{D}^{\text{EB}}_{t,2}$ and $\mathcal{D}^{\text{LC}}_{t,2}$ cover a six-day time period. No significant difference was found between the two setups, besides a more consistent convergence of the loss function when training the models on the biggest training set. The results will hence be reported for $\mathcal{D}_{t,2}$ and $\mathcal{D}_{v,2}$ only, referred to as $\mathcal{D}_t$ and $\mathcal{D}_v$ in the rest of this work.

The models were trained and validated using an Ubuntu 18.04 High-Performance Cluster, with 2 Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz, 187 GB of RAM, and four

n_1	n_2	n_3	n_4	D'	D''
128	64	32	16	0	0.2

(a) RNN

F'	F''	S_1	S_2	S_3	S_4	P
16	32	$[16, 3]$	$[3, 5]$	$[8, 3]$	$[4, 3]$	$[2, 1]$

(b) CNN

TABLE 2: DL models' optimized hyperparameters.

NVIDIA GeForce RTX 2080 Ti GPUs. The DL algorithms were implemented in Python, using the `Keras` library on top of the `Tensorflow` backend. For the benchmark SVR, the popular `Scikit-Learn` library was used.

The regression performance was assessed by looking at the Mean Squared Error (MSE) between the true validation data and the predictions, divided by the cardinality of the validation set $|\mathcal{D}_v|$. This allows to fairly compare models, which have been trained on different portions of the same dataset. To achieve reliable results, the experiments on each model were run three times, and the resulting output metrics were averaged.

For the number of training epochs, a unique upper-bound was adopted for all cases, regardless of the trained structure and the dataset. This choice aims to reduce the manual intervention in the training process, avoiding setting a customized hyperparameter for each case. The number of epochs was set to 30, also in the DTL framework, where the needed epochs are even fewer. Average values for the actual necessary number of training epochs will be reported in Section VI-D. Source code for the proposed strategies, along with the experiments' configuration and the code to reproduce the experiments is available online[1].

B. STAND-ALONE MODELS

As a first step of the analysis, RNN and CNN models are trained on the data collected from the three locations in Barcelona (PS, EB, and LC). The results in terms of MSE for the datasets $\mathcal{D}_t^{\text{PS}}$, $\mathcal{D}_t^{\text{EB}}$, and $\mathcal{D}_t^{\text{LC}}$ are reported in Table 3, Table 4 and Table 5, respectively. The results are detailed by varying the input window of observation p in the columns and the time interval from the input and the output tensor Δn in the rows. The lowest MSE between the CNN and RNN models performing the same task has been highlighted in yellow to improve the readability of the tables.

In most cases, the MSE increases with Δn, as expected. For example, the MSE for $\Delta n = 14$ doubles that of $\Delta n = 0$ with datasets $\mathcal{D}_t^{\text{PS}}$ and $\mathcal{D}_t^{\text{LC}}$. Interestingly, RNNs in general perform better with small values of Δn, whereas CNNs outperform for higher values. This phenomenon can be motivated by CNNs being able to better represent the temporal structure of input signals for longer time horizons, which leads to better predictions for higher Δn.

Another noticeable trend is that MSE values are stable with p for both RNNs and CNNs. This result suggests that information temporally distant from the models' output does not contribute to improving prediction accuracy, which can therefore be achieved using only the past 20 minutes ($p = 20$). This property will be confirmed by the XAI tools in Section VII.

[1] https://gitlab.cttc.es/supercom/traffic-prediction

C. DTL MODELS

For this analysis, the larger PS dataset is considered as the teacher, who transfers the knowledge obtained in the stand-alone model to the other two (student) datasets (EB and LC). The approach in [42] is followed, which consists of building the new (student) model by selecting a subset of layers of the teacher model (those closer to the input) as trained on the teacher dataset (an operation called *layer freezing*). The remaining layers, closer to the output, are re-trained based on the new (student) data. The rationale is based on the assumption that each layer is able to extract meaningful and increasingly specific features. Therefore, layers closer to the output are supposed to be those extracting features for that specific learning task, namely the traffic prediction of the student in the new location. Consequently, in this work, the model behavior is studied when re-training part of the layers close to the output. In particular, all the possible combinations in re-training each of the last three layers of the two DL models are considered. One of these combinations also includes the teacher model trained with the PS dataset (i.e., the student model with all frozen layers from the teacher). This instance is an extreme example of knowledge transfer without any extra training cost.

Table 6 shows the lowest MSE of the models among the different freezing combinations using DTL on EB data, using $\mathcal{D}_t^{\text{PS}}$ to train the teacher model. Table 7 provides the same results for the LC dataset. The blue color highlights which of the two considered DL models returns the lowest MSE.

The trends observed for the stand-alone models with respect to Δn and p are maintained. However, CNN models show higher accuracy in twice as many cases as RNNs. Specifically, CNN and RNN models perform equally well in 6 cases, while CNN models outperform in 13 cases and RNN models in 5. In fact, GRU models extract task-specific features already from the first layers. Therefore, GRU re-trained layers are not able to adapt to the change of domain as accurately as CNNs. This behavior could depend on CNN-based models holding a more hierarchical knowledge distillation architecture, in which the deepest layers are specialized for the specific prediction task to be performed.

Interestingly, no significant difference can be identified when applying DTL from a teacher model using $\mathcal{D}_{t,1}^{\text{PS}}$ or $\mathcal{D}_{t,2}^{\text{PS}}$. This result suggests that fourteen days of data are enough to train the ML model for extracting the traffic patterns and

$\Delta n \backslash p$	10		15		20		
0	0.0091	0.0088	0.0087	0.0089	0.0088	0.0090	MSE
4	0.0153	0.0156	0.0149	0.0148	0.0151	0.0155	MSE
9	0.0175	0.0176	0.0183	0.0174	0.0184	0.0180	MSE
14	0.0197	0.0196	0.0193	0.0193	0.0198	0.0192	MSE
	RNN	CNN	RNN	CNN	RNN	CNN	

TABLE 3: MSE of the stand-alone models trained on $\mathcal{D}_t^{\text{PS}}$.

$\Delta n \backslash p$	10		15		20		
0	0.0062	0.0065	0.0062	0.0070	0.0063	0.0068	MSE
4	0.0083	0.0079	0.0081	0.0081	0.0084	0.0076	MSE
9	0.0081	0.0079	0.0081	0.0088	0.0078	0.0087	MSE
14	0.0090	0.0089	0.0099	0.0092	0.0089	0.0090	MSE
	RNN	CNN	RNN	CNN	RNN	CNN	

TABLE 4: MSE of the stand-alone models trained on $\mathcal{D}_t^{\text{EB}}$.

$\Delta n \backslash p$	10		15		20		
0	0.0089	0.0089	0.0086	0.0090	0.0089	0.0089	MSE
4	0.0144	0.0141	0.0136	0.0141	0.0137	0.0141	MSE
9	0.0164	0.0164	0.0166	0.0159	0.0161	0.0165	MSE
14	0.0187	0.0184	0.0187	0.0183	0.0190	0.0190	MSE
	RNN	CNN	RNN	CNN	RNN	CNN	

TABLE 5: MSE of the stand-alone models trained on $\mathcal{D}_t^{\text{LC}}$.

$\Delta n \backslash p$	10		15		20		
0	0.0060	0.0060	0.0059	0.0060	0.0060	0.0059	MSE
4	0.0077	0.0079	0.0078	0.0074	0.0077	0.0078	MSE
9	0.0081	0.0078	0.0085	0.0079	0.0080	0.0082	MSE
14	0.0092	0.0089	0.0096	0.0088	0.0096	0.0091	MSE
	RNN (T)	CNN (T)	RNN (T)	CNN (T)	RNN (T)	CNN (T)	

TABLE 6: MSE of the best transferred models for each of the considered cases, trained on $\mathcal{D}_t^{\text{EB}}$.

being able to transfer them to a student model.

Finally, DTL enables an accuracy improvement in more than 85% of the studied cases (41 out of 48) with respect to the stand-alone counterpart. This result confirms that knowledge transfer is a valid method to accurately train models when available data is scarce (as in the case of EB and LC).

D. COMPLEXITY ANALYSIS AND ENERGY COST

The computational complexity of the proposed solutions depends on four factors: the dataset size, the model structure, the number of training epochs, and the number of trainable parameters in the model. As a proxy for the complexity, an analysis based on those factors has been conducted, following the Green AI principles [5]. In particular, the average training time of a single model among the different values of Δn is provided to offer a more reliable estimation of the required resources, as the temporal distance of the prediction from the input instances does not affect the training time. Moreover, the drained total training energy is calculated using the online tool Green Algorithms [43]. Results are reported in Table 8 for both stand-alone and DTL cases and using $\mathcal{D}_t^{\text{EB}}$.

RNN models contain a total of $100\,981$ parameters. Instead, CNN-based models have a number of parameters varying with p, namely $33\,285$, $34\,885$ or $37\,285$, with p equal to 10, 15 or 20, respectively. In fact, for CNNs with fixed filter sizes in each layer, the number of input values to the final dense layer depends on the number of the filters' strides. This is reflected in a linear increase in the number of weights for each output neuron. Complexity metrics vary very little for CNNs as a function of p, so the table contains only values for $p = 10$. The number of parameters for CNN-based stand-alone models is roughly one-third that of the corresponding RNNs. This highlights the intrinsic advantage of the convolutional layers, which is reflected in the reduced number of training epochs to reach the minimum loss and the training time that is 10 times lower than RNN. In turn, this difference impacts the energy footprint, which is 10 times lower for CNNs.

Considering the DTL models, RNNs need a smaller number of epochs to reach the loss minimum. However, CNN models require less than half of the RNNs training time. This translates into a lower energy footprint of CNNs, which is

$\Delta n \backslash p$	10		15		20		
0	0.0084	0.0084	0.0085	0.0085	0.0085	0.0085	MSE
4	0.0136	0.0136	0.0135	0.0134	0.0135	0.0135	MSE
9	0.0162	0.0164	0.0163	0.0160	0.0161	0.0160	MSE
14	0.0183	0.0180	0.0184	0.0181	0.0181	0.0180	MSE
	RNN (T)	CNN (T)	RNN (T)	CNN (T)	RNN (T)	CNN (T)	

TABLE 7: MSE of the best transferred models for each of the considered cases, trained on $\mathcal{D}_t^{\mathrm{LC}}$.

TABLE 8: Model training complexity and energy consumption for both stand-alone and DTL models, trained on $\mathcal{D}_t^{\mathrm{EB}}$, $p = 10$.

	Stand-alone		DTL	
	RNN	CNN	RNN	CNN
N. of parameters	100 981	33 285	9 449.25	13 321.25
N. of training epochs	13	9	6	8
Training time [s]	74.409	3.908	3.804	1.586
Energy drained [Wh]	19.8	1.0	1.0	0.4
% of saved energy by DTL	-	-	94.95	60.00

40% with respect to RNNs, as with stand-alone models. It can be argued that the energy footprint figures are tightly related to the training time, as experienced both for DTL and stand-alone; thus, CNNs represent the most energy-efficient model. On the other hand, the gain of DTL in RNN with respect to stand-alone models is 95%, whereas the CNN architectures reach 60%, as detailed in Table 8. Consequently, DTL has been able to reduce energy usage more for the ML architecture that presents the highest complexity (RNN).

Concluding, in addition to substantially decreasing the used energy, DTL also enables an accuracy improvement in 41 out of 48 (85%) studied cases with respect to the stand-alone counterpart.

E. COMPARISON WITH BENCHMARK

In this subsection, the proposed RNN and CNN-based models are compared with a benchmark based on SVR [44]. A transformation based on the Radial Basis Function (RBF) kernel [45] is applied to handle the nonlinearity of the datasets. The best-performing hyperparameters are considered, which are the result of a grid search carried out considering several values of the margin width ϵ and the regularization parameter C.

Table 9 contains the results of the SVR model in terms of MSE as a function of Δn and p. For PS, which is used for stand-alone models only, the corresponding entries are highlighted in orange where the SVR model performs better, in green when performing worse, and left blank (no color) when the performance is the same. For the EB and LC, used for both stand-alone and DTL models, a different color mapping is adopted. A red-filled cell indicates better MSE of SVR compared to both the stand-alone and DTL models; a gray-filled cell means an MSE equal to the DTL model, but better than the stand-alone; a blank cell implies an MSE equal to the stand-alone model, but worse than the DTL; a green-filled cell indicates an MSE worse than both the stand-alone and DTL models.

SVR performs better for low values of p and Δn. This is due to the fact that in such cases the prediction task is easier, since the amount of input data is the smallest among all the studied cases and the prediction is closer in time to the input frame. This can be spotted by looking at the high concentration of green-filled cells for Δn equal to 9 and 14 and orange-filled cells for Δn equal to 0. Similarly, SVR performance is often comparable to or better than DL models in EB and LC due to the smaller cardinality of these training sets. This is highlighted in the table by the high number of orange and red-filled cells in the columns of EB and LC.

Therefore, SVR achieves higher accuracy than DL in 9 cases whereas DL outperforms SVR in 14 cases out of a total of 39 (with both achieving the same accuracy in 16 cases). However, DL models outperform SVR when i) the cardinality of the training set is larger, ii) the dimensionality of the input data increases (p) and, iii) the prediction is distant from the input (Δn). This represents a limit in the generalization properties of SVR applied to big data problems for mobile traffic prediction tasks, as the opposite of the proposed DL models, which scale better in such scenarios.

Moreover, SVR presents significant limitations in the applicability of DTL techniques such as layer freezing. In the literature, approaches like Transfer Component Analysis (TCA) [46] can be applied, though they rely on jointly processing the teacher and student datasets. Thus, using SVR does not allow the separation of the training phase into two (teacher and student learning).

VII. MODELS EXPLAINABILITY

Two fairly known XAI algorithms are tailored, namely SmoothGrad [13] and LRP [14], to provide interpretable insights into the introduced RNN and CNN-based models. The

	Poble Sec			El Born			Les Corts			
$\Delta n \backslash p$	10	15	20	10	15	20	10	15	20	
0	0.0086	0.0087	0.0087	0.0061	0.0062	0.0063	0.0085	0.0086	0.0086	MSE
4	0.0153	0.0152	0.0153	0.0075	0.0077	0.0076	0.0138	0.0137	0.0137	MSE
9	0.0181	0.0181	0.0181	0.0080	0.0080	0.0081	0.0164	0.0163	0.0164	MSE
14	0.0202	0.0202	0.0204	0.0087	0.0088	0.0091	0.0184	0.0186	0.0189	MSE

TABLE 9: MSE of the benchmark SVR models for each of the considered cases, trained on $\mathcal{D}_t^{\text{PS}}$, $\mathcal{D}_t$,$^{\text{EB}}$ and $\mathcal{D}_t^{\text{LC}}$.

motivation behind this choice is the need for solutions that provide an explanation based on the whole dataset, rather than being based on single samples. For this reason, other popular XAI techniques such as Local Interpretable Model-agnostic Explanations (LIME) [47] are disregarded. Similarly, SHapley Addictive exPlanations (SHAP) [48] is not contemplated, as it would not allow us to compare CNN results to RNN due to the difference in calculating their Shapley coefficients.

Model interpretation through SmoothGrad rests upon the definition of the sensitivity maps $M_i(\boldsymbol{x}(n))$, which aims to differentiate the prediction function $S_i(\boldsymbol{x}(n))$ for each output feature $i \in [1,\dots,q]$ with respect to the input $\boldsymbol{x}(n)$. The algorithm proposed in [13] is adopted, in which the authors modified its standard calculation to compensate for the rapid fluctuations of the computed partial derivatives. The final sensitivity map results from the average of N sensitivity maps $M_i(\boldsymbol{x}(n))$ computed on different perturbed versions of $\boldsymbol{x}(n)$ through Gaussian kernels, and in detail:

$$\hat{M}_i(\boldsymbol{x}(n)) = \frac{1}{N}\sum_{k=1}^{N} M_i\left(\boldsymbol{x}(n)+\epsilon_k\right) \text{ with } i \in [1,\dots,q]. \quad (1)$$

where $(\epsilon_1,\dots,\epsilon_N)$ are sampled from a Gaussian random variable $\mathcal{N}\left(0,\sigma^2\right)$, and σ measures the intensity of the perturbation.

Instead, LRP defines the metric relevance, which is calculated in an iterative fashion from the output to the input neurons. The idea is to start with a neuron k representing the output layer and calculate the relevance value of the neuron at the previous layer j using this equation:

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \quad (2)$$

where a_j denotes the activation of the neuron j, and w_{jk} is the weight between the two neurons i and j. This formula is used iteratively to compute the relevance R for every neuron of the previous layer. The process is then repeated layer by layer until reaching the input. The obtained final metric (from output to input layers) shows which input features are most relevant to the output.

The two XAI algorithms have been implemented using the `iNNvestigate` Python library [49].

A. EXPLAINABILITY ANALYSIS

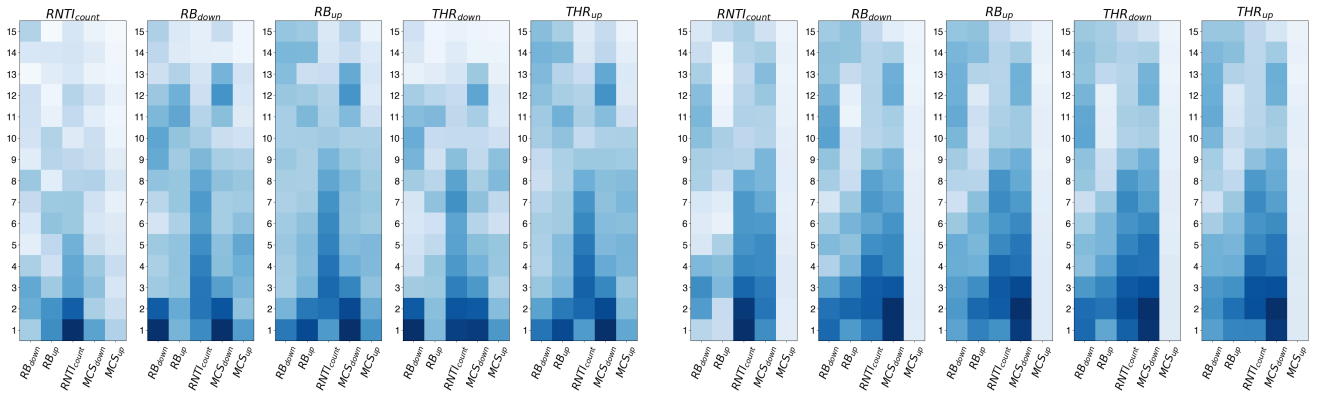
It is to be noted that both SmoothGrad and LRP are meant to be applied to a single specific input sample. Instead, the aim here is to interpret the behavior of the models on the whole dataset. Therefore, the whole $\mathcal{D}_t^{\text{PS}}$ dataset is used to obtain the correspondent sensitivity and relevance values. For the sake of visibility, the next figures show the squared average of the resulting values and highlight the input features that play a key role in the prediction across the studied dataset, independently of the sign of their contribution. In line with the XAI paradigm, all the processed results have been plotted in groups of five heatmaps, each one corresponding to one of the q output labels, i.e., $RNTI_{count}$, RB_{down}, RB_{up}, THR_{down}, and THR_{up}. The x-axis contains the m input variables, and the y-axis shows the progressive temporal index of the p input observations. Sensitivity and relevance are reported in the table with a color format. Darker shades represent features with higher values, i.e., more important in making predictions.

Fig. 7a and Fig. 7b contain the visual representation of the output heatmaps of the SmoothGrad and LRP algorithms, respectively, applied to the CNN model trained on $\mathcal{D}_t^{\text{PS}}$ with a BS of 256 and with $\Delta n = 0$. Fig. 7a indicates that the $RNTI_{count}$ feature is the most informative for predicting $RNTI_{count}$. Instead, RB_{down} and THR_{down} predictions depend strongly on RB_{down} and MCS_{down}; RB_{up} and THR_{up} predictions are mainly influenced by MCS_{down}, RB_{up} and $RNTI_{count}$. This means that forecasting $RNTI_{count}$ mainly depends on the historical data of $RNTI_{count}$ itself, which is due to the fact that the number of users is the variable with the lowest variance in time, as well as to the small Δn value. In particular, the low variability allows for accurate predictions using the last input observations.

The LRP heatmap in Figure 7b presents similar results to SmoothGrad and confirms the dependencies among input features and predicted labels. Again, it confirms that $RNTI_{count}$ predictions strongly rely on the historical data of the same variable, as noticed with SmoothGrad. Moreover, MCS_{up} appears to be almost irrelevant for the predictions (even less than with SmoothGrad), probably due to the limited traffic in uplink from the captured datasets. The entries in the upper part of the heatmaps (i.e., higher p) result to be less significant. The results in Fig. 7a and Fig. 7b confirm the analysis in Section VI-B, where we discussed that longer observation windows do not increase the prediction accuracy. On the other hand, it emerges that MCS_{down} plays an important role, likely due to its central position in the input tensor and its correlation with the variables lying on the borders, i.e., RB_{down} and MCS_{up}.

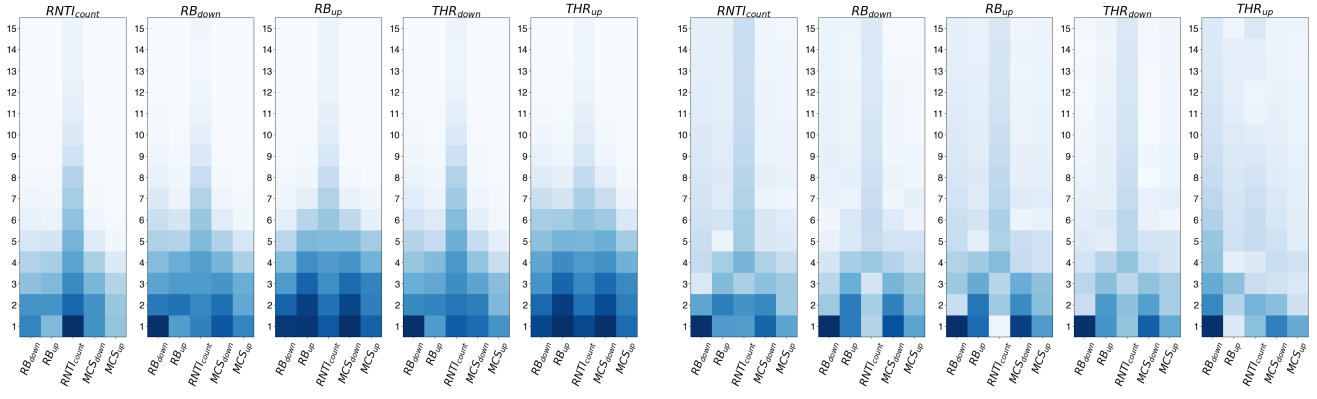
Figure 8a and Figure 8b contain the heatmaps of SmoothGrad and LRP applied to the RNN model trained on $\mathcal{D}_t^{\text{PS}}$

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3518483



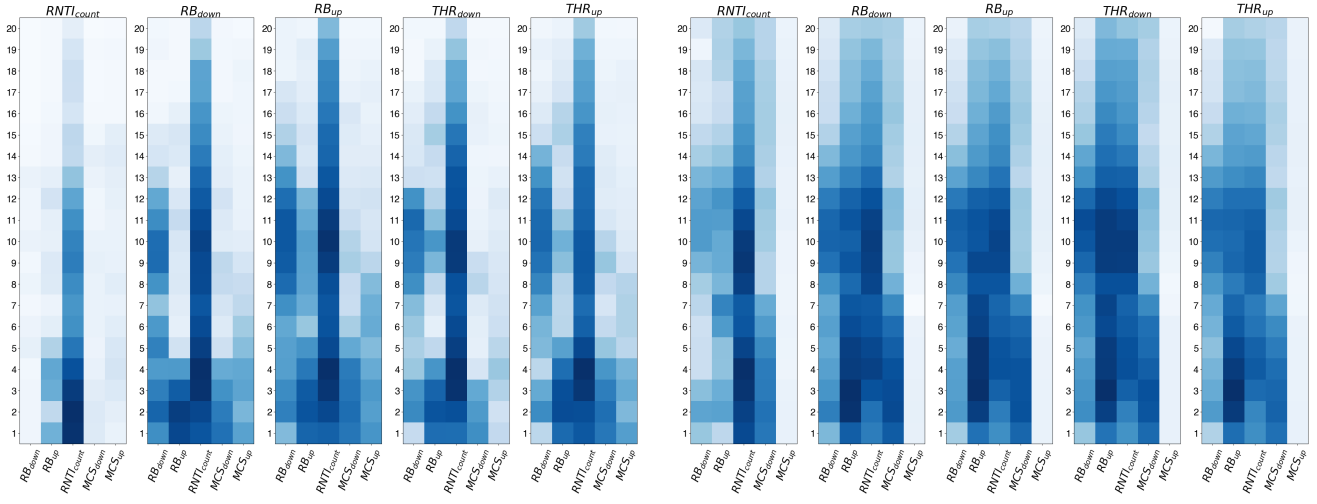
(a) SmoothGrad (b) LRP

FIGURE 7: SmoothGrad and LRP algorithms' heatmaps when applied to the CNN model, $p = 15$, $\Delta n = 0$.



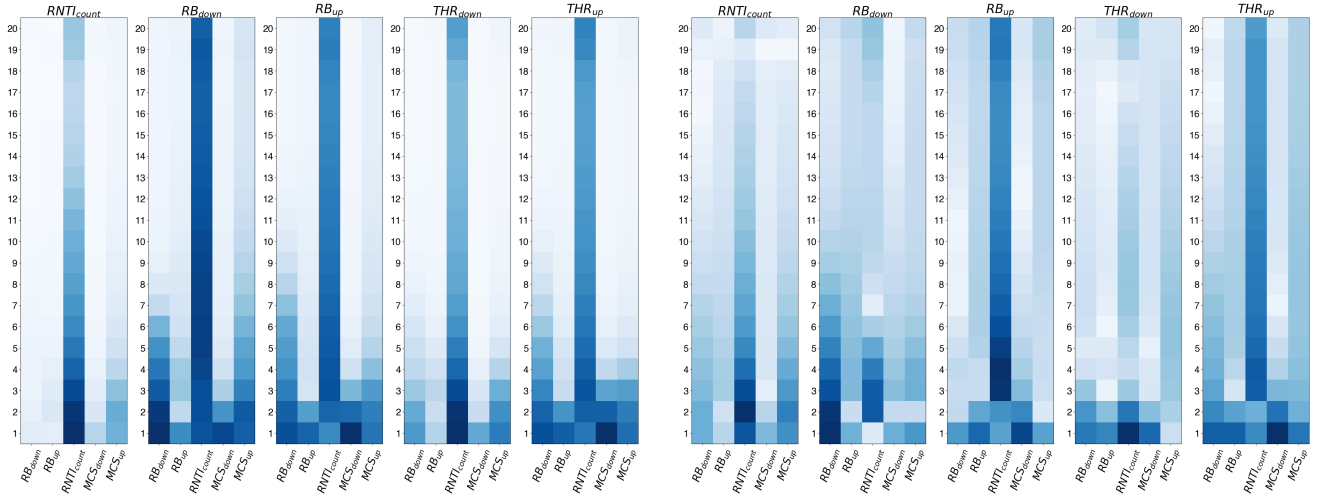
(a) SmoothGrad (b) LRP

FIGURE 8: SmoothGrad and LRP algorithms' heatmaps when applied to the RNN model, $p = 15$, $\Delta n = 0$.



(a) SmoothGrad (b) LRP

FIGURE 9: SmoothGrad and LRP algorithms' heatmaps when applied to the CNN model, $p = 20$, $\Delta n = 14$.



(a) SmoothGrad (b) LRP

FIGURE 10: SmoothGrad and LRP algorithms' heatmaps when applied to the RNN model, $p = 20$, $\Delta n = 14$.

with a BS of 128 and for $\Delta n = 0$. Results indicate a different relationship between input features and predicted labels compared to the CNN model. Figure 8a confirms that the $RNTI_{count}$ feature is the most informative for predicting $RNTI_{count}$. Instead, RB_{down} and THR_{down} predictions depend strongly on RB_{down} and MCS_{down} and $RNTI_{count}$; RB_{up} and THR_{up} predictions are instead influenced by all the input variables. Fig. 8b shows that $RNTI_{count}$ predictions depend mainly on RB_{down}. No other significant differences can be appreciated with respect to Figure 8a. Both figures highlight that longer input observations do not have any influence on the predicted labels for RNN models, with the only exception in the $RNTI_{count}$ case.

In Figure 9 and Figure 10, the analysis on the future predictions Δn is presented, by showing the heatmaps of the two XAI algorithms with CNN and RNN models trained on $\mathcal{D}_t^{\text{PS}}$ with $\Delta n = 14$. In Section VI-B, we discussed that the CNN model generally performs better in tasks where Δn is large, indicating that CNNs can extract much more information from the input samples compared to RNNs. This is clearly visible in Figure 9 graphs, which show darker pixels for higher values of p (y axis) for both SmoothGrad and LRP algorithm.

B. FUTURE RESEARCH DIRECTIONS

Based on the outcomes of XAI analysis, several optimizations can be performed to the DL architectures used here for mobile traffic prediction. Considering RNNs, a straightforward optimization is represented by the reduction of the input observation window (p) for the simplest tasks (i.e., for small Δn), which could lead to lower computational costs without jeopardizing the performance. Regarding more complicated extensions, the combination of multi-task [50] and ensemble model approaches [51] could be considered for creating a meta-model where each output parameter has only the most important input features according to XAI outcomes. For example, RB_{up} could be removed from the base estimators of the global ensemble model corresponding to several predictions where the importance of such an input feature has been shown as negligible by XAI. Similarly, $RNTI_{count}$ can be predicted almost entirely using previous $RNTI_{count}$ values, so simpler models can be used for this task.

On the other hand, XAI algorithms fail to provide a clear interpretation of the neurons' activation and interconnections. We believe this is a key open issue to enhance the black box approach to neural networks. With more information about the internal structure of the model, DTL layer freezing can be optimized towards a better energy-accuracy trade-off. In particular, LRP is suitable by design to extract networks' internal layers information, but the enormous number of neurons complicates the extraction of interpretable observations. To overcome this issue, the general global network can be divided into sub-networks that can be more easily managed with aggregated metrics. This would allow to evaluate relevance on a more fine-grained basis and could be used to detect insights about those sub-networks. This procedure has to be properly designed to find the correct dimension of the sub-network and, thus, avoid the introduction of more computational complexity compared to the one that would be saved by the DTL algorithm.

VIII. CONCLUSIONS

This paper presented a novel edge computing framework for mobile traffic prediction, utilizing three datasets obtained from decoding the unencrypted LTE control channel at the network edge. Two main DL architectures were designed, based on CNNs and RNNs, using both a stand-alone approach and DTL techniques to reduce the required computational resources.

Results show that the CNN architectures outperform the

RNNs in terms of accuracy, and in both cases DTL enables an accuracy improvement in 85% of the studied cases with respect to the stand-alone counterpart. Moreover, DTL is able to significantly decrease the computation complexity and thus the energy consumption relative to training, allowing an energy footprint reduction of 60% and 90% for CNNs and RNNs, respectively. Finally, two cutting-edge XAI techniques were employed to explain DL models' prediction behavior and provide insights into the accuracy results.

The proposed approach faces challenges with Stochastic Gradient Descent optimization, particularly due to variations in the granularity and nature of the diverse inputs and outputs, as well as the significant presence of outliers in the data. This can impact the model's overall performance. To address this limitation, future work could consider adopting multi-task learning or ensemble model approaches to significantly enhance prediction accuracy and reduce energy consumption, offering a more efficient solution.

Based on the XAI outcomes, some possible extensions to this work have been derived. Regarding the architecture, the combination of multi-task and ensemble model approaches can be used for implementing a meta-model in which the prediction of each output parameter uses only its most important input features. For XAI, DTL layer freezing can be optimized by studying extensions of LRP that provide insights into the activation of internal neurons and interpretation of their interconnections. To achieve this, the general global network can be divided into sub-networks that are more easily managed with aggregated metrics.

REFERENCES

[1] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y.-J. A. Zhang, "The roadmap to 6g: Ai empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, 2019.

[2] A. de Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, 2023.

[3] G. Intelligence, "Green is good for business: embedding sustainability in digital transformation (part i)." https://www.gsmaintelligence.com/green-is-good-for-business/, 2023.

[4] G. Luo, Q. Yuan, J. Li, S. Wang, and F. Yang, "Artificial intelligence powered mobile networks: From cognition to decision," *IEEE Network*, vol. 36, no. 3, pp. 136–144, 2022.

[5] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green ai," *Commun. ACM*, vol. 63, pp. 54–63, nov 2020.

[6] D. Thembelihle, M. Rossi, and D. Munaretto, "Softwarization of mobile network functions towards agile and energy efficient 5g architectures: a survey," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[7] E. Ahvar, A.-C. Orgerie, and A. Lebre, "Estimating energy consumption of cloud, fog and edge computing infrastructures," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2019.

[8] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies*, vol. 11, no. 2, p. 40, 2023.

[9] R. Sharma, S. Biookaghazadeh, B. Li, and M. Zhao, "Are existing knowledge transfer techniques effective for deep learning with edge devices?," in *2018 IEEE International Conference on Edge Computing (EDGE)*, pp. 42–49, 2018.

[10] G. Alicioglu and B. Sun, "A survey of visual analytics for explainable artificial intelligence methods," *Computers and Graphics*, 2021. Publisher Copyright: © 2021 Elsevier Ltd.

[11] N. Bui and J. Widmer, "Owl: A reliable online watcher for lte control channel measurements," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ATC '16, (New York, NY, USA), pp. 25–30, Association for Computing Machinery, 2016.

[12] Y. Liu and J. Y. Lee, "An empirical study of throughput prediction in mobile data networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2015.

[13] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *CoRR*, vol. abs/1706.03825, 2017.

[14] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, pp. 1–46, 07 2015.

[15] R. Gupta, D. Reebadiya, and S. Tanwar, "6g-enabled edge intelligence for ultra-reliable low latency applications: Vision and mission," *Computer Standards & Interfaces*, vol. 77, p. 103521, 2021.

[16] S. Parikh, D. Dave, R. Patel, and N. Doshi, "Security and Privacy Issues in Cloud, Fog and Edge Computing," *Procedia Computer Science*, vol. 160, pp. 734–739, 2019.

[17] X. Cheng, Y. Hu, and L. Varga, "5g network deployment and the associated energy consumption in the uk: A complex systems' exploration," *Technological Forecasting and Social Change*, vol. 180, p. 121672, 2022.

[18] Statista, "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025." https://www.statista.com/statistics/871513/worldwide-data-created/, 2023.

[19] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "Ai models for green communications towards 6g," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 210–247, 2022.

[20] 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," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[21] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence." arXiv:1909.00560, 2019.

[22] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1040–1050, 2019.

[23] DOCOMO, "White paper 5g evolution and 6g (version 5.0)." https://www.docomo.ne.jp/english/corporate/technology/whitepaper_6g/, 2023.

[24] A. Pelati, M. Meo, and P. Dini, "Traffic anomaly detection using deep semi-supervised learning at the mobile edge," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8919–8932, 2022.

[25] Y. Yu, J. Wang, M. Song, and J. Song, "Network traffic prediction and result analysis based on seasonal arima and correlation coefficient," in *2010 International Conference on Intelligent System Design and Engineering Application*, vol. 1, pp. 980–983, IEEE, 2010.

[26] A. Biernacki, "Improving quality of adaptive video by traffic prediction with (f) arima models," *Journal of communications and networks*, vol. 19, no. 5, pp. 521–530, 2017.

[27] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1790–1821, 2017.

[28] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.

[29] D. C. Nguyen, P. Cheng, M. Ding, D. Lopez-Perez, P. N. Pathirana, J. Li, A. Seneviratne, Y. Li, and H. V. Poor, "Wireless ai: Enabling an ai-governed data life cycle," *arXiv preprint arXiv:2003.00866*, 2020.

[30] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1656–1659, 2018.

[31] 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 *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp. 1–9, IEEE, 2017.

[32] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[33] C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *2017 IEEE 28th annual interna-*

tional symposium on personal, indoor, and mobile radio communications (PIMRC), pp. 1–6, IEEE, 2017.
[34] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 231–240, 2018.
[35] L. Zhang, C. Zhang, and B. Shihada, "Efficient wireless traffic prediction at the edge: A federated meta-learning approach," *IEEE Communications Letters*, pp. 1–1, 2022.
[36] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, "Federated learning for 5g base station traffic forecasting," *Computer Networks*, vol. 235, p. 109950, 2023.
[37] 3GPP, *Tech. Specif. Group Radio Access Network; Physical layer procedures (Release 9), 3GPP TS 36.213*.
[38] H. D. Trinh, E. Zeydan, L. Giupponi, and P. Dini, "Detecting mobile traffic anomalies through physical control channel fingerprinting: A deep semi-supervised approach," *IEEE Access*, vol. 7, pp. 152187–152201, 2019.
[39] A. Pelati, M. Meo, and P. Dini, "Traffic anomaly detection using deep semi-supervised learning at the mobile edge," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8919–8932, 2022.
[40] R. Cahuantzi, X. Chen, and S. Güttel, "A comparison of lstm and gru networks for learning symbolic sequences," 2021.
[41] T. Bluche, C. Kermorvant, and J. Louradour, "Where to apply dropout in recurrent neural networks for handwriting recognition?," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 681–685, 2015.
[42] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, Springer, 2018.
[43] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: Quantifying the carbon footprint of computation," *Advanced Science*, vol. 8, no. 12, p. 2100707, 2021.
[44] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, V. Vapnik, *et al.*, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155–161, 1997.
[45] W. Wang, Z. Xu, W. Lu, and X. Zhang, "Determination of the spread parameter in the gaussian kernel for classification and regression," *Neurocomputing*, vol. 55, no. 3, pp. 643–663, 2003. Evolving Solution with Neural Networks.
[46] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.
[47] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.
[48] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.
[49] M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, and P.-J. Kindermans, "innvestigate neural networks!," *J. Mach. Learn. Res.*, vol. 20, no. 93, pp. 1–8, 2019.
[50] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2022.
[51] Y. Ren, L. Zhang, and P. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions [review article]," *IEEE Computational Intelligence Magazine*, vol. 11, no. 1, pp. 41–53, 2016.

ALFREDO PETRELLA received the B.Sc. degree in mathematics and the M.Sc. degree in data science from the University of Padua, Italy, in 2018 and 2021, respectively. In 2021, he was a Research Assistant in the Sustainable Artificial Intelligence research unit at CTTC. Since 2022, he has been working as a consultant in Artificial Intelligence and Machine Learning in Milan, Italy, being deeply involved in both the theoretical and practical aspects of AI, and applying cutting-edge research to real-world problems. His research interests include optimization models, deep learning, and natural language processing.

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. In June 2008 he joined the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). His main research interests are: sustainable mobile networks, green wireless networking, multi-agent systems, machine learning, green AI, distributed and collaborative learning, and pervasive AI.

PAOLO DINI received M.Sc. and Ph.D. from the University of Roma La Sapienza, in 2001 and 2005, respectively. He is currently a Leading Researcher with the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC) and coordinates the activities of the Sustainable AI research unit. His current research interests include sustainable computing for cyber-physical systems, distributed optimization and optimal control, machine learning, multi-agent systems. He received two awards from the Cisco Silicon Valley Foundation for his research on distributed control over heterogeneous mobile networks, in 2008 and 2011, respectively. He has been involved in more than 20 research and development projects. He is currently the Coordinator of CHIST-ERA SONATA project on sustainable computing and communication at the edge and the Scientific Coordinator of the EU H2020 MSCA Greenedge European Training Network on edge intelligence and sustainable computing.