Multi-Task Learning at the Mobile Edge: an Effective Way to Combine Traffic Classification and Prediction

Multi-Task Learning at the Mobile Edge: an Effective Way to Combine Traffic Classification and Prediction

Arcangela Rago, *Student Member, IEEE*, Giuseppe Piro, *Member, IEEE*, Gennaro Boggia, *Senior Member, IEEE*, and Paolo Dini

***Abstract*—Mobile traffic classification and prediction are key tasks for network optimization. Most of the works in this area present two main drawbacks. First, they treat the two tasks separately, thus requiring high computational capabilities. Second, they perform data mining on the information collected from the data plane, which is unsuitable for the mobile edge. To bridge this gap, this paper properly tailors a Multi-Task Learning model running directly at the edge of the network to anticipate information on the type of traffic to be served and the resource allocation pattern requested by each service during its execution. Our study exploits data mining from the control channel of an operative mobile network to also reduce storage and monitoring processing. Different configurations of neural networks, which adopt autoencoders (i.e. Undercomplete Autoencoder or Sequence to Sequence Autoencoder) as key building blocks of the proposed Multi-Task Learning methodology for common feature representations, are investigated to evaluate the impact of the observation window of traffic profiles on the classification accuracy, prediction loss, complexity, and convergence. The comparison with respect to conventional single-task learning approaches, that do not use autoencoders and tackle classification and prediction tasks separately, clearly demonstrates the effectiveness of the proposed Multi-Task Learning approach under different system configurations.**

***Index Terms*—Machine Learning, Mobile Data, Deep Learning, Traffic Classification, Traffic Prediction**

I. Introduction

Machine Learning (ML) is the branch of Artificial Intelligence (AI) that investigates algorithms able to learn and improve their experience and performance over time directly from data examples, without being explicitly programmed. With these algorithms, a system can scrutinize data and deduce knowledge: hidden patterns in the training data are identified and used to analyze unknown information and drive the execution of a given task (typically classification, prediction, or clustering) [1]. To improve these capabilities, deep learning further enables the mining of valuable information of data coming from heterogeneous sources and unveils hidden correlations automatically, which would have been too complex to extract by human experts [2]. Recently, ML-based solutions have been applied to the mobile networking domain [3], where the growing diversity and complexity of the mobile network architectures made the monitoring and the managing of the multitude of network elements intractable [4], [5]. At the same time, networking researchers have been recognizing the importance of deep learning and its ability to solve specific problems in current and future generations of mobile systems [2], [6], [7].

In line with this emerging research trend, we investigate in this paper the potential of deep learning for mobile traffic classification and prediction, which are key tasks for network optimization. In fact, the envisaged architecture of the fifth generation (5G) of mobile broadband systems will integrate new technology components (e.g., massive MIMO, mm-Wave communication, network slicing, vehicular networks, more and broader frequency bands), a higher variety of devices (e.g., smartphone, sensors, and different types of machines), a larger number of services (typical broadband services, as well as some advanced applications such as extended reality and automated driving) with tighter latency requirements, so that resource allocation is expected to reach unprecedented complexity [8]–[10]. In this context, network optimization frameworks may be supported by deep learning algorithms, which, when properly tailored, may anticipate information on: i) the type of traffic to be served, e.g. its main characteristics in terms of bandwidth and latency requirements (i.e. *traffic classification*) and ii) the resource allocation pattern requested by each service along its duration (i.e. *traffic prediction*).

Most of the literature in this field treat traffic classification and prediction separately [11]–[22] (please see Section II for further details). Instead, we propose a Multi-Task Learning (MTL) approach [23], which reduces the number of training samples to be learnt by the two tasks and leads to performance improvement compared with learning them individually [24].

At the same time, it is important to remark that offloading the huge amount of data generated from edge to cloud is

A. Rago, G. Piro, and G. Boggia are with the Department of Electrical and Information Engineering (DEI), Politecnico di Bari, Italy, and with Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT); e-mail: {arcangela.rago, giuseppe.piro, gennaro.boggia}@poliba.it.

P. Dini is with Centre Tecnologic de Telecomunicacions de Catalunya (CTTC/CERCA), Barcelona, Spain; e-mail: paolo.dini@cttc.es.

This work was supported by the PRIN project no. 2017NS9FEY entitled "Realtime Control of 5G Wireless Networks: Taming the Complexity of Future Transmission and Computation Challenges" funded by the Italian MIUR, by the Apulia Region (Italy) Research project INTENTO (36A49H6), by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 675891 (SCAVENGE), and by Spanish MINECO grant TEC2017-88373-R (5G-REFINE). It has been also partially supported by the Italian MIUR PON projects Pico&Pro (ARS01_01061), AGREED (ARS01_00254), FURTHER (ARS01_01283), and RAFAEL (ARS01_00305).

intractable in 5G scenarios since it causes oppressive network congestions. Therefore, it is highly preferable that deep learning algorithms run at the edge of the network and give online support to optimization frameworks to promptly take decisions and trigger the proper management actions (e.g., radio resource scheduling, cell selection, and sleep mode enabling, to name a few) [25]–[27]. Almost all the approaches presented in the current state of the art implement data mining on the huge amount of information collected at the network or application layers of the data plane. Differently, the proposed MTL model considers data belonging to the control plane, as recently investigated, and it is trained with information extracted from the Physical Downlink Control CHannel (PDCCH) of an operative mobile network in Spain. The rationale behind the choice of using the control channel is twofold. First, the volume of control messages from the control plane is much smaller than the user traffic from the data plane (which may also be encrypted), leading to fast and efficient classification and prediction, which are still evaluated on the derived data plane information. Specifically, the classification task registers an accuracy up to 99% and the prediction task ensures a Mean Square Error (MSE) lower than 10^{-3}. Second, the algorithm runs at the radio interface, which allows fast execution of the two tasks directly at the edge.

In summary, the original contributions of this work are:

- *Joint traffic classification and prediction* though a MTL model running at the edge of the network;
- *Data mining from the PDCCH control channel*, which guarantees reduced storage requirements, fast data processing, and limited monitoring complexity;
- *Use of autoencoders as key building blocks* of the proposed MTL methodology for common feature representations, shared by both classification and prediction tasks. Specifically, Undercomplete and Sequence to Sequence (Seq2Seq) architectures are tailored for our scenario and their performance are compared;
- *Comparison with conventional single-task learning* approaches for traffic classification and prediction, that do not use autoencoders and tackle classification and prediction tasks separately.

The remainder of the paper is as follows. In Section II we introduce the related work on this area and identify the gaps, which we intend to fill with this paper. Section III is dedicated to the proposed MTL approach, including the design criteria and the data processing for training. In Section IV we analyze and compare the performance achieved by single-task models for traffic classification and prediction used as benchmarks. Finally, Section V concludes the paper and draws future research activities.

II. State of the art

As already anticipated in the Introduction, ML has been recently applied to the mobile networking domain [3]. Possible applications include radio access technology selection [34], malware detection [35], development of networked systems [36], energy saving [37], panoramic video streaming [38], and cloudlets activation for scalable Mobile Edge Computing [39]. Several approaches, based on Support Vector Machine and Random Forest algorithms, have been also conceived to identify applications or smartphone types starting from the observation of encrypted communication flows [40]–[43]. Nevertheless, mobile data are usually generated by heterogeneous sources, exhibit non-trivial spatio/temporal patterns, and often embrace high volumes of different information [44]. Flows' characteristics are also rapidly prone to be out of date and need to be frequently updated [45]. In these complex and dynamics conditions, ML algorithms generally fail to automatically extract and use the key features describing the investigated flows [6]. On the contrary, deep learning methods demonstrated to be able to overcome the traditional ML approaches because of their native ability to successfully support traffic analysis and accurately characterize traffic dynamics [6], [8], [45]–[50]. Unfortunately, mobile networking and deep learning problems have been explored mostly independently and only recently crossovers between the two research areas have emerged.

Reference deep learning solutions for traffic classification leverage Convolutional Neural Networks with one-dimensional [11]–[13] or two-dimensional [13], [14] convolutional layers, Stacked Autoencoder with five stacked layers [12]–[14], Multi-Layer Perceptron (MLP) with one [13] or two hidden layers [13], [14], and standard or hybrid Long Short-Term Memory (LSTM) combined with two-dimensional convolutional layers [13]. However, only [13] focuses on mobile networks. Among the other important investigations it provides, the work [13] also demonstrates how deep neural networks guarantee greater accuracy levels than conventional ML approaches in mobile networks. On the other hand, deep learning also outperforms baseline approaches for traffic prediction, including the conventional Auto Regressive Integrated Moving Average scheme [6]. Here, reference methodologies are based on densely connected Convolutional Neural Networks with two-dimensional convolutional layers [15] and LSTMs [16]–[19], as well as on their combination [20]–[22], that can extract spatial and temporal correlations of data through the convolutional operation and LSTM memory cells, respectively. In that case, all of the reviewed contributions focus on mobile networks.

The analysis of the state of the art on deep learning strategies highlights that traffic classification and prediction are generally treated separately. In other words, classification and prediction are achieved by means of two separate single-tasks. Unfortunately, this represents an important drawback because their parallel execution involves the training of different learning architectures, as well as an inevitable increment of computational requirements [24].

The MTL approach solves the aforementioned issue, while often reaching greater performance levels when compared with single-tasks approaches [24], [51]. Differently from the single-task scheme, MTL basically embraces a learning architecture that extracts common feature representations from the training data and jointly executes multiple, but related, tasks. Therefore, MTL emerges as a suitable solution for meeting the computational and memory constraints affecting mobile networks [6]. Valuable contributions in this direction

TABLE I
COMPARISON AMONG OUR WORK AND THE OTHER CONTRIBUTIONS FOCUSING ON TRAFFIC ANALYSIS THROUGH DEEP LEARNING

Contributions	**Task**			**Mobile scenario**	**Processed messages**		**Dataset type**		
	Classification	Prediction	*Joint*		Data plane	*Control plane*	Network/ application level data	Traffic volume/ load	*Radio link-level data*
[11], [12], [14]	✓				✓		✓	✓	
[13]	✓			✓	✓		✓	✓	
[15]–[22]		✓		✓	✓		✓	✓	
[28], [29]	✓				✓		✓	✓	
[30], [31]	✓			✓		✓		✓	✓
[32]	✓			✓		✓	✓	✓	✓
[33]		✓		✓		✓		✓	✓
Our work			✓	✓		✓	✓	✓	✓

are presented in [28], [29], where a MTL architecture is designed to implement multiple tasks related to the traffic classification only. Unfortunately, they do not address traffic prediction and do not focus on mobile networks.

Another important consideration emerging from the scientific literature is that all the investigated contributions perform data mining from the messages exchanged over the data plane (i.e., traffic volume/load collected at the network or application layers, equipped with application labels for classification task). Therefore, by considering the huge amount of data handled by mobile systems, the reviewed methodologies cannot be applied to the control plane and require high computational and memory capabilities, thus becoming unfeasible for the mobile edge.

The goal of this paper is to adopt a MTL architecture at the edge of the network to jointly classify mobile services and forecast future traffic demands. Our study exploits data mining from the unencrypted control channel of an operative mobile network to properly characterize the mobile traffic at the radio interface, in addition to getting out data plane information (i.e., traffic volume/load and application labels) and reducing storage and monitoring processing. Therefore, even if the data mining is performed on the control plane, the accuracy of the classification and prediction tasks is still evaluated on the derived data plane information. Interesting contributions in this direction address traffic pattern analysis and classification [30]–[32] and traffic prediction [33] through data mining performed on the PDCCH. The proposed solutions, however, are not based on the MTL approach.

In this work, we still pursue the idea that traffic classification and prediction at the radio interface can enable advanced Quality of Service and Quality of Experience enforcement policies based on a priori knowledge of application behaviors. Thus, network operators can configure and manage network resources in a more intelligent and prolific mode thanks to the knowledge extracted by deep learning algorithms. Nevertheless, differently from the current state of the art, and for the best of our knowledge, we formulate a novel methodology that applies MTL to classify and predict mobile traffic at the mobile edge, as we are proposing in this work.

To conclude, Table I summarizes the goals and the methodologies followed by the scientific contributions reviewed in this section, while highlighting the main differences with respect to the MTL model proposed herein.

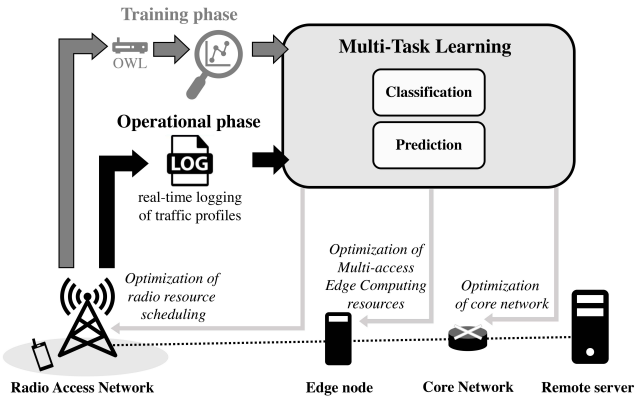


Fig. 1. Input and output of the proposed MTL approach in a mobile network.

III. THE PROPOSED MULTI-TASK LEARNING APPROACH

The developed methodology originates from the consideration that any active session can be described, at the radio link-level, through a traffic profile reporting the amount of data exchanged between the base station and mobile terminal during the time, simply referred to as *radio utilization pattern*. Therefore, by observing such a profile during a time interval T, it could be possible to classify the application type which the investigated session belongs to (task 1) and predict the radio utilization pattern that the session will experience in the upcoming time instants (task 2). This goal is successfully achieved through a MTL architecture running directly at the edge of a mobile network (Fig. 1). Without loss of generality, the contribution directly focuses on the downlink communication. However, the whole approach can be applied to the uplink as well.

To facilitate the understanding of the notations adopted in what follows, a summary of symbols is reported in Table II.

Following these initial considerations, the proposed MTL approach grounds its roots into the *feature learning representation* concept [24], according to which the features for a common representation of our input (i.e. traffic profiles) are extracted and jointly used to execute the two tasks (i.e.,

TABLE II
LIST OF MATHEMATICAL SYMBOLS

Symbol	Description
i	Traffic session index
j	Time instant index
$\boldsymbol{\mathcal{D}}$	Original input matrix with traffic profiles
$\mathbf{d_i}$	Row vector in $\boldsymbol{\mathcal{D}}$ that represents traffic session
$r_{\boldsymbol{\mathcal{D}}}$	Number of rows (traffic sessions) in $\boldsymbol{\mathcal{D}}$
Δ	Number of columns (time instants) in $\boldsymbol{\mathcal{D}}$
$\mathbf{c}$	Column vector of labels associated to $\boldsymbol{\mathcal{D}}$
c_i	Label (i.e. component of $\mathbf{c}$) associated to $\mathbf{d_i}$
T	Observation window
$\boldsymbol{\mathcal{M}}$	Pre-processed input matrix with traffic profiles
$\mathbf{m_i}$	Row vector in $\boldsymbol{\mathcal{M}}$ that represents traffic session lasting T
$m_{i,j}$	Component of $\mathbf{m_i}$ during the j-th time instant
$r_{\boldsymbol{\mathcal{M}},tr}$	Number of rows of $\boldsymbol{\mathcal{M}}$ selected as training set
$\boldsymbol{\mathcal{H}}$	Codeword matrix with feature learning representations of $\boldsymbol{\mathcal{M}}$
$\mathbf{h_i}$	Feature learning representation (i.e. component of $\boldsymbol{\mathcal{H}}$) of $\mathbf{m_i}$
$\hat{\boldsymbol{\mathcal{M}}}$	Reconstructed input matrix
$\hat{\mathbf{m}}_\mathbf{i}$	Reconstructed traffic session in $\hat{\boldsymbol{\mathcal{M}}}$
$\hat{m}_{i,j}$	Reconstructed component of $\hat{\mathbf{m}}_\mathbf{i}$ during the j-th time instant
$\mathbf{l}$	Column vector of labels associated to $\boldsymbol{\mathcal{M}}$
l_i	Label (i.e. component of $\mathbf{l}$) associated to $\mathbf{m_i}$
$\hat{\mathbf{l}}$	Column vector of learned labels associated to $\boldsymbol{\mathcal{M}}$
$\hat{l}_i$	Learned label (i.e. component of $\hat{\mathbf{l}}$) associated to $\mathbf{m_i}$
$\mathbf{m_{T+1}}$	Column vector with data exchanged at $T+1$
$m_{i,T+1}$	Component subsequent to $\mathbf{m_i}$ with data exchanged at $T+1$
$\hat{\mathbf{m}}_\mathbf{T+1}$	Predicted column vector with data exchanged at $T+1$
$\hat{m}_{i,T+1}$	Predicted component at $T+1$ related to $\mathbf{m_i}$
$\mathcal{L}_\mathcal{A}$	Mean Square Error (loss) of the Autoencoder
$\mathcal{L}_\mathcal{C}$	Mean Square Error (loss) of the Classifier
$\mathcal{A}_\mathcal{C}$	Classifier accuracy
$\mathcal{L}_\mathcal{P}$	Mean Square Error (loss) of the Predictor
$\mathcal{P}_{\mathcal{MTL}}$	Multi-objective performance metric for the MTL model

classification and prediction). In particular, the conceived methodology uses an autoencoder to obtain the common feature representations of input data because it can directly accomplish this operation without requiring the knowledge of data distribution nor the explicit identification of a certain structure [49]. Classification and prediction tasks are later executed through softmax and fully-connected layers, respectively. Accordingly, the autoencoder is a key building and enabling block of the proposed MTL methodology, that effectively allows the joint execution of classification and prediction tasks.

As depicted in Fig. 1, the outcomes of the proposed scheme can be exploited to implement advanced methodologies for the management and the optimization of mobile networks. Our approach is conceived to process data directly at the edge, so that the right actions may be triggered faster and locally. Possible strategies that may benefit from the implementation of our architecture range from radio resource scheduling and admission control, mobility management and energy saving mechanisms, to network slicing and dynamic placement of virtualized functions, as well as to the optimization of computing resources at both edge and core network (see Fig. 1). Nevertheless, note that the rest of this Section focuses on the MTL approach and the reference dataset taken into account for training purposes. Any other considerations related to network optimization aspects, however, remain out of the scope of this work and they will be addressed in the future.

A. The training dataset

Being our approach intended to work at the mobile edge, data exchanged through the radio interface are needed for training our model. An operator owing the mobile infrastructure can simply retrieve this information and use it for both the training and operating phases. However, in our case, we use the dataset created in our previous work [32], which consists of traffic traces containing the Downlink Control Information (DCI) messages carried within the PDCCH with a time granularity of 1ms. This information is used by the eNodeB to communicate scheduling information to the connected mobile terminals. DCI messages are unencrypted and be decoded by a specific hardware/software tool called Online Watcher for LTE (OWL) [52]. A key characteristic of the training dataset is that it is gathered from the control channel, which simplifies the monitoring system complexity, assures fast data processing, and reduces the storage capacity due to the limited volume of data.

The captured traces are generated by different applications running in a mobile terminal under our control and attached to an operative mobile network in Spain. Six different applications grouped in three categories have been tested: YouTube and Vimeo for *video-streaming*, Spotify and Google Music for *audio-streaming*, and Skype and WhatsApp Messanger for *video-call*. We selected those applications because they generally produce, according to recent Ericsson [53] and Cisco [54] reports, more than 80% of the mobile data traffic and require optimal resource management due to their strict quality requirements. The proposed approach, however, can be safely applied to other mobile network scenarios with a different set of applications and services, only requiring a new training procedure. Also, after an effective training, our methodology is extendable to any number of classes because it is general and not restricted to a specific use-case (see Section IV-E for more details).

Among the several parameters extracted from the DCI messages, we used the Transport Block Size (TBS), which specifies the length of the packet burst to be sent to/from the considered mobile terminal in the current time slot [55]. Then, TBS values are processed to generate the radio utilization patterns describing the amount of data exchanged between the base station and mobile terminal during the time, with a time granularity of 1s.

Formally, let $r_{\boldsymbol{\mathcal{D}}}$ be the number of traffic sessions collected in a period of time equal to Δ. In this work, $r_{\boldsymbol{\mathcal{D}}} = 11574$ and $\Delta = 60$s. The distribution of the sessions among the considered application categories is reported in Fig. 2. The original training dataset contains a matrix $\boldsymbol{\mathcal{D}}$ and a vector $\mathbf{c}$ of labels. In particular, the original input matrix $\boldsymbol{\mathcal{D}}$ describes the captured traffic profiles (also referred to as the radio utilization patterns) of $r_{\boldsymbol{\mathcal{D}}}$ different sessions for the amount of time equal to Δ. Thus, the matrix $\boldsymbol{\mathcal{D}}$ has a dimension equal to $r_{\boldsymbol{\mathcal{D}}} \times \Delta$, where $r_{\boldsymbol{\mathcal{D}}}$ and Δ are the number of rows (traffic sessions) and the number of columns (time instants) in $\boldsymbol{\mathcal{D}}$. The vector $\mathbf{c}$ of labels contains the application type of the controlled sessions, with a dimension equal to $r_{\boldsymbol{\mathcal{D}}} \times 1$. For example, given the i-th investigated session, it holds that $d_{i,j} \in \boldsymbol{\mathcal{D}}$ and $c_i \in \mathbf{c}$ are the

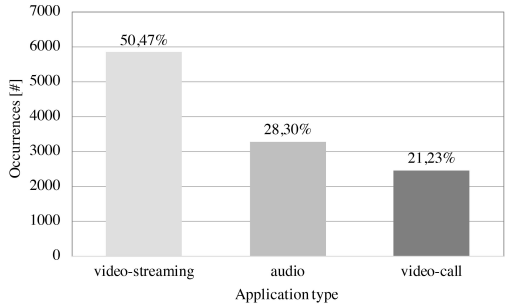


Fig. 2. Number of sessions vs application types in the considered dataset.

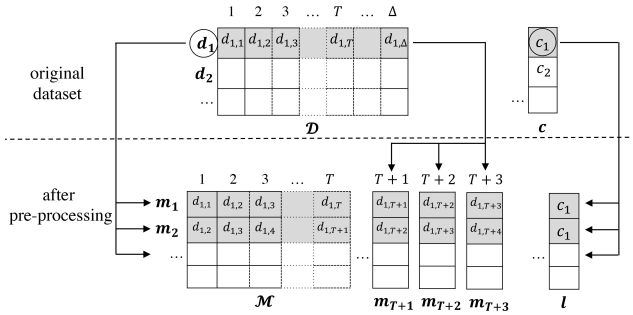


Fig. 3. Pre-processing of the training dataset.

amount of data delivered across the radio interface during the j-th time slot and the label describing the application type of the i-th session, respectively. All the values stored in $\boldsymbol{\mathcal{D}}$ are normalized within the range [0,1] to accelerate the training convergence [56].

The training dataset has been conveniently pre-processed to be managed by our deep learning models. For the sake of clarity, the pre-processing procedure has been depicted in Fig. 3. A new matrix $\boldsymbol{\mathcal{M}}$ is generated from $\boldsymbol{\mathcal{D}}$, whose rows represent the observation windows of duration T. The resulting matrix $\boldsymbol{\mathcal{M}}$ has a dimension of $r_{\boldsymbol{\mathcal{D}}}(\Delta - T + 1) \times T$. The vector $\mathbf{c}$ is used to generate a new set of labels, namely $\mathbf{l}$, describing the application type associated to each portion of the investigated session stored in $\boldsymbol{\mathcal{M}}$. The vector $\mathbf{l}$ has a dimension of $r_{\boldsymbol{\mathcal{D}}}(\Delta - T + 1) \times 1$. A set of new column vectors, namely $\mathbf{m_{T+1}}$, $\mathbf{m_{T+2}}$, and $\mathbf{m_{T+3}}$, with dimension $r_{\boldsymbol{\mathcal{D}}}(\Delta - T + 1) \times 1$, are generated from $\boldsymbol{\mathcal{D}}$ to store the amount of data exchanged between base station and the mobile terminal after the observation window T.

Finally, 80% of $\boldsymbol{\mathcal{M}}$ is used as training set, while the remaining 20% is used as validation set. The number of rows of the matrix $\boldsymbol{\mathcal{M}}$ selected as training set, whose performance will be listed and evaluated, is simply denoted with $r_{\boldsymbol{\mathcal{M}},tr}$.

B. Components of the developed MTL model

Fig. 4 shows the proposed MTL model, embracing three main components: autoencoder, classifier, and predictor. Each component presents specific input and output parameters. The training of the developed MTL model is divided into two stages. The first stage consists of the training of autoencoder. The second stage refers to the training of both classifier and

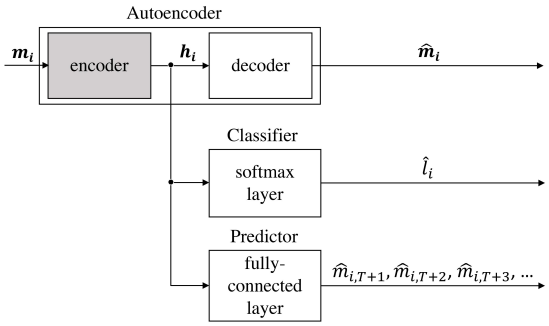


Fig. 4. Our proposed MTL model.

predictor, known the set of feature learning representations provided by the encoder.

1) The autoencoder: It represents a particular Artificial Neural Network (ANN) implementing two key functionalities. Given an input data $\mathbf{m_i} = \{m_{i,1}, ..., m_{i,T}\}$, that is a row of the matrix $\boldsymbol{\mathcal{M}}$, the encoder generates the corresponding feature representation, namely $\mathbf{h_i}$, which then allows the joint execution of the two tasks. Specifically, $\mathbf{h_i} \in \boldsymbol{\mathcal{H}}$ appears like a compression of input data [49] and it is referred to as codeword in the next sections. On the other hand, the decoder provides a reconstruction of the input data, namely $\hat{\mathbf{m}}_\mathbf{i} = \{\hat{m}_{i,1}, ..., \hat{m}_{i,T}\}$, starting from the aforementioned feature learning representation. The autoencoder uses the sigmoid activation function for the output layer and Rectified Linear Unit (ReLU) for other layers [6]. In addition, it also uses weights, that are properly configured during the training phase.

This work investigates two different autoencoder schemes:

- the *Undercomplete Autoencoder*, leveraging regular densely-connected neural network layers, based on MLP [57]. In particular, MLP is a fully-connected and feed-forward neural network, that has low computational complexity.
- the *Seq2Seq Autoencoder*, that manages encoder and decoder functionalities through LSTM [58]. The LSTM is a popular variant of Recurrent Neural Networks (RNNs) that can extract long range temporal dependencies through input, forget, and output gates and mitigate gradient vanishing and exploding problems. This type of neural network is suitable for processing time series because the output of each memory cell may depend on the entire sequence of previous cell states [6], [13], [59]. Due to the intrinsic temporal relations in mobile traffic data, LSTM-based architecture appears as the logical choice, at the cost of higher computational complexity.

To train the two types of autoencoder, weights are iteratively updated in order to minimize the MSE loss function $\mathcal{L}_\mathcal{A}$, formally defined as [57], [60]:

$$\mathcal{L}_\mathcal{A} = \frac{1}{r_{\boldsymbol{\mathcal{M}},tr}} \sum_{i=1}^{r_{\boldsymbol{\mathcal{M}},tr}} \sum_{j=1}^{T} (m_{i,j} - \hat{m}_{i,j})^2 \qquad (1)$$

As shown in Fig. 4, the common feature representation $\mathbf{h_i}$ generated by the autoencoder is provided to both classifier and predictor for driving classification and prediction tasks.

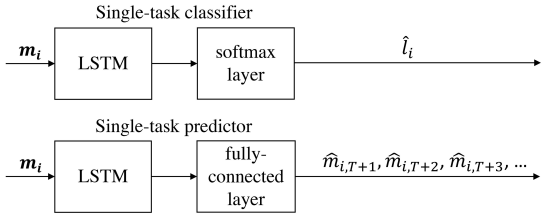


Fig. 5. Baseline single-task learning architectures for classifier and predictor.

2) The classifier: It maps the feature learning representation $\mathbf{h_i}$ to a learned label $\hat{l}_i$ describing the application type of the investigated session. To this end, it uses the softmax layer, based on the softmax activation function [6], working with a number of classes (i.e., the considered application types) equal to 3, even if our methodology is extendable to any number of classes.

The softmax layer of the classifier is configured by penalizing the MSE loss function $\mathcal{L}_\mathcal{C}$ between the true label l_i associated to the input data $\mathbf{m_i}$ and the learned label $\hat{l}_i$ associated to the feature learning representation $\mathbf{h_i}$:

$$\mathcal{L}_\mathcal{C} = \frac{1}{r_{\mathcal{M},tr}} \sum_{i=1}^{r_{\mathcal{M},tr}} \left(l_i - \hat{l}_i\right)^2. \tag{2}$$

Once configured, the classifier accuracy $\mathcal{A}_\mathcal{C}$ quantifies the percentage of correct classifications with respect to the total number of classifications [61]:

$$\mathcal{A}_\mathcal{C} = \frac{\text{number of correct classifications}}{r_{\mathcal{M},tr}} \cdot 100. \tag{3}$$

3) The predictor: It predicts the amount of data that a given session is expected to exchange with the base station after the observation window T, that are: $\hat{m}_{i,T+1}$ stored in $\mathbf{\hat{m}_{T+1}}$, $\hat{m}_{i,T+2}$ stored in $\mathbf{\hat{m}_{T+2}}$, $\hat{m}_{i,T+3}$ stored in $\mathbf{\hat{m}_{T+3}}$, and so on. It makes use of a fully-connected layer with the ReLU activation function [6].

The predictor is configured in order to minimize the MSE loss function $\mathcal{L}_\mathcal{P}$, formulated for $T + 1s$ as [62]:

$$\mathcal{L}_\mathcal{P} = \frac{1}{r_{\mathcal{M},tr}} \sum_{i=1}^{r_{\mathcal{M},tr}} \left(m_{i,T+1} - \hat{m}_{i,T+1}\right)^2. \tag{4}$$

Of course, it is expected that the prediction loss function, which minimizes the difference between the true and the predicted amount of exchanged data, will increase with the time distance between the latest value of the investigated traffic profile and the predicted one.

IV. Performance Evaluation

The conceived MTL architectures have been implemented in Keras, a high-level neural networks API written in Python, running on top of TensorFlow [63], and simulations have been executed on an Intel Core i7 CPU with 16 GB of RAM. Moreover, different configurations of neural networks are investigated to quantify the impact of the observation window, T, on the classifier accuracy, $\mathcal{A}_\mathcal{C}$, and the prediction loss, $\mathcal{L}_\mathcal{P}$. Once the best solutions are selected, we present a complete analysis on the classification and prediction performance together with a discussion on the complexity and convergence of the selected architectures.

To simplify the understanding of the analysis presented in this section, the proposed MTL architectures are named as follows: MTL-U refers to the MTL architecture based on the Undercomplete Autoencoder; MTL-S2S refers to the MTL architecture based on the Seq2Seq Autoencoder.

Assuming to describe the ratio between the size of the input layer and the size of hidden layers in the form *X:Y* for the neural networks with only one hidden layer and *X:Y:Z* for the neural networks with two hidden layers, the investigated configurations include *8:5*, *8:6*, *8:8*, and *8:5:3*. The observation window T is chosen in the range from 5 to 20. Regarding the autoencoder, the size of the codeword is also set to different values (please see Tables III and IV for further details).

The training phase for all the components belonging to the designed MTL architectures is done with 200 epochs. The Adam optimization is used to iteratively update the network weights based on the training data [64].

To provide further insight, the comparison with baseline single-task learning architectures, that do not use the autoencoder and that tackle traffic classification and prediction separately, is presented as well. In particular, the reference single-task architectures selected for the cross-comparison are based on LSTM because, as stated in Section III-B, this type of neural network is suitable for processing time series. Furthermore, due to the wide adoption of LSTM in the state-of-the-art deep learning models (e.g., [13], [16]–[19]), LSTM-based architecture appears as the logical choice for the comparison (single-task learning) schemes, as well as for the MTL approach. Assuming to work with the same training dataset and to adopt the same set of symbols, the single-task classifier and the single-task predictor are depicted in Fig. 5.

A. Selection of suitable MTL architectures

Autoencoder loss, $\mathcal{L}_\mathcal{A}$, classification accuracy, $\mathcal{A}_\mathcal{C}$, and prediction loss, $\mathcal{L}_\mathcal{P}$, achieved for all the configurations of the designed MTL architectures are reported in Tables III and IV. The same performance indexes obtained with single-task approaches are reported in Table V. For both MTL and single-task architectures and for each observation window T, these results are used to select the configurations that ensure the best performance.

Regarding the conceived MTL architectures, the analysis concerns multiple objectives, that refer to the maximization of $\mathcal{A}_\mathcal{C}$ and the minimization of $\mathcal{L}_\mathcal{P}$. To this end, a performance metric, $\mathcal{P}_{\mathcal{MTL}}$, is defined in (5) as a weighted linear sum of obtained results for each task, where the weight α may assume an arbitrary value from 0 to 1 [65], [66]. Since the higher the loss, the lower the performance, the min-max normalization is performed for $\mathcal{L}_\mathcal{P}$ to properly combine the two metrics [61], considering the minimum prediction loss reported in Tables III, IV, and V (i.e., $\mathcal{L}_{\mathcal{P}_{min}}$), the maximum prediction loss reported in Tables III, IV, and V (i.e., $\mathcal{L}_{\mathcal{P}_{max}}$), the value of the normalized metric describing the worst performance

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TVT.2020.3005724

$$\mathcal{P}_{\mathcal{MTL}} = \alpha \mathcal{A}_{\mathcal{C}} + (1-\alpha)\left[\frac{\mathcal{L}_{\mathcal{P}} - \mathcal{L}_{\mathcal{P}min}}{\mathcal{L}_{\mathcal{P}max} - \mathcal{L}_{\mathcal{P}min}}(\mathcal{L}'_{\mathcal{P}max} - \mathcal{L}'_{\mathcal{P}min}) + \mathcal{L}'_{\mathcal{P}min}\right] \tag{5}$$

TABLE III
PERFORMANCE OF MTL-U. FOR EACH T, THE BEST CONFIGURATION IS HIGHLIGHTED.

T [s]	Codeword	MTL-U											
		1 hidden layer									2 hidden layers		
		8:5			8:6			8:8			8:5:3		
		$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]
5	**3**	0.83	89.20	2.56	6.69	**93.02**	**2.54**	5.68	91.72	2.50	23.10	86.86	2.60
	4	0.83	88.39	2.56	4.25	91.22	2.50	2.12	92.37	2.53	23.09	87.51	3.49
10	**3**	6.64	95.02	1.77	6.04	95.41	1.70	2.26	91.93	1.63	8.20	93.21	1.67
	4	2.93	93.47	1.73	4.98	**96.99**	**1.66**	3.94	94.61	1.61	6.73	92.99	1.68
	5	2.06	91.02	1.73	4.85	95.45	1.72	2.12	92.90	1.61	4.49	93.33	1.69
15	**3**	4.32	90.56	1.27	5.12	94.75	1.18	2.28	90.51	1.15	4.25	94.95	1.21
	4	3.70	90.16	1.24	5.15	93.90	1.17	0.57	94.92	1.15	2.86	94.19	1.19
	5	1.97	93.82	1.18	3.62	98.08	1.20	0.25	94.37	1.14	4.80	97.53	1.16
	10	3.61	96.21	1.18	2.69	96.31	1.11	2.49	**98.75**	**1.03**	1.82	93.51	1.22
20	**3**	3.84	88.97	0.96	4.56	97.64	0.91	2.50	90.50	0.83	1.43	95.41	0.88
	4	0.52	94.57	0.95	3.32	99.43	0.88	0.30	94.94	0.81	3.38	95.22	0.85
	5	0.43	94.91	0.91	3.44	98.26	0.90	0.34	91.16	0.79	3.04	94.10	0.91
	10	2.29	96.60	0.90	2.48	**99.36**	**0.81**	1.80	97.23	0.73	1.56	94.95	0.87

TABLE IV
PERFORMANCE OF MTL-S2S. FOR EACH T, THE BEST CONFIGURATION IS HIGHLIGHTED.

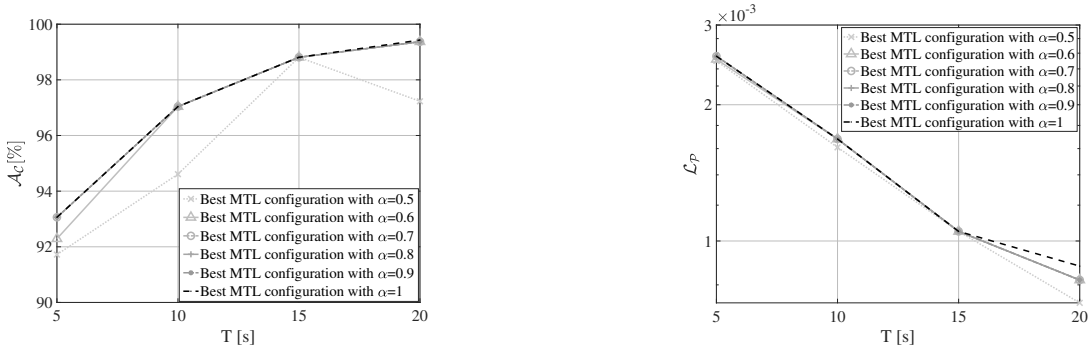
T [s]	Codeword	MTL-S2S											
		1 hidden layer									2 hidden layers		
		8:5			8:6			8:8			8:5:3		
		$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]	$\mathcal{L}_{\mathcal{A}}$ [$\cdot 10^{-3}$]	$\mathcal{A}_{\mathcal{C}}$ [%]	$\mathcal{L}_{\mathcal{P}}$ [$\cdot 10^{-3}$]
5	**3**	0.014	92.72	2.41	0.024	92.40	2.48	0.058	92.25	2.33	16.68	90.09	2.44
	4	16.72	92.56	2.50	0.011	92.35	2.39	16.70	92.10	2.39	16.73	90.59	2.45
	5	0.027	94.55	2.46	0.048	94.26	2.42	0.017	**94.59**	**2.33**	16.70	90.36	2.51
10	**3**	0.0081	96.52	1.55	0.016	97.19	1.51	0.029	93.53	1.50	16.69	92.22	1.54
	4	0.0045	96.71	1.51	0.0099	96.69	1.53	0.0045	95.67	1.40	15.50	96.44	1.61
	5	0.019	96.17	1.51	0.021	**97.33**	**1.46**	0.0032	94.45	1.38	0.060	97.52	1.54
15	**3**	0.0066	96.21	1.17	0.023	97.15	1.07	0.011	98.03	0.87	16.68	91.99	0.97
	4	11.16	98.54	1.01	0.018	95.48	1.03	0.011	97.75	1.02	11.13	96.60	1.22
	5	0.0083	98.16	1.03	0.014	95.22	0.98	0.0034	98.26	0.95	0.0076	95.48	1.01
	10	0.0029	97.96	0.91	0.0048	98.06	0.86	0.0075	**99.33**	**0.81**	0.019	95.41	1.01
20	**3**	0.014	97.85	0.78	0.0035	92.52	0.66	0.012	93.68	0.61	0.0047	98.47	0.70
	4	0.0087	98.18	0.75	0.0035	94.07	0.74	0.0060	98.50	0.67	0.016	97.56	0.77
	5	0.0039	98.00	0.81	0.0045	94.92	0.75	0.0032	97.84	0.61	0.019	98.23	0.75
	10	0.0032	98.95	0.77	0.0034	97.92	0.67	0.0072	**99.64**	**0.62**	0.0055	95.02	0.68

(i.e., $\mathcal{L}'_{\mathcal{P}max} = 0$), and the value of the normalized metric describing the best performance (i.e., $\mathcal{L}'_{\mathcal{P}min} = 100$).

Figs. 6 and 7 show the performance of the MTL configurations that register the highest $\mathcal{P}_{\mathcal{MTL}}$ metric as a function of α, for MTL-U and MTL-S2S, respectively. These figures help to identify the suitable values of α to be used for the selection of the best MTL configurations. Reported curves demonstrate that $\alpha = 0.5$ and $\alpha = 1$ cannot be used for this purpose. In fact, if $\alpha \leq 0.5$, the multi-objective metric $\mathcal{P}_{\mathcal{MTL}}$ suggests to select configurations that register low classification accuracy. On the contrary, when $\alpha = 1$, the multi-objective metric $\mathcal{P}_{\mathcal{MTL}}$ suggests to select configurations that register higher prediction losses, especially when T increases. Other values of α provide similar outcomes. Thus, the rest of this paper considers the best configurations of the proposed MTL architectures selected with $\alpha = 0.8$. They are highlighted in Tables III and IV.

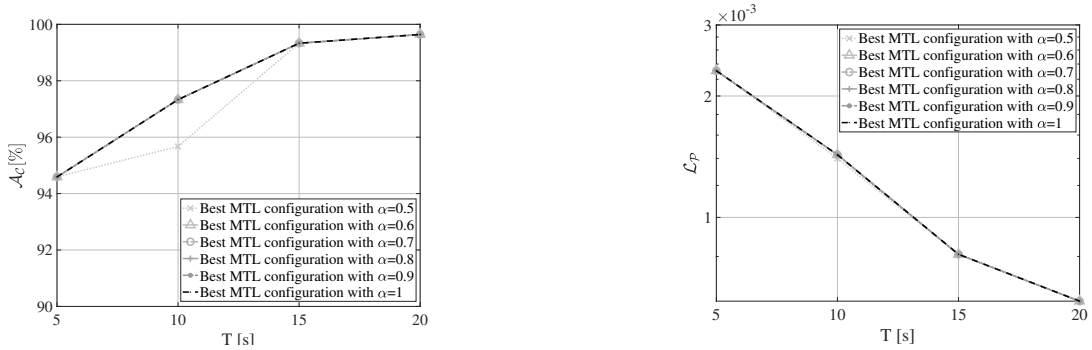
Regarding the single-task approaches, the best configurations are simply selected by considering those that offer better performance for each T. Also in this case, they are highlighted in Table V.

In general, we note that the performance of both MTL and single-task approaches improve when T increases because more data are used to make decisions. Focusing the attention on the proposed MTL model, there is not a precise relationship between MTL performance and codeword size: while MTL-S2S always achieves the best performance with the biggest codeword size, the same consideration cannot be done for



(a) (b)

Fig. 6. (a) Classification accuracy and (b) prediction loss at $T + 1s$ registered by the best configurations of MTL-U, as a function of α.



(a) (b)

Fig. 7. (a) Classification accuracy and (b) prediction loss at $T + 1s$ registered by the best configurations of MTL-S2S, as a function of α.

TABLE V
PERFORMANCE OF THE SINGLE-TASK APPROACH. FOR EACH T, THE BEST CONFIGURATION IS HIGHLIGHTED.

T [s]	Single-task classifier			
	1 hidden layer			2 hidden layers
	8:5	8:6	8:8	8:5:3
	$\mathcal{A}_\mathcal{C}$ [%]	$\mathcal{A}_\mathcal{C}$ [%]	$\mathcal{A}_\mathcal{C}$ [%]	$\mathcal{A}_\mathcal{C}$ [%]
5	88.02	**92.52**	91.44	90.81
10	95.56	94.69	**95.97**	93.22
15	96.76	96.60	**97.18**	96.07
20	95.36	**97.73**	97.52	97.04

T [s]	Single-task predictor			
	1 hidden layer			2 hidden layers
	8:5	8:6	8:8	8:5:3
	$\mathcal{L}_\mathcal{P}$ [$\cdot 10^{-3}$]	$\mathcal{L}_\mathcal{P}$ [$\cdot 10^{-3}$]	$\mathcal{L}_\mathcal{P}$ [$\cdot 10^{-3}$]	$\mathcal{L}_\mathcal{P}$ [$\cdot 10^{-3}$]
5	2.64	2.56	2.47	**2.43**
10	1.77	1.67	**1.48**	1.55
15	1.22	1.12	**0.97**	1.11
20	0.91	0.84	**0.77**	0.79

MTL-U.

B. Classification performance

Fig. 8 depicts the classification accuracy of the selected architectures as a function of T. As already anticipated, the performance always improves when T increases because all the learning architectures can use a higher number of training data to perform session classification. It is also evident that the single-task approach registers lower accuracy levels, ranging from 92.52% to 97.73%. On the contrary, better results are registered by the proposed MTL architectures: in this case, it is possible to reach an accuracy level up to 99.64%. The conducted study also demonstrates that MTL-S2S achieves higher classification accuracy for each T.

TABLE VI
F-SCORE ANALYSIS.

Architecture	F-score			
	T=5s	T=10s	T=15s	T=20s
MTL-U	0.9312	0.9755	0.9904	0.9926
MTL-S2S	0.9501	0.9652	0.9927	0.9971
Single-task classifier	0.9198	0.9586	0.9817	0.9866

Classification performance can be further investigated through the *F-score* [61] index. Theoretically, the higher the *F-score* value, the better the ability of the classifier to make proper decisions. The results summarized in Table VI generally confirm what already discussed. In fact, *F-score*

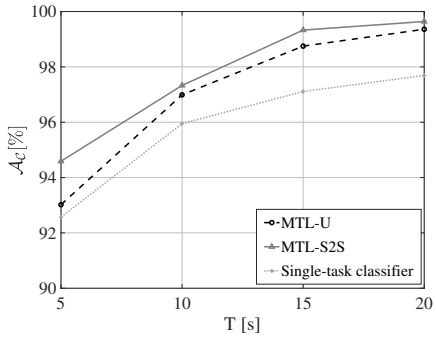


Fig. 8. Classification performance.

improves when T increases, and the single-task approach always registers the lowest *F-score* values. Regarding the proposed MTL architectures, an exception is reported when $T = 10s$: in that case, even if MTL-U registers the highest *F-score*, it achieves a lower classification accuracy than MTL-S2S because of a higher error rate for a specific application type (see the study on the confusion matrices proposed below).

To analyze which classes are mismatched in the classification process, the confusion matrices are provided in Fig. 9 for each T. In general, both MTL architectures misclassify video-streaming sessions with audio-streaming ones. Nonetheless, such an error classification rate decreases when T increases. When $T = 5s$, in fact, 14% and 13% of video-streaming sessions are (wrongly) classified as audio-streaming by MTL-U and both MTL-S2S and the single-task classifier, respectively. These percentages decrease to 2% for MTL-U, 1% for MTL-S2S, and 4% for the single-task classifier when $T = 20s$. However, also in this case, it is possible to observe how the proposed MTL architectures always provide better results with respect to those measured for the single-task approach. Going more into detail, MTL-S2S presents the highest percentage of sessions, which are correctly classified, for each T, except for $T = 10s$. When $T = 10s$, as anticipated with the analysis of *F-score*, MTL-U reports a lower $\mathcal{A}_C$ than MTL-S2S. However, MTL-U reports a higher *F-score*. The confusion matrices show the reason why it occurs. The percentages of video-streaming sessions which are correctly classified by MTL-U (see Fig. 9(b), on the left) and MTL-S2S (see Fig. 9(b), in the middle) are 94% and 92%, respectively.

C. Prediction performance

Fig. 10 shows the prediction loss registered for the time instants $T + 1s$, $T + 2s$, and $T + 3s$. First of all, it is evident that the curves for $T + 3s$ are incomplete. In this case, the training process always fails when $T = 5s$. As expected, the prediction loss decreases with the observation window T, because the learning architectures have more training data to make a prediction. Regarding the prediction performed at both $T+1s$ and $T+2s$, MTL-S2S and MTL-U always register the best and the worst performance levels, respectively. On the other hand, when the prediction is done a $T + 3s$, the single-task approach slightly exceeds the prediction losses registered by MTL-U.

TABLE VII
COMPLEXITY ANALYSIS OF LEARNING ARCHITECTURES.

Architecture		# Parameters			
		T=5s	T=10s	T=15s	T=20s
MTL-U	Autoencoder	98	314	805	960
	Classifier	72	189	543	618
	Predictor	43	114	411	486
MTL-S2S	Autoencoder	806	1607	5496	8781
	Classifier	438	665	2513	3603
	Predictor	386	563	2211	3201
Single-task	Classifier	111	513	1068	1068
	Predictor	82	491	1036	1781

In summary, MTL-S2S always guarantees the lowest prediction losses, at the cost of higher complexity (see Section IV-D). MTL-U registers the worst performance when the prediction is done at $T+1s$ and $T+2s$. The single-task approach exhibits intermediate performance levels when $T+1s$ and $T+2s$, but it registers the highest prediction losses at $T + 3s$. Obtained results also confirm the ability of LSTM, which is exploited in both MTL-S2S and the single-task scheme, to suitably process time series by taking into account the temporal sequence of TBS values.

D. Complexity and convergence analysis

The complexity of selected learning architectures is evaluated by measuring the number of trainable parameters: the higher the number of parameters, the higher the complexity level. Results are summarized in Table VII. Firstly, it is evident that the complexity of all the investigated learning architectures increases when the observation window T increases. MTL-S2S always registers the highest complexity. Also the single-task approach, based on LSTM, has a high complexity because of the structures of LSTM cells. On the contrary, MTL-U guarantees the lowest complexity for each observation window T.

The convergence analysis evaluates the performance of the investigated learning architectures (including autoencoder loss, classification accuracy, and prediction loss) as a function of the number of epochs considered during the training phase. Fig. 11 shows the autoencoder loss as a function of the number of epochs. MTL-S2S has the slowest convergence time, while providing the lowest autoencoder loss. Fig. 12 depicts the classification accuracy as a function of the number of epochs. While the proposed MTL architectures reach similar performance, the single-task approach always registers the highest convergence time. Fig. 13 shows the prediction loss as a function of the number of epochs. In this case, it is possible to observe that MTL-S2S achieves lower performance losses, at the cost of a slower convergence time.

E. A further evaluation with more classes

As described in Section III-A, the proposed MTL approach can be applied to different scenarios with a higher number of classes. To provide further insight, the training dataset considered in this work allowed us to evaluate the performance of the proposed methodology when considering the

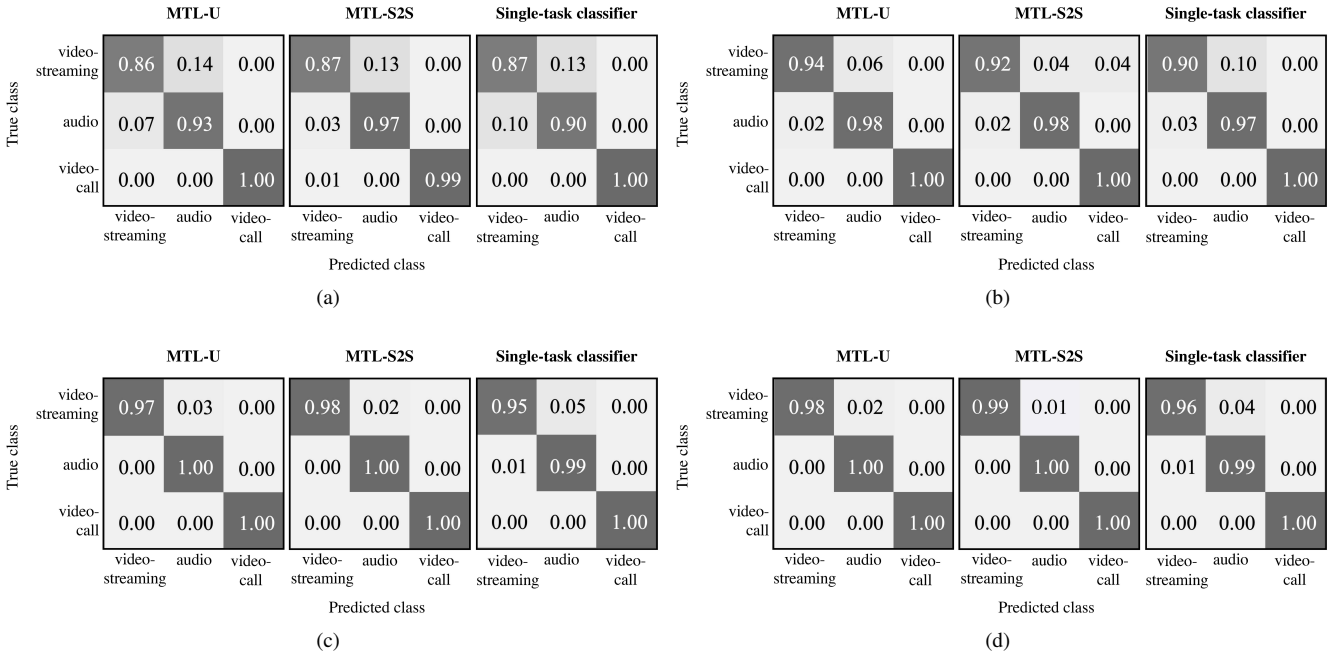


Fig. 9. Confusion matrix for MTL-U, MTL-S2S, and the single-task classifier when (a) $T = 5s$, (b) $T = 10s$, (c) $T = 15s$, and (d) $T = 20s$.

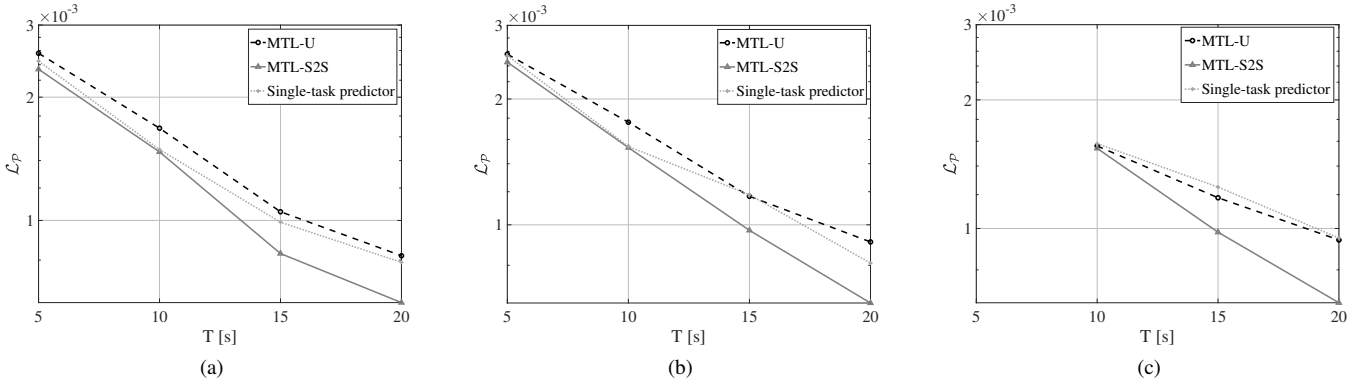


Fig. 10. Prediction performance of the best configurations of MTL-U and MTL-S2S and the single-task approach: a) prediction loss at $T + 1s$, b) prediction loss at $T + 2s$, and c) prediction loss at $T + 3s$.

six available classes of applications: YouTube, Vimeo, Spotify, Google Music, Skype, and WhatsApp Messanger. Specifically, differently from the original investigation, the applications belonging to the same service category have been treated as separate classes. We tested the configurations of the MTL-S2S approach that achieved the best performance in the analysis of three service categories only. Figs. 14 and 15 depict the classification accuracy and the prediction loss of MTL-S2S and the single-task schemes with six classes as a function of T. Obtained results further confirm that the proposed MTL approach outperforms baseline single-task scheme also in scenarios with a higher number of classes. Differently from the previous case, however, lower accuracy levels are caused by very similar patterns of applications (especially those of audio-streaming type) and it is increasingly difficult to distinguish the different applications when the observation window T decreases.

V. Conclusions

This work has tailored a Multi-Task Learning model for traffic classification and prediction at the mobile edge, which leverages data mining from the Physical Downlink Control Channel and two types of autoencoders (i.e., the Undercomplete Autoencoder and the Sequence to Sequence Autoencoder) exploited as key building blocks for obtaining common feature representations. Different configurations of neural networks have been trained with a real dataset collected from an operative mobile network in Spain. Moreover, a wide set of simulations has investigated the performance of the developed approach in terms of classification accuracy, prediction loss, complexity, and convergence. A cross-comparison with respect to conventional single-task learning schemes, that do not use autoencoders and that are generally investigated in the current state of the art for traffic classification and prediction, has also demonstrated that: i) the Multi-Task Learning architec-

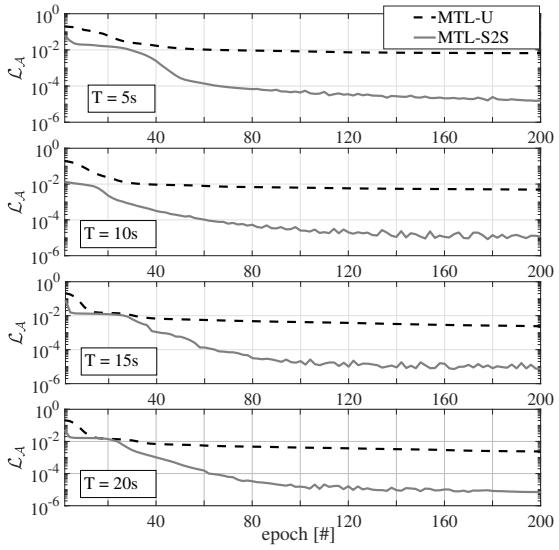


Fig. 11. Autoencoder loss vs number of epochs.

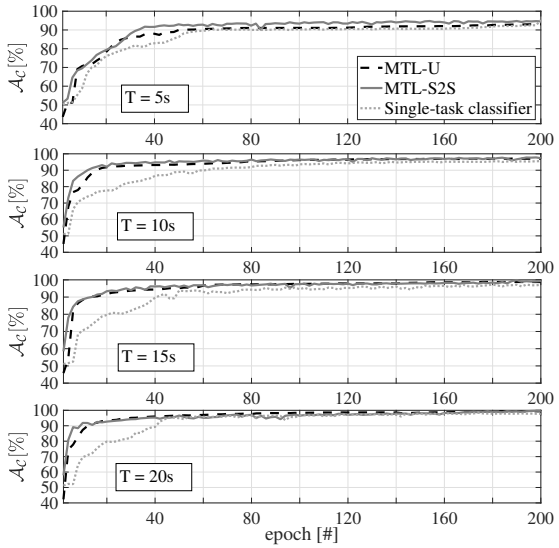


Fig. 12. Classification accuracy vs number of epochs.

tures, leveraging the autoencoders, always guarantee higher performance than the single-task learning approach, ii) the Multi-Task Learning architecture based on the Sequence to Sequence Autoencoder always achieves the highest classification accuracy and the lowest prediction losses, at the cost of a higher complexity and convergence time. Further research activities will exploit the conceived methodology to properly design advanced techniques for mobile network optimization, ranging from radio resource scheduling and admission control, mobility management and energy saving mechanisms, to network slicing and dynamic placement of virtualized functions.

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] M. Paolini, "Mastering Analytics: How to benefit from big data and network complexity: An Analyst Report." *RCR Wireless News*, 2017.

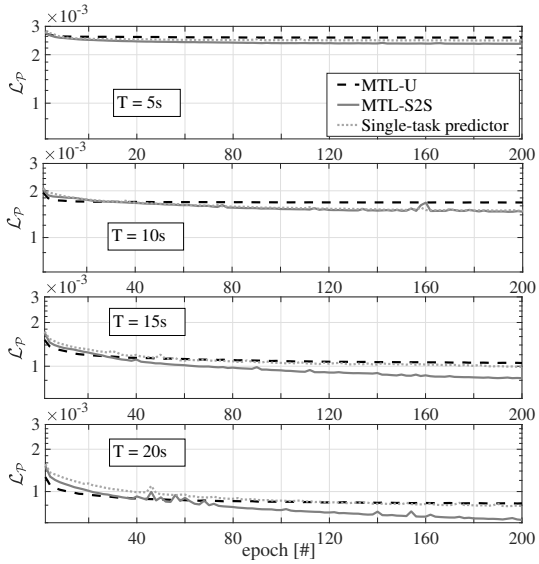


Fig. 13. Prediction loss vs number of epochs.

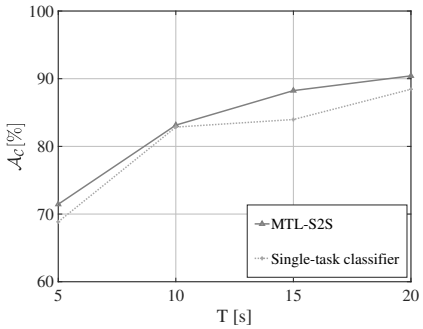


Fig. 14. Classification performance with six application classes.

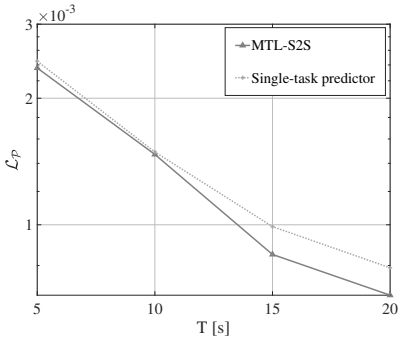


Fig. 15. Prediction performance at $T + 1s$ with six application classes.

[3] 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," *J. Internet Services Appl.*, vol. 9, no. 1, p. 16, 2018.

[4] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, "Big Data-Driven Optimization for Mobile Networks toward 5G," *IEEE Netw.*, vol. 30, pp. 44–51, 2016.

[5] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.

[6] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 21,

no. 3, pp. 2224–2287, 2019.
[7] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, 2019.
[8] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
[9] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski, "Five Disruptive Technology Directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, 2014.
[10] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design Considerations for a 5G Network Architecture," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 65–75, 2014.
[11] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks," in *Proc. Int. Conf. Intell. Secur. Inform.*, 2017, pp. 43–48.
[12] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning," *Soft Comput.*, pp. 1–14, 2017.
[13] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, June 2019.
[14] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway," *IEEE Access*, vol. 6, pp. 55 380–55 391, 2018.
[15] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks," *IEEE Commun. Lett*, vol. 22, no. 8, pp. 1656–1659, 2018.
[16] 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 Int. Conf. Comput. Commun. (INFOCOM)*, 2017, pp. 1–9.
[17] L. Chen, D. Yang, D. Zhang, C. Wang, J. Li *et al.*, "Deep Mobile Traffic Forecast and Complementary Base Station Clustering for C-RAN Optimization," *J. Netw. Comput. Appl.*, vol. 121, pp. 59–69, 2018.
[18] J. Feng, X. Chen, R. Gao, M. Zeng, and Y. Li, "DeepTP: An End-to-End Neural Network for Mobile Cellular Traffic Prediction," *IEEE Netw.*, vol. 32, no. 6, pp. 108–115, 2018.
[19] 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 Commun. Mag.*, 2019.
[20] C.-W. Huang, C.-T. Chiang, and Q. Li, "A Study of Deep Learning Networks on Mobile Traffic Forecasting," in *Proc. 28th IEEE Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2017, pp. 1–6.
[21] C. Zhang and P. Patras, "Long-term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. and Comput.*, 2018, pp. 231–240.
[22] C. Zhang, M. Fiore, and P. Patras, "Multi-Service Mobile Traffic Forecasting via Convolutional Long Short-Term Memories," in *Proc. IEEE Int. Symp. Meas. Netw. (M&N)*, July 2019, pp. 1–6.
[23] R. Caruana, "Multitask Learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.
[24] Y. Zhang and Q. Yang, "A Survey on Multi-Task Learning," *arXiv preprint arXiv:1707.08114*, 2017.
[25] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine Learning Meets Computation and Communication Control in Evolving Edge and Cloud: Challenges and Future Perspective," *IEEE Commun. Surveys Tuts.*, 2019.
[26] K. Bian, C. Gao, Y. Tao, Y. Zhang, L. Song, S. Dong, and X. Li, "Learning at the Edge: Smart Content Delivery in Real World Mobile Social Networks," *IEEE Netw.*, vol. 33, no. 4, pp. 208–215, 2019.
[27] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Trans. Emerg. Topics Comput*, pp. 1–1, 2019.
[28] S. Rezaei and X. Liu, "Multitask Learning for Network Traffic Classification," *arXiv preprint arXiv:1906.05248*, 2019.
[29] H. Sun, Y. Xiao, J. Wang, J. Wang, Q. Qi, J. Liao, and X. Liu, "Common Knowledge Based and One-Shot Learning Enabled Multi-Task Traffic Classification," *IEEE Access*, vol. 7, pp. 39 485–39 495, 2019.
[30] A. Rago, G. Piro, H. D. Trinh, G. Boggia, and P. Dini, "Unveiling Radio Resource Utilization Dynamics of Mobile Traffic through Unsupervised Learning," in *Proc. IEEE Netw. Traffic Meas. Anal. Conf. (TMA)*, Paris, France, June 2019.
[31] F. Meneghello, M. Rossi, and N. Bui, "Smartphone Identification via Passive Traffic Fingerprinting: a Sequence-to-Sequence Learning Approach," *IEEE Netw.*, vol. 34, no. 2, pp. 112–120, 2020.
[32] H. D. Trinh, A. F. Gambin, L. Giupponi, M. Rossi, and P. Dini, "Mobile Traffic Classification through Physical Channel Fingerprinting: a Deep Learning Approach," *arXiv preprint arXiv:1910.11617*, 2019.
[33] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile Traffic Prediction from Raw Data Using LSTM Networks," in *Proc. 29th IEEE Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2018, pp. 1827–1832.
[34] D. D. Nguyen, H. X. Nguyen, and L. B. White, "Reinforcement Learning With Network-Assisted Feedback for Heterogeneous RAT Selection," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 6062–6076, 2017.
[35] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of Machine Learning Classifiers for Mobile Malware Detection," *Soft Comput.*, vol. 20, no. 1, pp. 343–357, Jan. 2016.
[36] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed Machine Learning Approaching LAN Speeds," in *Proc. USENIX Conf. Networked Syst. Des. Implementation (NSDI)*, Berkeley, CA, USA, 2017, pp. 629–647.
[37] M. Miozzo, N. Piovesan, and P. Dini, "Coordinated Load Control of Renewable Powered Small Base Stations through Layered Learning," *IEEE Trans. Green Commun. Netw.*, pp. 1–1, 2019.
[38] Y. Zhang, Y. Guan, K. Bian, Y. Liu, H. Tuo, L. Song, and X. Li, "EPASS360: QoE-aware 360-degree Video Streaming over Mobile Devices," *IEEE Trans. Mobile Comput.*, pp. 1–1, 2020.
[39] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets Activation Scheme for Scalable Mobile Edge Computing with Transmission Power Control and Virtual Machine Migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, 2018.
[40] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, "Who do you sync you are? Smartphone Fingerprinting via Application Behaviour," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2013, pp. 7–12.
[41] Y. Liu, S. Zhang, B. Ding, X. Li, and Y. Wang, "A Cascade Forest Approach to Application Classification of Mobile Traces," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.
[42] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I Know What You Did On Your Smartphone: Inferring App Usage Over Encrypted Data Traffic," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2015, pp. 433–441.
[43] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust Smartphone App Identification Via Encrypted Network Traffic Analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, 2018.
[44] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2658–2713, 2018.
[45] P. Wang, X. Chen, F. Ye, and Z. Sun, "A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.
[46] 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 Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1790–1821, 2017.
[47] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-Art Deep Learning: Evolving Machine Intelligence Toward Tomorrow's Intelligent Network Traffic Control Systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, 2017.
[48] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar, "Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1988–2014, 2019.
[49] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R. Murthy, and M. van der Schaar, "Machine Learning in the Air," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2184–2199, Oct. 2019.
[50] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, 2019.
[51] X. Song, H. Kanasugi, and R. Shibasaki, "DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level." in *IJCAI*, vol. 16, 2016, pp. 2618–2624.
[52] N. Bui and J. Widmer, "OWL: A Reliable Online Watcher for LTE Control Channel Measurements," in *Proc. ACM Workshop All Things Cellular Operations Appl. Challenges*, 2016, pp. 25–30.
[53] Ericsson, "Ericsson Mobility Report," November 2019.
[54] Cisco, "Cisco Annual Internet Report (2018–2023)," *White Paper*, March 2020.
[55] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," 3rd Generation Partnership Project (3GPP), Tech. Specification (TS) 36.213, May 2016.
[56] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[58] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[59] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[60] S. Hashem and B. Schmeiser, "Improving Model Accuracy Using Optimal Linear Combinations of Trained Neural Networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 792–794, 1995.
[61] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.
[62] H. Feng and Y. Shu, "Study on Network Traffic Prediction Techniques," in *Proc. IEEE Int. Conf. Wireless Commun. Netw. Mobile Comput.*, vol. 2, 2005, pp. 1041–1044.
[63] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[65] Y. Liao, S. Kodagoda, Y. Wang, L. Shi, and Y. Liu, "Understand Scene Categories by Objects: A Semantic Regularized Scene Classifier Using Convolutional Neural Networks," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, 2016, pp. 2318–2325.
[66] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2018, pp. 7482–7491.

Gennaro Boggia (S'99-M'01-SM'09) received, with honors, the Dr. Eng. and Ph.D. degrees in electronics engineering, both from the Politecnico di Bari, Bari, Italy, in July 1997 and March 2001, respectively. Since September 2002, he has been with the Department of Electrical and Information Engineering, Politecnico di Bari, where he is currently a Full Professor. From May 1999 to December 1999, he was a Visiting Researcher with the TILab, TelecomItalia Lab, Torino, Italy, where he was involved in the study of the core network for the evolution of Third-Generation (3G) cellular systems. In 2007, he was a Visiting Researcher at FTW, Vienna, Austria, where he was involved in activities on passive and active traffic monitoring in 3G networks. He has authored or coauthored more than 150 papers in international journals or conference proceedings, gaining more than 2300 citations. He is active in the IETF ICNRG working group and in the IEEE WG 6TiSCH. He is also regularly involved as a Member of the Technical Program Committee of many prestigious international conferences. His research interests include the fields of Wireless Networking, Cellular Communication, Internet of Things (IoT), Network Security, Security in Iot, Information Centric Networking (ICN), Protocol stacks for industrial applications, Internet measurements, and Network Performance Evaluation. Dr. Boggia is currently an Associate Editor for the IEEE Commun. Mag., the ETT Wiley Journal, and the Springer Wireless Networks journal.

Arcangela Rago (S'19) received the M.Sc. degree (with honors) in telecommunication engineering from Politecnico di Bari, Bari, Italy, in 2018, where she is currently pursuing the Ph.D. degree with the Department of Electrical and Information Engineering. Her main research interests include machine learning and data analytics for network optimization. She was a recipient of the Best Poster Award at SMFC 2019, held in conjunction with IEEE SMC 2019. She is involved in the Apulia Region (Italy) Research project INTENTO (36A49H6).

Giuseppe Piro (S'10-M'13) Since November 2018, he is an Assistant Professor in Telecommunication at Politecnico di Bari. In March 2018, he held the habilitation as "Associate Professor" in Telecommunications Engineering, according to the National Scientific Habilitation procedure (ASN 2016-2018). He received a first level degree and a second level degree (both cum laude) in Telecommunications Engineering from "Politecnico di Bari", Italy, in 2006 and 2008, respectively. He received the Ph.D. degree in Electronic Engineering from "Politecnico di Bari", Italy, on March 2012. His main research interests include secure Internet of Things and Industry 4.0, 5G systems, data-centric and programmable architectures for the Future Internet, nano-networks, Internet models and network measurements. His research activity is documented in more than 80 peer-reviewed international journals and conference papers, accounting for more than 3800 citations and a H-index of 24 (Scholar Google). At the time of this writing, he is the local investigator of the PRIN project no. 2017NS9FEY entitled "Realtime Control of 5G Wireless Networks: Taming the Complexity of Future Transmission and Computation Challenges". Moreover, he is involved in the European EU H2020 GUARD project. He is also involved in Italian MIUR PON projects (Pico&Pro, FURTHER, AGREED, RAFAEL) and in Apulia Region (Italy) Research project INTENTO. He founded 5G-ai-simulator, LTE-Sim, and NANO-SIM projects and is a developer of Network Simulator 3. In the past, he was involved in EU H2020 projects, like FANTASTIC-5G, BONVOYAGE, and symbIoTe, as well as in the "Apulia Israel joint Accelerator (AIJA)" project. He is also regularly involved as member of the TPC of many prestigious international conferences. Currently, he serves as Associate Editor for Sensors journal (MDPI), Internet Technology Letter (Wiley) and Wireless Communications and Mobile Computing journal (Hindawi).

Paolo Dini received M.Sc. and Ph.D. from the Università di Roma La Sapienza, in 2001 and 2005, respectively. He is currently a Senior Researcher with the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). His current research interests include sustainable networking and computing, distributed optimization and optimal control, machine learning and data analytics. 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 more than 20 research and development projects and is currently the Coordinator of the EU H2020 MSCA SCAVENGE European Training Network on sustainable mobile networks with energy harvesting capabilities. He serves as a TPC in many international conferences and workshops and as a reviewer for several scientific journals of the IEEE, Springer, Wiley, and Elsevier.