IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 November 2022, accepted 23 December 2022, date of publication 11 January 2023, date of current version 19 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3236261

SURVEY

Forecasting Network Traffic: A Survey and Tutorial With Open-Source Comparative Evaluation

GABRIEL O. FERREIRA[1], **(Member, IEEE), CHIARA RAVAZZI**[1], **(Member, IEEE), FABRIZIO DABBENE**[1], **(Senior Member, IEEE), GIUSEPPE C. CALAFIORE**[2,3], **(Fellow, IEEE), AND MARCO FIORE**[4], **(Senior Member, IEEE)**

[1]IEIIT CNR Institute, Politecnico di Turin, 10129 Torino, Italy
[2]Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Turin, Italy
[3]College of Engineering and Computer Science (CECS), VinUniversity, Hanoi 12426, Vietnam
[4]IMDEA Networks Institute, 28918 Leganés, Spain

Corresponding author: Fabrizio Dabbene (fabrizio.dabbene@ieiit.cnr.it)

This work was supported in part by the European Commission through the Horizon 2020 Framework Programme under Grant H2020-MSCA-ITN-2019, and in part by Marie Sklodowska-Curie Actions Innovative Training Networks (MSCAITN)-EID through Big dAta aNalYtics for radio Access Networks (BANYAN) under Proposal 860239.

ABSTRACT This paper presents a review of the literature on network traffic prediction, while also serving as a tutorial to the topic. We examine works based on autoregressive moving average models, like ARMA, ARIMA and SARIMA, as well as works based on Artifical Neural Networks approaches, such as RNN, LSTM, GRU, and CNN. In all cases, we provide a complete and self-contained presentation of the mathematical foundations of each technique, which allows the reader to get a full understanding of the operation of the different proposed methods. Further, we perform numerical experiments based on real data sets, which allows comparing the various approaches directly in terms of fitting quality and computational costs. We make our code publicly available, so that readers can readily access a wide range of forecasting tools, and possibly use them as benchmarks for more advanced solutions.

INDEX TERMS Artificial neural networks, forecasting models, network traffic, prediction, statistical models.

I. INTRODUCTION

Over the past decade, networks have embraced a softwarization process that is granting increasing control capabilities by operators on their infrastructures. This trend creates new opportunities for network management, many of which build upon anticipatory decision-making paradigms: traffic engineering, resource allocation or service orchestration policies are enacted in advance of future fluctuations in traffic demands, as reviewed by Bui et al. [1]. Anticipatory networking yields the potential to improve network resource utilization and end user's quality of service substantially with respect to traditional reactive, human-in-the-loop strategies. Yet, its actual gains largely depend on the accuracy of the predictions that proactive decisions are taken upon. Therefore, the role of traffic time-series forecasts is today more central than ever in the design of cutting-edge solutions for network management.

Network traffic prediction is in fact a largely explored subject in networking, with early works dating back to the seventies and a flurry of recent proposals fostered by the success of machine and deep learning tools. Several surveys already exist that review and classify the many works on network traffic forecasting based on different approaches, from statistical models to artificial neural network ones.

The most recent reviews focus on specific forecasting approaches, such as deep neural networks (DNN) targeted by Cao et al. [2] and Huang et al. [3] or the graph neural networks (GNN) considered by Jiang and Luo [4]; these surveys are hence limited to a subset of the techniques available in the literature, whereas our goal is to provide a comprehensive view of all major prediction strategies. Closer to

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

our approach, other reviews encompass multiple classes of models. For instance Joshi and Hadi [5] compile an extensive survey on network analysis and traffic prediction, describing pre-processing techniques such as discretization method and feature selection, which are very relevant to improve the data quality due to the presence of outliers or inconsistent information, and linear and non-linear approaches for network traffic prediction. Also, Hendikawati et al. [6] survey papers using the standard linear models (ARMA, ARIMA, SARIMA) as well as multivariate methods where the prediction is a function of two or more variables like vector autoregressive (VAR), vector moving average (VMA), and vector autoregressive moving average (VARMA); they also list papers using neural networks, support vector machines, wavelet transform, fuzzy logic, and Adaptative Neuro-Fuzzy Inference System. Finally, Jiang [7] reviews studies that specifically target cellular traffic prediction, and classify the models into statistical, machine learning, and deep learning models.

All the existing surveys above take a traditional approach of listing a variety of works proposing network traffic predictors, briefly explaining their operation and proposing a classification or taxonomy. Instead, this paper has a different and more ambitious objective: as in regular surveys, we present a wide range of previous studies that employ statistical and artificial neural network techniques for network traffic forecasting; however, at the same time we also propose a *complete and self-contained tutorial* on those prediction techniques. To this end, we include a full explanation of the mathematical formulation of each technique, and of all the steps that one should follow to best configure the associated model. Unlike previous reviews, this paper thus lets the reader gain a deep understanding of the functioning of available traffic forecasting approaches.

Moreover, we *implement all reviewed techniques for traffic prediction*, and run numerical experiments with all the techniques presented. Our experimental approach is new, since no previous survey does the same, and it has a twofold advantage. First, it allows performing a direct comparison of the many solutions proposed in the literature in terms of prediction quality and computational cost; we employ real-world citywide datasets of mobile network traffic to perform such a comparative evaluation, and provide a first-of-its-kind ranking of techniques. Second, we publicly release our implementations of network traffic predictors,[1] which enables non-experts to get acquainted with practical realizations of the different forecasting models, and also provides researchers with a sound lineup of benchmarks that can be used in comparative analyses of novel solutions. When combined with open datasets of traffic, such as the one we consider for our analysis, our implementations can improve the reproducibility and verifiability of research results aiming at traffic prediction.

TABLE 1. Mobile devices generation technology.

Generation	Base Station	Controller
2G GSM	Base Transceiver Station	Base Station Controller
3G UMTS	Node B	Radio Network Controller
4G LTE	eNodeB	-

The document is organized as follows. In Section II, we give a brief introduction to network topology and the advantages of 5*th* generation mobile network. Besides, we present a mathematical formulation for the problem, some literature approaches, and their performance evaluation. In Section III, we discuss ARMA, ARIMA, and SARIMA models, presenting the mathematical formulation of each and explaining their limitations. Besides, we also illustrate pre-processing approaches such as time-series differencing and decomposition, and briefly present clustering techniques, which may be useful to improve the prediction results. Section IV introduces different types of artificial neural network models: three recurrent neural networks (RNN, LSTM, and GRU) and convolutional neural networks (CNN). Again, we detail the mathematical foundations of each model and explain their internal operation with intuitive figures. In Section V, we map relevant papers targeting traffic forecasting to the different techniques presented in the previous sections, while also outlining the associated data sets and evaluation strategies. Section VI present comprehensive numerical experiments comparing all the presented approaches in presence of real-world mobile traffic demands, when considering different configurations for the statistical and neural networks approaches. Section VII provides open research directions. Finally, Section VIII draws conclusions.

II. BACKGROUND

A simplified cellular network topology can be divided into three main parts: radio access network, core network, and user equipment.

- *Radio access network (RAN)*: responsible for the radio communication between mobile device and network, and includes several base stations (BS) and one base station controller (BSC), which is responsible for communication resources management, handover tasks, etc. According to Lin and Zhang [8], BSs are responsible for about 80% of energy consumption of a cellular network. Each generation of wireless cellular technology has its specific BS, as presented in Table 1 (the first generation was based on analog technology).
  GSM, UMTS, and LTE are Global System for Mobile Communications, Universal Mobile Telecommunications System, and Long-Term Evolution, respectively. 4G LTE has no controller because eNodeB is also able to accomplish this task.
- *Core Network*: it is responsible for connecting different RANs among themselves and also to external networks, as the internet.
- *User Equipment*: mobile devices of the users.

[1] The code and documentation are made available, see the Appendix for details.

Note that *RAN* and *Core Network* are complex parts of a cellular network. Details about them can be found in Kanani et al. [9], Abed et al. [10], and Seddigh et al. [11].

Nowadays, studies related to 5G are on the rise. This technology presents improvements compared to the previous generations, as presented in Al-Falahy and Alani [12]: smaller latency, faster download and upload, higher bandwidth, higher number of connected devices, data processing speed ($Mbps/m^2$) 100 times faster than 4G, reduction in energy consumption, etc. All these developments allow many applications that depend on device-to-device communication, as Internet of Things and autonomous cars.

5G network architecture is heterogeneous, that is, includes macro and small cells. Hence, optimization problems concerning spectrum allocation, location of these cells and energy saving of those not serving users are studied, as in Su et al. [13]. Many of these optimization techniques are based on big data generated by both user and network operator. As presented by Zheng et al. [14], this data needs to be processed, due to incomplete or uncertain information, and then transformed into knowledge, such as users location, mobility pattern, and communication behavior. Details on 5G network topology and technical aspects (frequency range, power consumption, channel bandwidth) are presented in Gupta and Jha [15]. In this work, the base stations are also referred to as antennas or towers, regardless of their generation technology.

It is important to emphasize that the network traffic load can also be measured and predicted inside fixed environments, such as Local Area Network (LAN), where different computers or smartphones are connected to routers or switches. The advantages of network traffic prediction (management, anticipatory decision making, and resource allocation) are also required in such a context. Some of the works cited in this survey design forecasting models for these kind of data sets, as it will be further presented.

A. NETWORK TRAFFIC FORECASTING

Consider that a time-series containing T network traffic load observations at a tower j is given by $X_t^{(j)} = [x_{t-T}^{(j)}, \ldots, x_{t-1}^{(j)}]$. The elements of this vector could have different time information depending on the data set, such as: amount of transferred data (bytes), number of internet connections, number of incoming/outgoing cellphone calls and SMS, or any other relevant information.

In network traffic forecasting, the objective is to design a model to predict the future load $X_{t^+}^{(0)} = [x_t^{(0)}, \ldots, x_{t+k}^{(0)}]$ for the next k instants ahead, based on previous observations in $X_t^{(0)}, X_t^{(j)}$ with $j = 1, \ldots, n_k$, as well as any other available information that the data set may provide (geographical position, past errors, weather condition, etc). The predicted value is given by

$$x_t^{(0)} = f(\Lambda\, T_1(M_t) + \sum_{i=1}^{\xi} \Gamma_i\, T_i(Q_{i_t})), \qquad (1)$$

where

$$M_t = \begin{bmatrix} X_t^{(1)} & \ldots & X_t^{(2)} & X_t^{(3)} & X_t^{(4)} & \ldots & X_t^{(5)} \\ & \ddots & & \vdots & & \iddots & \\ X_t^{(6)} & \ldots & X_t^{(7)} & X_t^{(8)} & X_t^{(9)} & \ldots & X_t^{(10)} \\ X_t^{(11)} & \ldots & X_t^{(12)} & X_t^{(0)} & X_t^{(13)} & \ldots & X_t^{(14)} \\ X_t^{(15)} & \ldots & X_t^{(16)} & X_t^{(17)} & X_t^{(18)} & \ldots & X_t^{(19)} \\ & \iddots & & \vdots & & \ddots & \\ X_t^{(20)} & \ldots & X_t^{(21)} & X_t^{(22)} & X_t^{(23)} & \ldots & X_t^{(n_k)} \end{bmatrix},$$

$$Q_{i_t} = \begin{bmatrix} \Upsilon_{i_t}^{(1)} & \ldots & \Upsilon_{i_t}^{(2)} & \Upsilon_{i_t}^{(3)} & \Upsilon_{i_t}^{(4)} & \ldots & \Upsilon_{i_t}^{(5)} \\ & \ddots & & \vdots & & \iddots & \\ \Upsilon_{i_t}^{(6)} & \ldots & \Upsilon_{i_t}^{(7)} & \Upsilon_{i_t}^{(8)} & \Upsilon_{i_t}^{(9)} & \ldots & \Upsilon_{i_t}^{(10)} \\ \Upsilon_{i_t}^{(11)} & \ldots & \Upsilon_{i_t}^{(12)} & \Upsilon_{i_t}^{(0)} & \Upsilon_{i_t}^{(13)} & \ldots & \Upsilon_{i_t}^{(14)} \\ \Upsilon_{i_t}^{(15)} & \ldots & \Upsilon_{i_t}^{(16)} & \Upsilon_{i_t}^{(17)} & \Upsilon_{i_t}^{(18)} & \ldots & \Upsilon_{i_t}^{(19)} \\ & \iddots & & \vdots & & \ddots & \\ \Upsilon_{i_t}^{(20)} & \ldots & \Upsilon_{i_t}^{(21)} & \Upsilon_{i_t}^{(22)} & \Upsilon_{i_t}^{(23)} & \ldots & \Upsilon_{i_t}^{(n_k)} \end{bmatrix},$$

$\Lambda \in \mathbb{R}^{1\times T}$ and $\Gamma \in \mathbb{R}^{1\times T}$ are vectors containing the parameters to be estimated. $T_1(M_t) \in \mathbb{R}^{T\times 1}$ is a tensor operation that generates a vector $\hat{X}_t = [\hat{x}_{t-T}, \ldots, \hat{x}_{t-1}] \in \mathbb{R}^{T\times 1}$, e.g.:

$$\hat{x}_{t-i} = \frac{1}{n_k}\sum_{j=1}^{n_k} x_{t-i}^{(j)}, \qquad i = 1, \ldots, T,$$

$$\hat{x}_{t-i} = \max x_{t-i}^{(j)}, \quad j = 1, \ldots, n_k, \quad i = 1, \ldots, T,$$

or convolution operations may be used, as it will be later presented. Note that if just information about tower $X_t^{(0)}$ is available, then $M_t = X_t^{(0)}$. ξ is the number of factors (besides the time-series traffic load) that are used to construct the model, such as prediction errors (in the case of moving average models), hidden state (when using recurrent neural networks), or any other variables that may impact the prediction quality and are available on the data set. $T_i(Q_{i_t}) \in \mathbb{R}^{T\times 1}$ are calculated in a similar procedure and $\Upsilon_{i_t}^{(j)}$ is a vector containing information about tower j used to construct the prediction model.

It is important to notice that the position of each $X_t^{(j)}$ in matrix M_t is not necessarily determined by the geographical position of each tower. It could also be determined via other methodologies and metrics, later presented, used to cluster the time-series. The same analysis is extended to $\Upsilon_{i_t}^{(j)}$ in Q_{i_t}.

A prediction model that is capable of forecasting the traffic load with small error may have big commercial value for companies that design the network. With good k estimates for $x_t^{(0)}, \ldots, x_{t+k}^{(0)}$, these companies can assign optimized amount of hardware/software, making the network capable of attending the demands inside an area, but also achieving the optimal tradeoff between capacity (number of users, data transmission rate, coverage area, etc) and monetary cost. According to Bega et al. [16], not only over-dimensioned networks causes economic losses, but also under-dimensioned, since they could lead to subscribers' churn rates and fees associated with Service-Level agreements.

Usually, the data sets shared by the companies have information about the traffic load of multiple antennas at a specif granularity time and the geographical position of each one. In Section VI, numerical experiments to compare the approaches discussed in this survey are performed on a data set of the telecommunication activity in Milan, Italy. There, the following information is captured in 10-minute time slots:

- received/sent SMS;
- incoming/outgoing call;
- internet connection start/end.

The items above form Call Detail Record (CDR), which were pre-processed by the operator so as to ensure the anonymity o the data subjects generating the traffic. The mobile traffic data is mapped on a regular spatial grid. For a full description of the data set, see Barlacchi et al. [17].

Our goal is to use equation (1) and, based on a time horizon of i observations, that is, for

$$X_t^{(j)} = [x_{t-i}^{(j)}, \dots, x_{t-2}^{(j)}, x_{t-1}^{(j)}]$$
$$\Upsilon_t^{(j)} = [\upsilon_{t-i}^{(j)}, \dots, \upsilon_{t-2}^{(j)}, \upsilon_{t-1}^{(j)}],$$

to predict k steps of the traffic load of a specific tower $j = 0$:

$$X_{t+}^{(0)} = [x_t^{(0)}, \dots, x_{t+k}^{(0)}].$$

Suitable values for prediction and observation horizons are discussed in the following Sections.

B. DATA-DRIVEN APPROACHES

This survey is focused on Autoregressive Moving Average and Neural Networks based prediction models. The first class is a linear combination of previous values of errors and traffic load, leading to relative simple solutions with high data interpretability. As it is discussed in most of the bibliography provided in this work, the forecast capability of neural networks based models present higher quality (in general) when compared to other approaches. Besides, it is relatively simple to combine different types of ANNs to obtain a model that is able to capture particular patterns, e.g., a CNN-RNN based approach can provide spatio-temporal knowledge about the network traffic load. Due to the aforementioned reasons, a great number of papers related to network traffic prediction uses ARIMA or ANN based strategies. Figure 1 illustrates this scenario. A search on Scopus for papers whose indexed keywords contain *time-series, forecast*, and the name of each of the approaches briefly presented below is done.

Linear Regression (LR) is a machine learning technique where one wants to estimate the parameters for a linear model. Basically, the dependent variable to predict (output) is calculated based on a linear combination of independent variables (input). The parameters are calculated by minimizing some loss function, for instance the sum of the squares or sum of absolute residuals. The first case is equivalent to maximizing the likelihood and a Gaussian distribution for the error term is assumed. The second one, known as Robust Linear Regression, a Laplace distribution is assumed and may provide better results when the data set contains outliers. In Lechowicz [18], Linear regression is used to predict bandwidth blocking probability of a network.

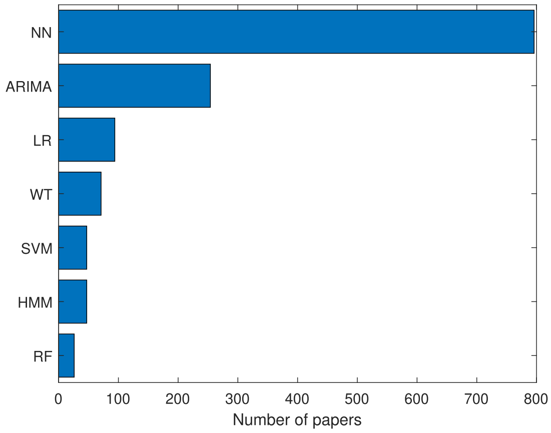


FIGURE 1. Number of papers on Scopus whose indexed keywords contain *time-series, forecast,* and one of the respective approaches.

1) SUPPORT VECTOR MACHINE

(SVM) is a machine learning technique widely used in classification problems. In summary, it is an optimization problem whose output is the parameters of a hyperplane that separates the input space into two halfspaces. Subsequently, the classification is performed according to the halfspace of each sample. SVM can also be used in regression and network traffic prediction problems, as in Stepanov et al. [19]. The authors compared SVM forecasting results with predictions calculated by using the Random Forecast algorithm.

2) RANDOM FOREST

(RF) is an ensemble of decision trees, where different random subsets of the training data are used to build different decision trees. Subsequently, a prediction is made by considering majority votes, for classification problems, or the mean of outputs in case of regression.

3) WAVELET TRANSFORM

(WT) can be used as a decomposition method where the signal features, as frequency or trends, change over time. Distinctly from the Fourier Transform, where the signals are decomposed into sine waves, Wavelet Transform generates wavelets, fast decaying wave-shaped oscillation with zero mean. As it will be later discussed, traffic network time-series are highly non-stationary, which makes such a tool very useful in this context. Despite the fact it is a decomposition technique, when used with prediction models, Wavelet Transform may present good forecasting results, as in Lu and Yang [20] and Tian [21].

4) HIDDEN MARKOV MODEL (HMM)

is a probabilistic model where the sequence of events can be described as a Markov process: a state at a time x_t is affected only by the previous one, that is, x_{t-1}. Additionally, these states are not directly measured, however impact an output y_t according to known rules. Hence, one wants to estimate x_t with observations about y_t. In Chadza et al. [22], the authors used HMM to predict network attacks on simulated period attack of 10 days containing 50 and 420 attacking/victim machines and 30 servers.

Autoregressive models are constructed as a linear combination of past observations. Usually, the model is estimated such that its parameters maximize the likelihood of making the correct predictions. Similarly, the moving average models are also linear, however the prediction is calculated based on previous errors. When these two models are used simultaneously, one has the so-called **autoregressive moving average models**, explained in Section III, that can be extended to ARIMA and SARIMA.

A very large class of functions can be approximated by using **Neural Networks**, that have the advantage of inserting non-linearity in the models. Many different prediction strategies are based on Neural Networks, where different patterns (such as spatio-temporal) can be learned during a training phase and subsequently used for predictions. Some of the techniques widely used in traffic load forecast are based on Convolutional and Recurrent Neural Networks, detailed in Section IV. Other models are discussed in Chen et al. [23].

Network traffic load prediction is divided (at least) into three big areas: time-series clustering, decomposition, and prediction model. In Figure 2 we summarize the main research topics involved in such a subject. Some examples of strategies regarding each field are given (many others can be found in the references later presented.) It is important to emphasize that these areas are connected and affect the prediction result: for instance, a good time-series decomposition could lead to better clustering result due to stochastic component filtering; subsequently, the estimation of the parameters for a prediction model can be calculated and, according to the cluster some network belongs, its traffic load is predicted.

C. PERFORMANCE EVALUATION

Different metrics can be used to evaluate the three areas presented in Figure 2. Concerning the prediction models, the quality is measured according to some errors, presented in Table 2, between the predicted and actual values of the time-series.

Usually, a good clustering result is the one where the elements inside the same cluster are similar, and elements in different ones have low similarity (according to some adopted distance measure). However, clustering is an unsupervised learning technique where the size and number of clusters may not be known a priori. Therefore, evaluating if the result of some clustering strategy is good or not is not a trivial task.

According to Aghabozorgi et al. [24], evaluation metrics are divided into two groups: **external** and **internal** indexes. In the first one, there is a ground truth where the cluster that each time-series belongs to is known. Hence, the results of a clustering strategy are compared with a validation data set by means of some metrics, as: Jaccard Score, Rand Statistic, Folkes and Mallow Index, purity, entropy, etc. This is used in Chiş et al. [25], where the authors propose a new algorithm for time-series clustering and the evaluation of the approach is performed by comparing, in terms of Jaccard Score, Rand Statistic, and Folkes and Mallow Index, the labels of their clustering results with the labels of the original clusters.

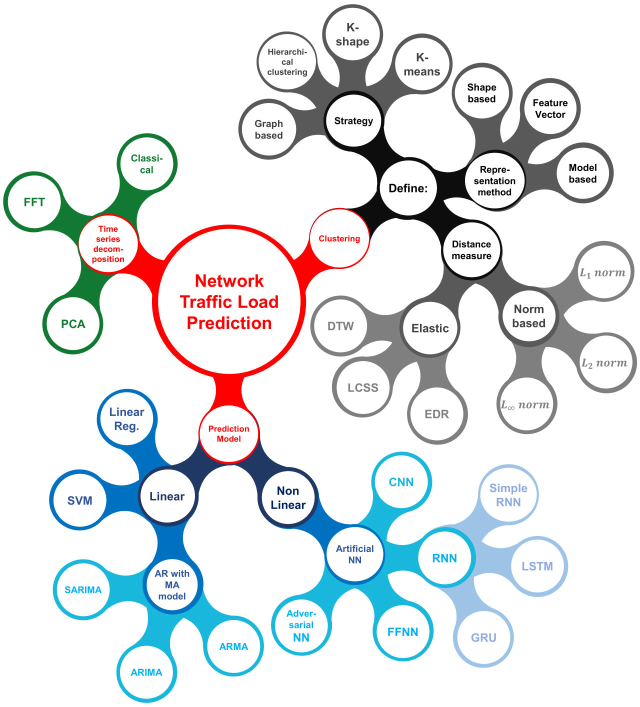


FIGURE 2. Overview of network traffic load prediction topics.

TABLE 2. Acronyms for errors measurement.

Model	Limitation
RE	Relative error
PE	Percent error
RE	Correlation coefficient
ME	Mean error
CC	Cross correlation
ACE	Absolute cumulative error
MAE	Mean absolute error
SSE	Sum of squared error
SER	Signal to error ratio
ACC	Prediction accuracy
$NMSE$	Normalized mean squared error
$SDAV$	Standard deviation absolute value
$AVRE$	Absolute value of relative error
$RMSE$	Root mean squared error
$CORR$	Empirical correlation coefficient
$MAPE$	Mean absolute percent error
$MARE$	Mean absolute relative error
$MOES$	Mean squared observed error
$PMCC$	Predictive model choice criteria
$NRMSE$	Normalized root mean squared error
$ARMSE$	Average root mean squared error
$RMSEP$	Relative mean separation

One can see that in traffic load prediction this may not be applicable, since the data sets usually do not provide information about clustering. In this case, a common practice is to minimize the errors concerning specific error metrics, such as SSE, root-mean-squared standard deviation (RMSSTD), semi-partial R-squared, R-squared, distance between two clusters, etc. By solving the optimization problem with one of the cited metrics, one expects to obtain high intra-cluster and low inter-cluster similarities. In Halkidi et al. [26], a detailed explanation of each one of these metrics is given.

III. SIMPLE STATISTICAL MODELS

Before introducing the models, the concept of **stationary** time-series is introduced. As reported by [27], let $X_t = [x_{t-T}, \ldots, x_{t-1}]$ be a time-series with $E(X_t^2) < \infty$. Its mean and covariance functions are given by

$$\mu_X(t) = E(X_t) \tag{2}$$

$$\gamma_X(i,j) = Cov(X_i, X_j) = E[(X_i - \mu_X(i))(X_j - \mu_X(j))], \tag{3}$$

for all integers i and j. X_t is stationary if:

1) $\mu_X(t)$ is independent of t;
2) $\gamma_X(t+k, t)$ is independent of t for each k.

When these two conditions are satisfied, the statistical properties of the time-series are similar to those of its shifted version (in time).

A. ARMA MODELS

In Hoong [28] and Tan et al. [29], prediction models for network traffic demand are developed by applying the Autoregressive Moving Average (ARMA) time-series modeling technique. Such model is composed of two main terms: an AutoRegressive component (AR), which is the sum of past observations with a white noise constant, and Moving average (MA) component, which is the sum of past white noise errors with the expected value of the time-series. Both are modeled, respectively, as

$$(AR) \quad x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \varepsilon_t \tag{4}$$

and

$$(MA) \quad x_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t, \tag{5}$$

where:

- $\phi_i, \ldots, \phi_p$ are the AR parameters to be determined;
- $\varepsilon_t \sim WN(0, \sigma^2)$ is white noise error value;
- $\theta_i, \ldots, \theta_p$ are the MA parameters to be determined;
- μ is the expected value of x_t;
- p is the autoregressive order;
- q is the moving average order.

Here x_t is the traffic load at time t, which we want to predict based on past instances.

According to Brockwell and Davis [27], X_t can be modeled as an ARMA(p, q) process if it is stationary and if for every t

$$x_t - \sum_{i=1}^{p} \phi_i x_{t-i} = \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t. \tag{6}$$

Additionally, it is ARMA(p, q) with mean μ if $X_t - \mu$ is an ARMA(p, q) process.

Considering $\mu = 0$, equation (6) becomes

$$x_t = \sum_{i=1}^{p} \phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t. \tag{7}$$

For instance, an ARMA(2,1) model with $\mu = 0$ can be represented as

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \theta_1 \varepsilon_{t-1} + \varepsilon_t. \tag{8}$$

Such models provide better results when applied to stationary data. Hence, when the data is non-stationary, some pre-processing is necessary. This is usually achieved by using *log return* transformation, as

$$r_t = \ln x_t - \ln x_{t-1}, \tag{9}$$

where r_t is the log return value at time t. Then, the model we need to estimate takes the form

$$r_t = \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t. \tag{10}$$

After the transformation, both papers make use of **Cronos**, an open-source tool written in *C#* programming language, to evaluate the predicting activity. This software makes use of the **Maximum Likelihood Estimation- MLE** to estimate the model parameters. With MLE, the model parameters are calculated in such way that the likelihood of making the correct observations are maximized.

To evaluate the quality of prediction, Hoong [28] used the mean squared error of the log returns, as

$$MSE = \frac{1}{n} \sum (value_a(i) - value_b(i))^2, \tag{11}$$

where:

- $value_a$ = i^{th} actual log return;
- $value_b$ = i^{th} predicted log return.

The experiment proposed by the author consisted in designing two models, ARMA(3,0) and ARMA(2,1), and comparing their MSE on 5 step size predictions. ARMA(3,0) models achieved better results for network traffic with seasonal pattern, while ARMA(2,1) presented lower MSE for network traffic with cyclical pattern.

MLE for Autoregressive term: consider that each term in the time-series is independent and identically distributed (iid) and the set of observations is defined as $\mathcal{D}$. Then

$$p(\mathcal{D}|\phi) \propto \prod_{t=1}^{n} p(x_t|\phi, x_{t-1}) \tag{12}$$

where $p(\mathcal{D}|\phi)$ is the likelihood to be maximized, and n is the number of observations. In linear regression problems, the noise terms are typically assumed to be Gaussian. Hence, $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ in equation (4), consequently

$$p(x_t|\phi, x_{t-1}) = \left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\left(\frac{1}{2\sigma^2}\right)(x_t - \Phi^T x)^2\right), \tag{13}$$

with $x^T = [x_{t-1}, \dots, x_{t-n}]$ and $\Phi^T = [\phi_1, \dots, \phi_n]$.

By considering equations (12) and (13) and applying the *negative log likelihood- NLL*, one has

$$NLL(\theta) = \frac{n}{2}\ln(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_t - \Phi^T x)^2. \tag{14}$$

Hence, we should compute Φ such that equation (14) is minimized. Consider that all x_t and x are concatenated in vectors Y and W, respectively. So,

$$\sum_{i=1}^{n}(x_t - \Phi^T x)^2 = \left\| Y - W^T\Phi \right\|_2^2 \tag{15}$$

This is a least square problem that can be solved by means of the Normal Equation, see Calafiore and El Ghaoui [30]

$$(WW^T)\Phi = WY. \tag{16}$$

B. ARIMA AND SARIMA MODELS

When the time-series is not stationary, the use of ARMA models is possible by using transformations, as discussed before. However, if the time-series is stationary after d differencing times, one may use the ARIMA(p, d, q) model, where the I stands for *integrated*, see Jung et al. [31]. Hence, ARMA is a special case of ARIMA models (when there is no differencing involved). This category of models is discussed next.

1) TIME-SERIES DIFFERENCING

The differenced time-series, as used in Alsharif et al. [32], is simply defined as $X'_t = [x'_{t-T+1}, \dots, x'_{t-1}]$, with

$$x'_{t-j} = x_{t-j} - x_{t-j-1}, \tag{17}$$

for $j = \{1, \dots, T-1\}$. When the differenced time-series is not stationary yet, one may use the second-order differencing, defined as $X''_t = [x''_{t-T+2}, \dots, x''_{t-1}]$, with:

$$x''_{t-j} = x'_{t-j} - x'_{t-j-1} \tag{18}$$

for $j = \{1, \dots, T-2\}$. For instance, consider the first sample, that is $j = 1$:

$$\begin{aligned} x''_{t-1} &= x'_{t-1} - x'_{t-2} \\ &= (x_{t-1} - x_{t-2}) - (x_{t-2} - x_{t-3}) \\ &= x_{t-1} - 2x_{t-2} + x_{t-3}. \end{aligned} \tag{19}$$

Considering the backward shift operator $Bx_t \doteq x_{t-1}$, equation (19) can be written as

$$\begin{aligned} x''_{t-1} &= B^0 x_{t-1} - 2\,Bx_{t-1} + Bx_{t-2} & (20)\\ &= B^0 x_{t-1} - 2\,Bx_{t-1} + B^2 x_{t-1} & (21)\\ &= (1 - 2B + B^2)x_{t-1} = (1-B)^2 x_{t-1}. & (22) \end{aligned}$$

A general formulation for a differencing of order d at any instant k is thus

$$x_k^{(d)} = (1-B)^d x_k. \tag{23}$$

Then, you construct the ARMA model on the new variables $x_k^{(d)}$. Note that X_t, X'_t, and X''_t have, respectively, T, $T-1$, and $T-2$ samples.

Another strategy sometimes used is the seasonal differencing, defined as $X_t^{(s)} = [x_{t-T+S}^{(s)}, \dots, x_{t-1}^{(s)}]$, where

$$x_{t-j}^{(s)} = x_{t-j} - x_{t-j-S}, \tag{24}$$

for $j = \{1, \dots, T-S\}$ and S is the lag value. For instance, if some event has seasonality of 24 hours, one may set $S = 24$.

2) MODEL DEFINITION

According to Yu et al. [33], ARMA models can be represented as

$$x_t = \sum_{i=1}^{p}\phi_i x_{t-i} - \sum_{j=0}^{q}\theta_j \varepsilon_{t-j} \tag{25}$$

or

$$\phi^{[p]}(B)x_t = \theta^{[q]}(B)\varepsilon_t \tag{26}$$

with

$$\begin{aligned} \phi^{[p]}(B) &= 1 - \phi_1 B - \phi_2 B^2 - \dots - \phi_p B^p, \\ \theta^{[q]}(B) &= 1 - \theta_1 B - \theta_2 B^2 - \dots - \theta_q B^q. \end{aligned}$$

If

$$\nabla = 1 - B \tag{27}$$

is the differencing operator, then the ARIMA(p, d, q) model is

$$\phi^{[p]}(B)\nabla^d x_t = \theta^{[q]}(B)\varepsilon_t. \tag{28}$$

For instance, consider $d = 1$ and $p = 2$, then the left side of equation (28) becomes

$$\begin{aligned} &(1 - \phi_1 B - \phi_2 B^2)(1-B)x_t \\ &\quad = (1 - B - \phi_1 B + \phi_1 B^2 - \phi_2 B^2 + \phi_2 B^3)x_t \\ &\quad = (x_t - x_{t-1}) - \phi_1(x_{t-1} - x_{t-2}) - \phi_2(x_{t-2} - x_{t-3}). \end{aligned} \tag{29}$$

Here, the parameter d represents the number of times differencing should be applied to make the time-series stationary.

In a similar way, if the time-series to be modeled are non-stationary and have a periodical component with period S, one may use seasonal differencing and equation (27) becomes

$$\nabla_S = 1 - B^S, \tag{30}$$

and the SARIMA$(p, d, q)(P, D, Q)_S$ model is described by

$$\phi^{[p]}(B)\Phi^{[P]}(B^S)\nabla^d\nabla_S^D x_t = \theta^{[q]}(B)\Theta^{[Q]}(B^S)\varepsilon_t \tag{31}$$

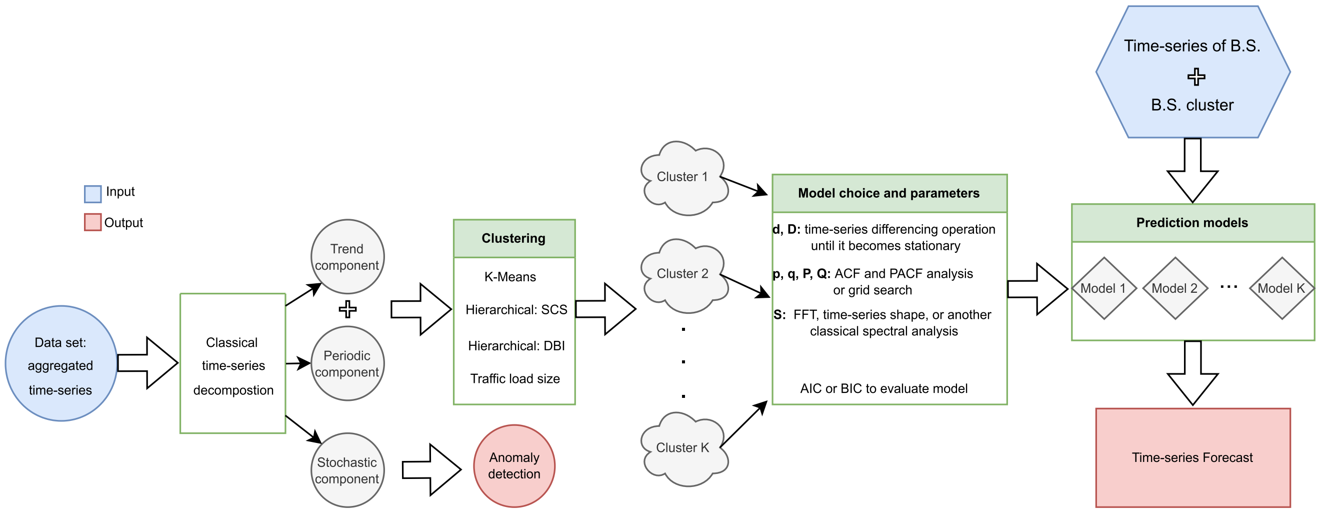

Input
Output
Data set: aggregated time-series
Classical time-series decompostion
Trend component
Periodic component
Stochastic component
Anomaly detection
Clustering
K-Means
Hierarchical: SCS
Hierarchical: DBI
Traffic load size
Cluster 1
Cluster 2
Cluster K
Model choice and parameters
d, D: time-series differencing operation until it becomes stationary
p, q, P, Q: ACF and PACF analysis or grid search
S: FFT, time-series shape, or another classical spectral analysis
AIC or BIC to evaluate model
Time-series of B.S.
B.S. cluster
Prediction models
Model 1
Model 2
Model K
Time-series Forecast

FIGURE 3. General procedure to obtain SARIMA$(p, d, q)(P, D, Q)_S$ models for network traffic prediction.

where:

- P: seasonal autoregressive order;
- D: seasonal difference order;
- Q: seasonal moving average order;
- $\Phi^{[P]}(B^S) = 1 - \Phi_1 B^S - \Phi_2 B^{2S} - \ldots - \Phi_p B^{PS}$;
- $\Theta^{[Q]}(B^S) = 1 - \Theta_1 B^S - \Theta_2 B^{2S} - \ldots - \Theta_q B^{QS}$.

C. TRAFFIC FORECAST WITH SARIMA MODELS

In this subsection, a general procedure to predict network traffic using SARIMA models is explained. It is important to emphasize that this is a general approach based on the literature review. For instance, some papers do not use cluster strategies, others cluster each base station after time-series decomposition, and some apply clustering strategies without considering time-series decomposition.

The most general procedure to obtain a model for prediction is illustrated in Figure 3. In short, some filtering technique is applied to the data to extract noise. Subsequently, a clustering strategy can be used to divide the antennas in subsets according to their traffic load similarity. One model for each cluster can be designed. Then, with the time-series of a specific antenna and the information of the cluster it belongs to, it is possible to perform the prediction. Each one of these steps is explained below.

1) TIME-SERIES DECOMPOSITION

Many of the papers that use ARIMA models for forecast, first apply time-series decomposition. This can be performed in several ways. For completeness, we report here the method introduced in Brockwell and Davis [27], and adopted for instance in Xu et al. [34] and Zhang et al. [35]. This method is useful when the time-series has a small trend and we may assume that the trend within each period is constant. The time-series can be described as $T = \{x_1, x_2, \ldots, x_n\}$ with each x_t representing the traffic volume in the t_{th} time slot.

TABLE 3. Time-series.

t	x_t	$\hat{m}_t$	$\hat{s}_t$
1	60	-	27
2	96	-	97
3	49	$\frac{0.5\times60+96+49+40+0.5\times108}{4} = 67.25$	-42.5
4	40	81.25	-81.5
5	108	92.75	27
6	160	98.75	97
7	77	107.25	-42.5
8	60	121.25	-81.5
9	156	132.75	27
10	224	138.75	97
11	105	147.25	-42.5
12	80	161.25	-81.5
13	204	172.75	27
14	288	178.75	97
15	133	187.25	-42.5
16	100	201.25	-81.5
17	252	212.75	27
18	352	$\frac{0.5\times100+252+352+161+0.5\times120}{4} = 218.75$	97
19	161	-	-42.5
20	120	-	-81.5

Then, the time-series can be written as

$$x_t = m_t + s_t + r_t, \quad t = 1, 2, \ldots, n \tag{32}$$

with:

- m_t = general trend of the series;
- s_t = periodic patterns of the series;
- r_t = stochastic component of the series.

The trend parameter $\hat{m}_t$ is estimated by applying a moving average filter, as follows

$$\hat{m}_t = \begin{cases} (0.5x_{t-q} + x_{t-q+1} + \ldots + x_{t+q-1} + 0.5x_{t+q})/S, \\ \text{for} \quad S = 2q, q < t < n-q. \\ \\ (x_{t-q} + x_{t-q+1} + \ldots + x_{t+q-1} + x_{t+q})/S, \\ \text{for} \quad S = 2q+1, q+1 < t < n-q. \end{cases} \tag{33}$$

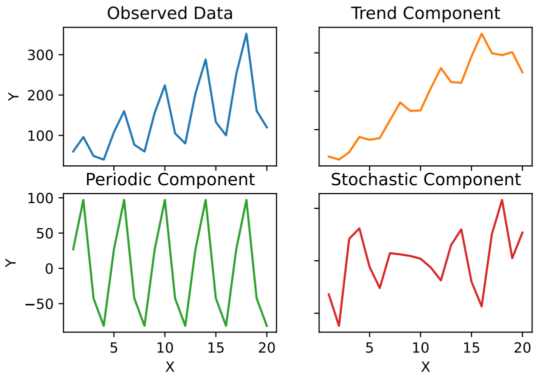


FIGURE 4. Time-series decomposition into 3 components.

The next steps are to calculate the deviation series

$$\{(x_{k+jS} - \hat{m}_{k+jS}), q < k + jS \leq n - q\}, \quad (34)$$

for $k = 1, 2, \ldots$, and calculate the average w_k of this series. Then, the periodic pattern s_k is computed as

$$\hat{s}_k = \begin{cases} w_k - S^{-1} \sum_{i=1}^{S} w_i, & k = 1, \ldots, S, \\ \hat{s}_{k-S}, & k > S. \end{cases} \quad (35)$$

The de-seasonality series d_t can be calculated by removing the periodic component from the original series, as

$$d_t = x_t - \hat{s}_t, \quad t = 1, 2, \ldots, n. \quad (36)$$

Then, the general trend can be computed by applying a moving average filter on d_t. As we want to observe the data from $0 < t \leq n$, define $x_t = x_1$ for $t < 1$ and $x_t = x_n$ for $t > n$, and

$$m_t = (2q + 1)^{-1} \sum_{j=-q}^{q} x_{t+j}. \quad (37)$$

At last, the residual component is

$$r_t = x_t - s_t - m_t, \quad t = 1, 2, \ldots, n. \quad (38)$$

Illustrative example (time-series decomposition)

By using equations (32)-(38) and $S = 4$, the time-series in Table 3 can be decomposed as

$$w_1 = \frac{x_5 - \hat{m}_5 + x_9 - \hat{m}_9 + x_{13} - \hat{m}_{13} + x_{17} - \hat{m}_{17}}{4},$$
$$w_2 = \frac{x_6 - \hat{m}_6 + x_{10} - \hat{m}_{10} + x_{14} - \hat{m}_{14} + x_{18} - \hat{m}_{18}}{4},$$
$$w_3 = \frac{x_7 - \hat{m}_7 + x_{11} - \hat{m}_{11} + x_{15} - \hat{m}_{15}}{3},$$
$$w_4 = \frac{x_8 - \hat{m}_8 + x_{12} - \hat{m}_{12} + x_{16} - \hat{m}_{16}}{3}.$$

$w_1 = 27.25$, $w_2 = 97.25$, $w_3 = -42.25$, and $w_4 = -81.25$. With all these values and equations (36), (37), and (38), the decomposition is done and the result is presented in Figure 4.

2) TIME-SERIES CLUSTERING

When the time-series of each tower is decomposed, it is possible to use the trend and periodical components to identify groups of base stations with similar traffic demand patterns. For each group, a separate forecasting model is designed. Hence, clustering algorithms represent an important subject for predicting models.

In many works, e.g. Vujičić et al. [36] and Trajković [37], K-means is used. Basically, this is an iterative approach that divides the data set into K pre-defined clusters. Subsequently, each element of the data set is assigned to only one cluster in such a way that the sum of the squared distances between all elements of the data set and their respective cluster's centroid is minimized. New centroids for each cluster are computed and the assignments of the elements are updated. This procedure repeats until the algorithm converges or other stop criterion is attended.

Another strategy is Hierarchical clustering. Proposed in Corpet [38], the basic idea of this algorithm is to consider each element as a cluster and merge the nearest two until some stop condition is achieved. The distance measure can be calculated via Davis-Bouldin Index (DBI) as in Zhang et al. [35], or via correlation coefficient between two time-series of aggregated traffic as in Cici et al. [39].

The base stations could also be clustered according to some specific characteristic, as their geographical position or traffic consumption level. For instance, Miao et al. [40] classified each antenna according to this last criterion, where the traffic load was used to label it as heavy load (HL), normal load (NL), and light load (LL).

Time-series clustering is not a trivial task due to the usual big dimension of the problem (long sequences of data), and some decisions that may vary from one application to another, as: appropriate representation method, suitable distance measure technique and an adequate clustering algorithm. According to Aghabozorgi et al. [24], comparisons between time-series can be performed in three different ways:

- **shape-based:** two time-series are considered similar if they have similar shapes. Through non-linear stretching of time axes, the sequential data are aligned and then standard clustering algorithms with specific distance measurements are applied;
- **feature-based:** vectors containing features of the time-series (standard deviation, mean, variance, maximum/minimum values, etc) are used as input to standard clustering algorithms. This approach has the advantage of dealing with low dimension vectors, since very long sequences are represented by their respective features;
- **model-based:** parametric models representing each time-series are designed. Subsequently, a clustering strategy is applied to the parameters of the models.

The algorithm strategy and distance measure choices may impact the clustering quality. In time-series comparison context, a widely used metric is the well known Euclidean distance (ED). For instance, consider $Y = (y_1, \ldots, y_m)$,

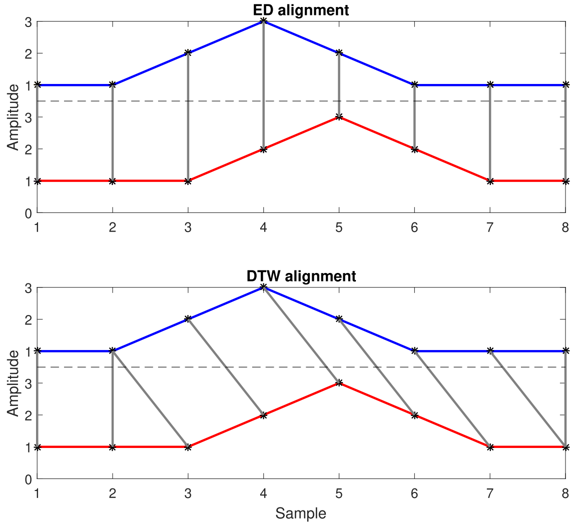


FIGURE 5. Difference between ED and DTW alignment.

$Z = (z_1, \ldots, z_m)$, then ED is simply defined as

$$ED(Y, Z) = \sqrt{\sum_{i=1}^{m} (y_i - z_i)^2}. \tag{39}$$

However, this approach has some issues: in case the time-series present the same shape and amplitude, but different phases, ED may be large, leading to a bad clustering result. Additionally, the time-series should have the same length. To circumvent these problems, elastic distance metrics, as Dynamic Time Warping (DTW), can be used, see Ding et al. [41] and Rakthanmanon et al. [42]. DTW is calculated by computing a cost matrix M_c whose elements $m_c(i, j)$ are

$$\begin{aligned} m_c(i,j) &= ED(y_i, z_j) + \min\{m_c(i-1, j-1), \\ &\quad \times m_c(i-1, j), m_c(i, j-1)\} \end{aligned} \tag{40}$$

and

$$DTW(Y, Z) = \min \sqrt{\sum_{i=1}^{k} v_i}, \tag{41}$$

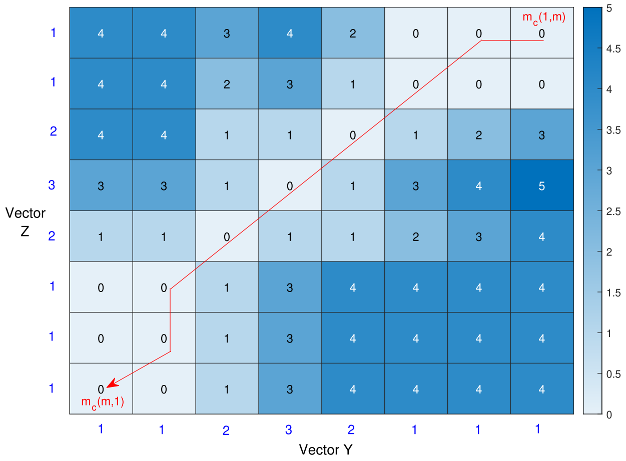
with $V = (v_i, \ldots, v_k)$ being vectors with the elements from matrix M_c forming paths from $m_c(m, 1)$ to $m_c(1, m)$. To illustrate, consider two vectors representing time-series:

- $Y = \{1,1,2,3,2,1,1,1\}$;
- $Z = \{1,1,1,2,3,2,1,1\}$.

It is easy to see that Z is just a shifted (in time) version of Y. The alignments obtained with ED and DTW distances are presented in Figure 5. Note that in this case the time-series have the same length, but this is not necessary to calculate DTW distance.

In Figure 6, it is possible to see a vector V such that equation (41) is satisfied.

For this example, $ED(Y, Z) = 2$ and $DTW(Y, Z) = 0$, presenting the efficacy of DTW to detect similar time-series even



1	4	4	3	4	2	0	0	0
1	4	4	2	3	1	0	0	0
2	4	4	1	1	0	1	2	3
3	3	3	1	0	1	3	4	5
2	1	1	0	1	1	2	3	4
1	0	0	1	3	4	4	4	4
1	0	0	1	3	4	4	4	4
1	0	0	1	3	4	4	4	4
	1	1	2	3	2	1	1	1

FIGURE 6. Cost matrix M_c considering DTW alignment.

when they have different phase. *ED* and *DTW* are between the most used time-series distance measures, however many other approaches exist, for instance: Hausdorff distance, Hidden Markov Model based distance, Longest Common Subsequence, Spatial Assembling distance, Edit Distance with Real Penalty. The reader may find works with comparison and applications of these methods in Zhang et al. [43] and Wang et al. [44].

According to Paparrizos and Gravano [45], an ideal *shape-based* approach is capable of placing time-series inside the same cluster if they have similar shape, regardless differences on both signal amplitude and phase. The authors propose a time-series clustering algorithm known as *k-Shape*, that is domain independent and preserves the time-series shapes while comparing them. To accomplish this task, they use a distance measurement, named *shape-based distance (SBD)*, that is invariant to scaling and shifting and depends on the cross-correlation function of the time-series to be compared. The approach was evaluated on 48 data sets, compared to other techniques and it outperformed many of them in accuracy.

Paparrizos and Gravano [45] define k-shape as a non-trivial k-means based procedure: in each iteration, the algorithm updates the clusters by assigning each time-series to its closest centroid; subsequently, new centroids are computed. These steps repeat until some stop criterion is reached, as convergence or the maximum number of iterations. The differences from k-means are on distance measurement (SBD) and centroid computations: while k-means uses the arithmetic mean of the coordinates of all sequences to compute an average sequence, k-shape computes the centroids as a minimization problem whose objective function is the sum of squared distances to all others time-series.

Surveys concerned with time-series clustering are presented in Warren Liao [46] and Aghabozorgi et al. [24]. The authors discuss time-series representation, similarity and dissimilarity measures, clustering algorithms and their taxonomy, and evaluation metrics for the clusters.

TABLE 4. ACF and PACF behavior for different models.

Model	ACF	PACF
AR(p)	Gradually decay	Cuts off after p lags
MA(q)	Cuts off after q lags	Gradually decay
ARMA(p,q)	Gradually decay	Gradually decay

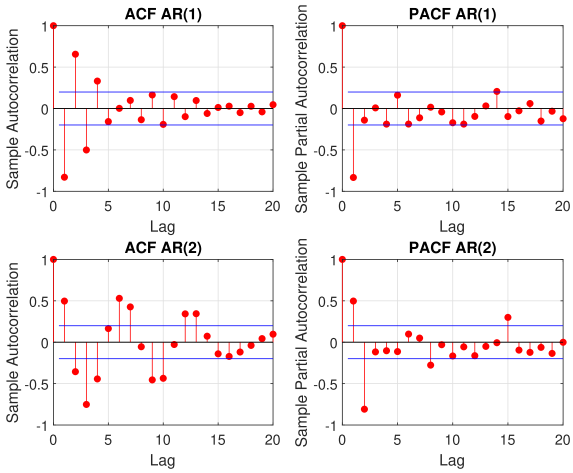


FIGURE 7. Estimating *p* and *q* with ACF and PACF.

3) MODEL CHOICE AND PARAMETERS p, q, P, Q, S DEFINITION

Autocorrelation function (ACF) and partial autocorrelation function (PACF) are widely used to determine the order of the AR and MA components of the models, as in Xu et al. [34], Suarez et al. [47], and Vujičić et al. [36]. ACF describes the autocorrelation between an observation and a preceding observation considering direct and indirect information. On the contrary, PACF considers only the direct relation between a value and the previous one. The behavior of both functions regarding a specific model is shown in Table 4.

Illustrative example (ACF/PACF use)

Consider 6 models: AR(1), AR(2), MA(1), MA(2), ARMA(1,1), and ARMA(2,2) respectively described by

$$\begin{aligned}
x_t &= 10 - 0.9x_{t-1} + \varepsilon_t,\\
x_t &= 4 + 0.9x_{t-1} - 0.8x_{t-2} + \varepsilon_t,\\
x_t &= 10 + \varepsilon_t + 1.1\varepsilon_{t-1},\\
x_t &= 5 + \varepsilon_t - \varepsilon_{t-1} + 0.9\varepsilon_{t-2},\\
x_t &= 10 - 0.9x_{t-1} + \varepsilon_t + 1.1\varepsilon_{t-1},\\
x_t &= 5 - 0.9x_{t-1} - 0.8x_{t-2} + \varepsilon_t - \varepsilon_{t-1} + 0.9\varepsilon_{t-2}.
\end{aligned}$$

Figures 7, 8, and 9 show exactly the behavior proposed above. Note that for ARMA models, the plots may give just an approximate idea. Besides, these analyses require stationary time-series. Hence, differencing, as proposed before, may be necessary.

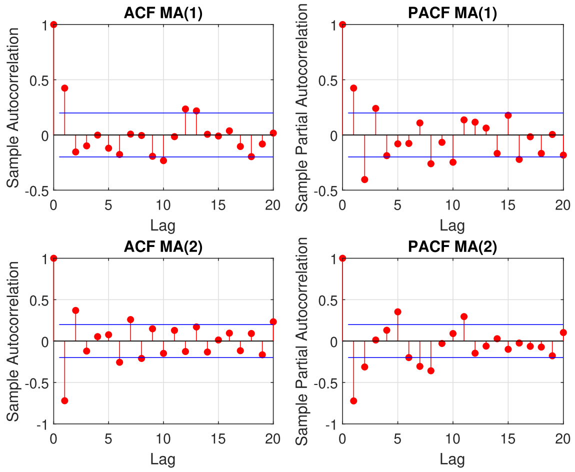


FIGURE 8. Estimating *p* and *q* for Moving Average models.

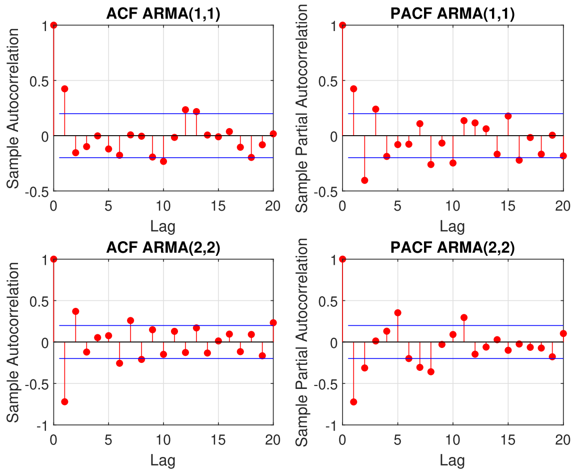


FIGURE 9. Estimating *p* and *q* for ARMA models.

The problem to determine these parameters for ARIMA and SARIMA model is similar, one should just apply the necessary differencing to make the time-series stationary, as done in Kumar and Vanajakshi [48] and Miao et al. [40]. For the SARIMA case, we also need to fit S, P and, Q. S can be set by just plotting the time-series and observing the seasonality of the signal or via spectral analysis, as done in Shu et al. [49]. For instance FFT, which is an algorithm to compute the Discrete Fourier Transform of vectors of data, may be used. Subsequently, P and Q are set by analyzing ACF and PACF behaviors in the lags that correspond to $S, 2S, 3S \ldots$.

As an example, consider a model describing a time-series that becomes stationary after two operations: one-order differencing and one seasonal differencing with $S = 6$. For the stationary series, the PACF plot gradually decays and the ACF plot has the following characteristics:

- one significant component at lag 1;
- significant components at lags 6 and 12.
- all other components are not significant.

Hence, a model that could fit the time-series is described as SARIMA$(0, 1, 1)(0, 1, 2)_6$. p, q, P, and Q could also be set by performing a grid search, where the combination that leads to the smallest Akaike Information Criterion (AIC), that considers the model simplicity and its maximum likelihood estimation, is the chosen one. In fact, a good practice is first to analyze ACF and PACF plots for having an approximation of the parameters values, and later perform a grid search varying them a little to choose the best model according to AIC.

Consider that $x_t^{(j)_K}$ denotes the traffic load at tower j at time t for a specific cluster K with n_k base stations. One may compute the average traffic time-series $T_k = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$, where

$$\hat{x}_t = \frac{1}{n_k}\sum_{j=1}^{n_k} x_t^{(j)_K}, \qquad t = 1, \ldots, n. \tag{42}$$

4) PREDICTION MODELS

The resulting series in equation (42) can be used to train a model corresponding to a cluster, as proposed in Figure 3. In this part, a study with different approaches to determine $\hat{x}_t$ can be done, since the average value may not be the best one. For instance, in some situations the cost associated to forecasting lower traffic load than the real one may be higher than the opposite, then one could use the biggest traffic load at each instant to train a model, as:

$$\hat{x}_t = \max x_t^{(j)_K}, \quad j = 1, \ldots, n_k, \quad t = 1, \ldots, n. \tag{43}$$

Hence, according to the time-series of a specific base station, it can be classified and then the assigned model is used to make the predictions. Subsequently, the quality of predictions may be measured by means of well-known metrics between real value and predicted one: mean absolute error, mean squared error, relative error, etc.

Note that training models for each base station may lead to more accurate results. Nevertheless, this can be inappropriate since the number of antennas may be very large.

IV. NEURAL NETWORKS

Although SARIMA models are an important tool regarding the forecast of time-series, in the past few years many papers work in such subject with a different and powerful approach: artificial neural networks (ANN). In fact, as introduced by Hornik et al. [50], standard multi-layer feed-forward networks using arbitrary squashing functions are universal approximators of a very large class of functions. This strategy may lead to more accurate results, since it allows a nonlinear approach that uses not just past values (as SARIMA) of a time-series, but also other types of information. For instance, Barabas et al. [51] compares network traffic prediction evaluated with statistical time-series models (as ARMA) and ANNs. The authors show that the approaches based on neural networks yield better results. Similar results are obtained in Azzouni and Pujolle [52], where ANN based approaches outperform linear ones; besides, ANN models based on LSTM (explained below) lead to even more accurate models.

A. A BRIEF INTRODUCTION TO NEURAL NETWORKS

Basically, a neuron is a unit that receives an input vector $V = [v_i, \ldots, v_q]$ associated to weights w_{ij}^l (weight associated to j_{th} neuron in layer l connected to i_{th} neuron in layer $l+1$). The neuron then applies a function on the weighted sum of the inputs to compute the output y. Subsequently, y can become the input of a new layer of neurons forming a neural network, which is a function $f : \mathbb{R}^g \rightarrow \mathbb{R}^q$ as represented in Figure 10, with g and q being input and output size, respectively. Note that there are no cycles or loops in the network, so this is called *feed-forward neural network* (FFNN). In case there are 2 or more hidden layers, then we have a *deep feed-forward neural network* (DFFNN).

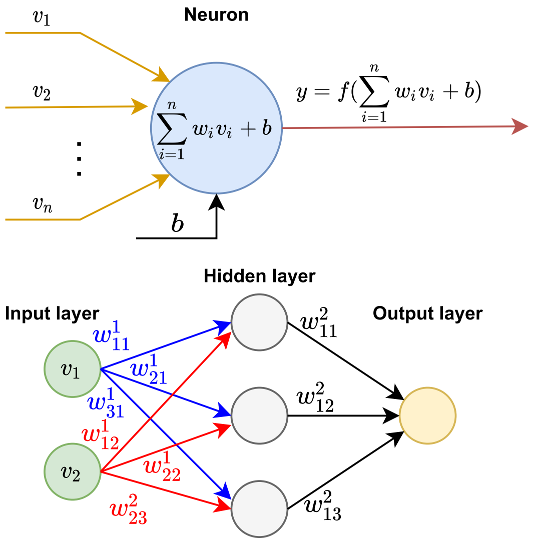


FIGURE 10. Neuron and a simple neural network.

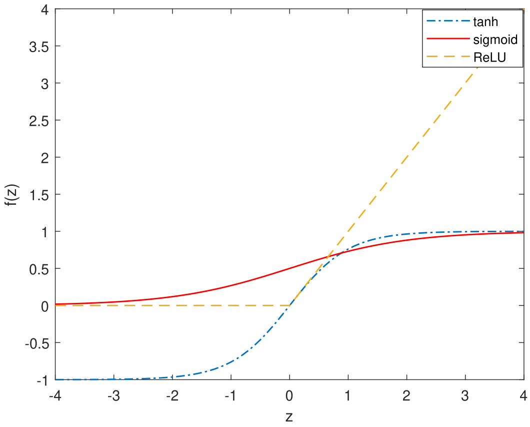


FIGURE 11. Nonlinear activation functions.

$f : \mathbb{R} \rightarrow \mathbb{R}$ in Figure 10 is called activation function, and the most used are *sigmoid*, *hyperbolic tangent*, and *rectified linear unit* (*ReLU*), shown in Figure 11. Here, b is a scalar representing a bias term.

The idea is that the weights associated with a neuron can be learned in the training phase by minimizing a loss function L, that measures how much the network output is different from the desired one. The procedure to minimize L is based on the gradient descent algorithm

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial w_i}, \tag{44}$$

and due to the non-convex behavior of the neural network, the global minimum is not guaranteed. The partial derivatives in equation (44) are usually calculated via *backpropagation algorithm* and η represents the learning rate. Small values for η can lead to time consuming training and big ones could make the procedure diverge.

The training phase also depends on the weights initialization. As proposed in Glorot and Bengio [53], a good practice is to use the *Glorot initialization*, where each initial weight is given by a zero mean Gaussian distribution with variance given by

$$\sigma^2 = \frac{2}{fan_{in} + fan_{out}}, \tag{45}$$

where fan_{in} and fan_{out} are numbers of inputs and outputs to a layer, respectively.

B. NEURAL NETWORKS AND TIME-SERIES FORECASTING

An important subject when designing a neural network is its architecture. Some types of ANN, as the feed-forward, have the vanishing gradient problem, which happens when the partial derivatives in equation (44) become very small during the back propagation algorithm, leading to little actualization of the weights w_i before a local minimum is achieved. Besides, the previous values of sequential data are not considered to predict the next one. To circumvent these problems, papers dealing with time-series prediction (values from previous steps must be considered in regression problems) make use of some different architectures, such as RNN and LSTM, presented below.

1) RECURRENT NEURAL NETWORKS

Recurrent neural networks (RNN), different from feed-forward ones, have the property of cycling the output from layers back to the network. Hence, the effect of initial values on the later ones of a sequential data can be modeled by RNNs. In Ramakrishnan and Soni [54], this approach is used to predict three network traffic parameters: volume, protocol classification and protocol distribution. In summary, the main difference between RNN and FFNN is the presence of feedback in hidden layers, as shown in Figure 12. Assume that the input is $a_t \in \mathbb{R}^{m\times 1}$ and $h_t \in \mathbb{R}^{p\times 1}$ is the hidden state (or memory) of t_{th} time step.

Therefore

$$h_t = \tanh(Wa_t + Uh_{t-1} + b), \tag{46}$$

where $W \in \mathbb{R}^{p\times m}$, $U \in \mathbb{R}^{p\times p}$, and $b \in \mathbb{R}^{p\times 1}$ are the model parameters to be determined during training. It is important to emphasize that there is just one hidden layer in Figure 12, represented at different time steps, and the estimated parameters are the same for all time steps.

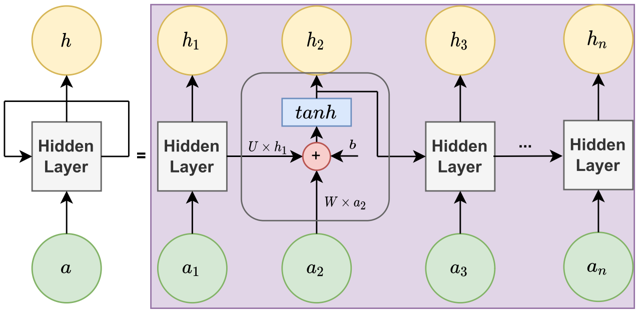


FIGURE 12. Recurrent neural network layer and inside cells operation.

Despite the fact that RNNs provide better results in prediction of sequential data when compared to FFNN, they still suffer from the vanishing gradient problem for very long sequences. Hence, special cases of RNNs were designed to retain information even in such cases, as LSTM and GRU.

2) LSTM

In Wang et al. [55] and Li et al. [56], Long Short-Term Memory (LSTM) neural networks are used to predict traffic in cellular networks and traffic flow in transportation systems, respectively. Introduced by Schmidhuber [57], LSTMs are specialized in predicting time sequences due to its cell architecture: three gates, presented below, that update the cell state and its hidden state:

- **forget gate:** decides if an information is relevant or should be "forgotten". The inputs of such gate, which are the values of current input a_t and previous hidden state h_{t-1}, pass through a sigmoid function to squeeze the values from 0 to 1, resulting in f_t. Subsequently, there is a pointwise product between f_t and previous cell state c_{t-1}, where values of c_{t-1} considered irrelevant are multiplied by values close to 0 of f_t, concluding the forget step;
- **input gate:** here, there is a pointwise product between i_t and $\tilde{C}_t$ (candidate values that could be considered relevant and added to cell state). $\tilde{C}_t$ is the hyperbolic tangent function (tanh) of a_t and h_{t-1} (with respective weight matrices), squeezing these values between -1 and 1. tanh is used to regulate the values inside the cell, avoiding them to become very large after the pointwise products. Later, the output of this step is calculated by a sum between the output of forget gate and the pointwise product between i_t and $\tilde{C}_t$.
- **cell state update:** the updated cell state c_t is the output of the previous step; c_t saves relevant information even from the earliest steps.
- **output gate:** this gate determines the next hidden state, which contains relevant information about the previous steps. a_t and h_{t-1} (with respective weight matrices) pass through a sigmoid function and is pointwise multiplied by the hyperbolic tangent of c_t. The result is the new hidden state h_t.

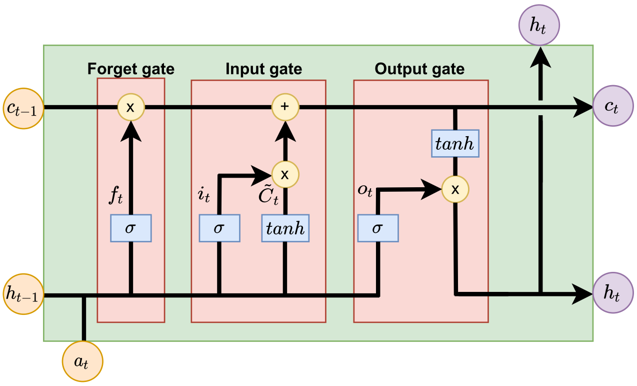


FIGURE 13. Operations inside LSTM cell.

Such process is represented in Figure 13. Assume that the input $a_t \in \mathbb{R}^{m \times 1}$ are vectors fed into the LSTM layer at each time step, and that the output at each iteration is $o_t \in \mathbb{R}^{p \times 1}$. Note that the input vector can be different at each t, and both c_t and h_t allow relevant information to be considered in subsequent LSTM cells. Then,

$$\begin{aligned} f_t &= \sigma(W_f a_t + U_f h_{t-1} + b_f), \\ i_t &= \sigma(W_i a_t + U_i h_{t-1} + b_i), \\ \tilde{C}_t &= \tanh(W_c a_t + U_c h_{t-1} + b_c), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{C}_t, \\ o_t &= \sigma(W_o a_t + U_o h_{t-1} + b_o), \\ h_t &= o_t \odot \tanh(c_t), \end{aligned} \tag{47}$$

with all matrices W $(W_f, W_i, W_c, W_o) \in \mathbb{R}^{p \times m}$, all bias b $(b_f, b_i, b_c, b_o) \in \mathbb{R}^{p \times 1}$, and all matrices U (U_f, U_i, U_c, U_o) $\in \mathbb{R}^{p \times p}$ being the parameters to be determined during the training phase. Besides $h_t \in \mathbb{R}^{p \times 1}$, $c_t \in \mathbb{R}^{p \times 1}$, $f_t \in \mathbb{R}^{p \times 1}$, $i_t \in \mathbb{R}^{p \times 1}$, and $\tilde{C}_t \in \mathbb{R}^{p \times 1}$.

3) GRU

Another type of RNN is the *Gated Recurrent Units (GRU)*. Introduced by Cho et al. [58], GRUs can be seen as a simple version of LSTMs, in the sense they have 2 gates, update and reset, and according to the authors are much simpler to compute and implement. In Patil et al. [59], IoT traffic prediction is developed with the use of GRUs, where the results are shown to be more accurate than the ones obtained via ARIMA modeling. In Fu et al. [60], GRU and LSTM NN are used to forecast traffic flow in the state of California; similarly to the previous paper, both approaches presented more accurate results than autoregressive moving average models. The architecture of a GRU cell is shown in Figure 14.

- **update gate:** it is responsible for controlling how much information from previous hidden state will be considered by the current one;
- **reset gate:** when its value is close to 0, the current hidden state is forced to ignore information from previous ones and then resets with only the information from the input.

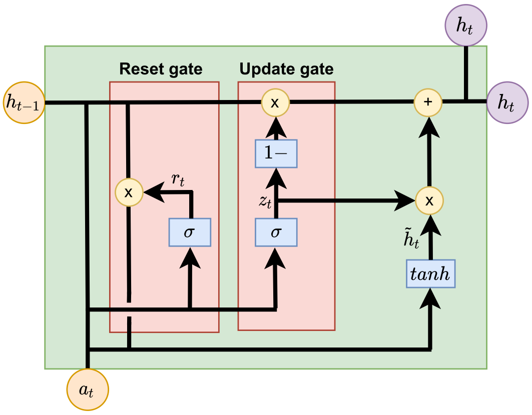


FIGURE 14. Operations inside GRU cell.

Then, considering the same input dimensions previously defined ($a_t \in \mathbb{R}^{m \times 1}$ and $h_t \in \mathbb{R}^{p \times 1}$)

$$\begin{aligned} z_t &= \sigma(W_z a_t + U_z h_{t-1} + b_z), \\ r_t &= \sigma(W_r a_t + U_r h_{t-1} + b_r), \\ \tilde{h}_t &= \tanh(W_h a_t + U_h (r_t \odot h_{t-1}) + b_h), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \end{aligned} \tag{48}$$

with all matrices W $(W_z, W_r, W_h) \in \mathbb{R}^{p \times m}$, all bias b $(b_z, b_r, b_h) \in \mathbb{R}^{p \times 1}$, and all matrices U $(U_z, U_r, U_h) \in \mathbb{R}^{p \times p}$ being the parameters to be determined during the training phase. Besides $z_t \in \mathbb{R}^{p \times 1}$, $r_t \in \mathbb{R}^{p \times 1}$, $f_t \in \mathbb{R}^{p \times 1}$, $\tilde{h}_t \in \mathbb{R}^{p \times 1}$, and $h_t \in \mathbb{R}^{p \times 1}$. z_t and r_t are update and reset gates, respectively.

4) CONVOLUTIONAL NEURAL NETWORKS

Another common approach used to predict time-series is based on Convolutional Neural Networks (CNNs). CNNs have a structure that allows them to identify local patterns in the feature space of a matrix [3]. In Bega et al. [16], a distance matrix based on the similarity between the time-series of base stations was constructed, and then a forecast procedure based on CNN was developed. Note that with the distance matrix, local patterns become an important subject to prediction. Traffic flow prediction is developed in Chen et al. [61] also with the use of CNNs. Convolutional neural networks have also big applicability in researches that work with image and video processing, since these two can be seen as matrices with high local patterns.

In brief, a convolutional layer can be understood as a filter, also called Kernel, that slides in spatial dimension of an input and then produces the output, as presented in Figure 15. Note that the dimension of the output is smaller. This operation is known as *valid convolution* and the Kernel does not go outside the matrix border. On the contrary, there are the *zero padding*, where everything outside the border is set to 0, and *symmetric padding*, where the terms beyond the border are

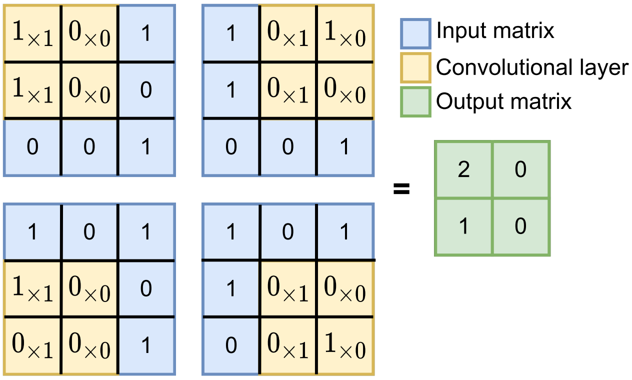

$1_{\times1}$ $0_{\times0}$ 1
$1_{\times1}$ $0_{\times0}$ 0
0 0 1
1 $0_{\times1}$ $1_{\times0}$
1 $0_{\times1}$ $0_{\times0}$
0 0 1
1 0 1
$1_{\times1}$ $0_{\times0}$ 0
$0_{\times1}$ $0_{\times0}$ 1
1 0 1
1 $0_{\times1}$ $0_{\times0}$
0 $0_{\times1}$ $1_{\times0}$
=
2 0
1 0
Input matrix
Convolutional layer
Output matrix

FIGURE 15. Operation with convolutional layer: valid convolution.

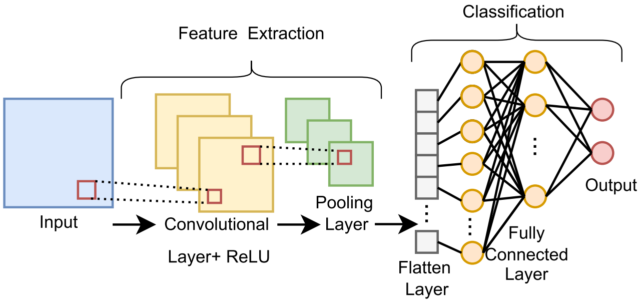


FIGURE 16. Standard architecture of a CNN.

mirrored. In both cases, the dimension of the input can be kept or increased.

After the operation with the convolutional layer, *ReLU* is applied, and then the pooling layer may be used. It reduces the size of the resulting matrix by using the *max pooling* or *average pooling*. The first approach constructs an output matrix by considering the maximum values of different regions of the input, while the second averages these values. Convolutional, *ReLU*, and pooling form one layer of a convolutional neural network. The output may have a smaller dimension and contains local patterns of the input matrix.

A big advantage of CNNs is that, different from Figure 10 where all nodes of the input layer are fully connected to the nodes of the hidden one, neurons between input and hidden layers or between each hidden layer may be connected just to some in the next layer, since CNNs need to identify localized features. This reduces the number of parameters to be determined during the training. Besides, different Kernels can be used in a single convolutional layer, where the resulting matrices (called feature maps) are stacked, and later flattened. The case with 3 Kernels is illustrated in Figure 16. Note that the output of the pooling layer could be the input of another convolutional layer. This procedure would lead to higher feature extraction, since each CNN layer is trained to detect a specific pattern.

Similar analysis can be extended to 3-D cases, where the concept of tensors is used.

Many papers that apply ANNs to predict time-series, as Ong et al. [62] and Deng et al. [63], use tensors. Basically, they are generalization of scalars (rank-0 tensor), vectors (rank-1 tensor), and matrices (rank-2 tensor). For instance, a rank-3 tensor can be seen as a 3-D matrix, which is very useful for traffic load prediction: consider a tensor Λ whose elements are λ^{ijk}, where i and j are related to the geographical position of a tower, and λ is its traffic load at time step k; this is a way to store spatio-temporal information of all towers that belong to a network (this is done in Bega et al. [16]).

5) GRAPH NEURAL NETWORKS

In many of the data sets containing information about the network traffic demand, the location of towers, routers, or switches are also available. Hence, the entire network can be seen as a graph whose nodes and links are the hardware generating the network and their connections, respectively. In recent years, Graph Neural Networks (GNNs), introduced by Scarselli et al. [64], are on the rise and have provided results superior to the aforementioned approaches when dealing with road traffic forecasting problems. As discussed by iang and Luo [4], GNNs have the property of learning directly from graphs, which makes them capable of extracting features from data represented by these structures.

In Wu et al. [65], an extensive survey on GNNs is developed. The authors propose a taxonomy, dividing GNNs into four groups, briefly commented below.

- **Recurrent GNNs:** also called RecGNNs, they are the first type of GNNs. In summary, it is considered that two neighbor nodes in a graph exchange information until some stop criterion or convergence is reached. The node representation learning is based on RNNs.
- **Convolutional GNNs:** known as ConvGNNs, they extend the idea of convolution from matrix (or tensor) to graph. Similarly to RecGNNs, a node representation is given by its features and its neighbors ones. Subsequently, multiple graph convolutional layers can be used to learn the graph main features.
- **Graph Autoenconders:** unsupervised learning that, different from the two aforementioned approaches, encodes the graph in a vector space, from where features can be learned and subsequently a decoder reconstructs the graph.
- **Spatio-temporal GNNs:** these GNNs consider space and time features simultaneously. They can combine graph convolutional layers with CNN and RNN.

It is important to emphasize that the GNNs above have different mathematical formulations and applications, where the details of each case are also discussed by Wu et al. [65].
’

In Wang et al. [66], the authors propose a type of GNN, named Time-Series Graph Attention Network (TSGAN), with the previously discussed elastic distance metric DTW to forecast the cellular network traffic in a real world data set. The prediction results obtained with TSGAN outperformed three standard GNNs and a GRU model in short-term, mid-term, and long-term scenarios. In Zhou et al. [67], a Spatio-

TABLE 5. Summary of limitations and advantages of techniques previously discussed.

	Technique	Limitations		Advantages	

Statistical	ARMA	Stationary data;
		Non-periodical data;
			Data prediction based exclusively on past values;	Mathematical simplicity;	High parameters interpretability;

	ARIMA	Non-periodical data;		Deals with non-stationary data;	
	SARIMA	Parameters definition complexity;		Deals with non-stationary and seasonal data;	

Neural Networks	NN	Vanishing gradient descent;	Low parameters interpretability;
			High number of hyper-parameters to set;
			Non-convex;
			High computational cost in training phase;
				Universal function approximator;	In general, better results when compared to statistical approaches;
					They can be used simultaneously in different layers to extract spatio-temporal features.

	RNN	Vanishing gradient descent for long sequences;		Feedback in hidden layers;	
	LSTM	Time consuming training;			
	GRU				

	CNN	Sizable training data set required;		Spatio-temporal pattern recognition;
				Localized features identification;	

	GNN	Difficulty to deal with heterogeneous graphs;		Direct learning from graphs;	

temporal Graph Convolutional Neural Network is presented and leads to better results than many state-of-art forecasting models: historical averaging, vector autoregressive, LSTM, ConvLSTM, GCN-CNN, attention mechanism GCN-CNN, and diffusion CNN-RNN.

Some characteristics of the approaches discussed above are shown in Table 5.

V. EXTENDED BIBLIOGRAPHY

In this Section, we present, in Table 6, a detailed list of papers using the different approaches discussed in the previous sections for network traffic prediction. In each row, the reader finds the authors, year of publication, the prediction technique, training/testing data set, and evaluation metrics for the respective paper. Besides, the Table is divided into three subgroups:

- Yellow subgroup: Autoregressive Moving Average based approaches;
- Blue subgroup: RNN, LSTM, GRU, or CNN based approaches;
- Green subgroup: diversified approaches.

The error acronyms are given in Table 2.

VI. NUMERICAL EXPERIMENTS

To compare many of the approaches described above and used in many of the previously cited papers, we apply SARIMA and RNN models on mobile phone activity data set from the city of Milan. The data set contains information about received and sent SMS, incoming and outgoing calls, internet activity, and geographical position of all antennas. Each of these operations is named as call detail record (CDR). Our goal is to predict the hourly aggregate CDRs for one week of cell ID 1051, based on previous 55 days traffic load. The hyper parameters of the numerical experiments performed with the approaches discussed in the survey were defined according to the authors' criteria, described below for each forecasting technique

A. SARIMA

The results presented below regarding the SARIMA models were obtained under the following assumptions:

- models with all possible combinations of parameters $p, d, q, P, D, Q = \{0, 1, 2\}$ were fitted;
- the model with the smallest AIC is used to predict the CDR traffic behavior;
- $S = 24$ (time-series with period of one day).
- the training data set is standardized according to

$$\tilde{a}_t = \frac{x_t - \mu}{\sigma}, \tag{49}$$

with μ and σ being the mean and standard deviation of the samples belonging to the training data set.

1) REGULAR SARIMA

In this approach, the standardized time-series regarding the training data set, without any previous filtering technique, is used to design the model. After fitting all possible models according to the interval declared above, they were compared according to AIC criterion, which is a parameter that considers not just how well the model fits to its observations, but also its simplicity (number of parameters). The optimum value (minimum) for AIC is obtained for SARIMA$(2, 0, 2)(1, 1, 2)_{24}$. With these parameters, the *MAE* and *MSE* between the measured 7 days time-series and predictions given by the model for the same period are, respectively, 2.911 and 12.827.

2) SARIMA WITH DECOMPOSED TIME-SERIES

In this approach, the time-series of the respective cell ID was decomposed, Figure 17, according to equations (32)-(38) and $S = 24$. The *resulting time-series*, which was standardized

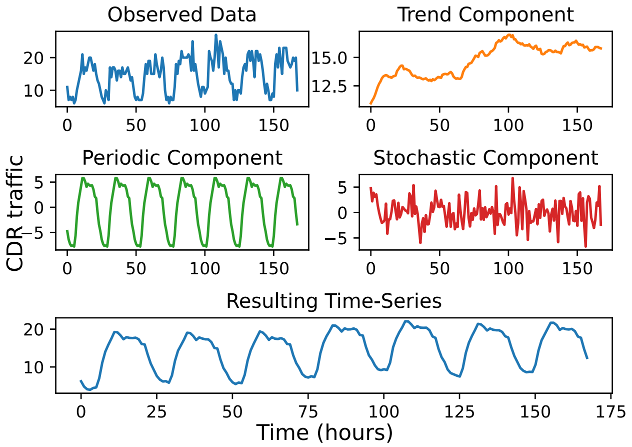


FIGURE 17. One week of measured CDRs traffic load for cell ID 1051 (observed data), its correspondents components, and filtered signal (sum of trend and periodic components).

according to equation (49) and used to fit the models, is the sum of trend and periodic components. The model with the smallest AIC is achieved for SARIMA(2, 0, 1)(1, 0, 1)$_{24}$ and provides $MAE = 2.733$ and $MSE = 11.452$ between the predictions given by the model and the real measured traffic load for the same period.

B. RECURRENT NEURAL NETWORKS

When training a neural network, some hyper parameters may affect the results, leading to over-fitting or under-fitting. As example, it is possible to cite *epochs* and *batch size*, defined as:

- *epochs (epc)*: hyper parameter that is used to determinate how many times the learning algorithm will work through the entire data set;
- *batch size (bs)*: hyper parameter that defines the number of samples that the learning algorithm should work with, before updating its internal weights. In brief, at the end of the batch, the error between prediction and output is calculated, then the gradient descent algorithm is applied to update the model's weights.

For instance, consider a data set with 100 samples, batch size set to 5, and 100 epochs, then:

1) the data set is divided into 20 batches, each with 5 samples. The model‘s weights are updated after each batch;
2) 1 epoch allows the model to update its weights and bias 20 times;
3) since there are 100 epochs, there are 2000 updates for the internal parameters of the model.

In the following subsections, the traffic load is predicted under the assumptions:

- $epc = 1000$;
- $bs = 32$;
- the number of neurons n_n in each hidden layer is $n_n = \{32, 64, 128, 256\}$;
- the number of LSTM, GRU or simple RNN hidden layers is $n_{hl} = \{1, 2, 3, 4\}$;
- tests run with Adam optimizer, introduced by [119] and proved to be computationally efficient with little memory requirements, and appropriate for non-stationary objectives;
- loss function as MSE;
- learning rate set to 0.0001;
- activation function of LSTM, GRU or simple RNN layers: tanh;
- each input vector (a_t in Figures 13 and 14) consists of 24 samples, that is, $a_t = \{\tilde{a}_{t-23}, \ldots, \tilde{a}_{t-2}, \tilde{a}_{t-1}, \tilde{a}_t\}$, where each $\tilde{a}_t$ is calculated by using equation (49). This standardization is may improve the network training;
- the algorithms have access to 55 days of CDR traffic: 50 are used to train the models and 5 days to validate them;
- the last 7 days are used just to evaluate the models performance by comparing their predictions and the real measured values.

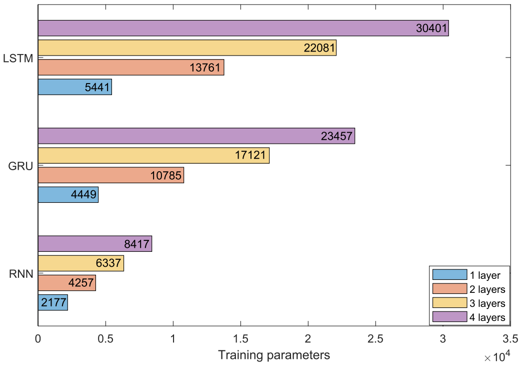


FIGURE 18. Training parameters for the three types of recurrent neural network considering different numbers of the hidden layers, each with 32 neurons.

Besides, after the last LSTM, GRU or simple RNN hidden layer, there are two layers: a flatten layer with 32 neurons and *reLU* activation function, an output layer with a single neuron (which is the predicted value) and linear activation function.

1) RNN

The results obtained with the simplest recurrent neural network, RNN, are presented in Table 7.

2) LSTM

The results obtained with LSTM recurrent NN are presented in Table 8.

3) GRU

The results obtained with GRU recurrent NN are presented in Table 9.

The previous three models have different numbers of training parameters, leading to specific computational costs. The quantity of training parameters of each approach for different numbers of layers is presented in Figure 18.

It is clear that the number of training parameters grows when more neurons are considered inside the same layer, which is illustrated in Figure 19.

TABLE 6. Traffic network prediction: detailed bibliography.

	Paper	Year	Technique	Data set	Evaluation
Autoregressive Moving Average based approaches	Shu et al. [49]	2003	Extended SARIMA.	GSM network of China Mobile Tianjin.	RE.
	Chen and Trajković [37]	2004	SARIMA; Cluster SARIMA.	Trunked radio network system from Vancouver.	NMSE.
	Papagiannaki et al. [68]	2005	ARIMA.	Incoming and outgoing link utilization of all routers in the Sprint IP backbone.	RE.
	Mao [69]	2005	Multiscale ARIMA.	LAN traffic at the University of Auckland.	NMSE; MARE.
	Vujičić et al. [36]	2006	SARIMA.	Trunked radio network system from Vancouver.	NMSE.
	Zhou et al. [70]	2006	ARIMA/GARCH.	LBL-TCP-3 traffic: TCP traffic from the Lawrence Berkeley Laboratory.	SER.

	Moayedi and Masnadi-Shirazi [71]	2008	ARIMA.	Simulated network.	MSE;
					NMSE;
					SER.

	Chen et al. [72]	2009	SARIMA.	WLAN traffic from study laboratories.	Average RE.

	Suarez et al. [47]	2009	AR;
			MA;
			ARIMA.
				Traffic load in Wi-fi network.	MSE;
					SDAV;
					CC.

	Guo et al. [73]	2009	Multiplicative SARIMA.	Network management systems of CMCC Heilongjiang Co LTD.	NRMSE; Average NRMSE.

	Christodoulos et al. [74]	2010	ARIMA with diffusion models.	World broadband and mobile telecommunications' penetration-ITU.	MSE;
					MAE;
					MAPE.

	Tan et al. [29]	2010	ARMA.	TCP network traffic volume data.	Log returns.

	Yu et al. [33]	2010	SARIMA.	Heilongjiang province in China.	MAPE;
					AE;
					APE.

	Hoong et al. [75]	2011	ARMA.	Number of packets in a private network.	MSE.
	Hoong [28]	2012	ARMA.	BitTorrent network traffic.	MSE.
	Miao et al. [40]	2014	SARIMA.	2G/3G data sets from China Mobile.	MAPE.
	Haviluddin and Rayner [76]	2014	SARIMA.	Network internet traffic from Mulawarman University.	Ljung-Box; Q statistic .

	Yoo and Sim [77]	2015	STL-ARIMA.	6 directional paths connecting 2 sites on the ESnet, US.	RMSE;
					MAE;
					ME;
					Training time.

	Xu et al. [34]	2016	SARIMA.	9000 cellular towers from Shanghai.	AVRE.
	Medhn et al. [78]	2017	SARIMA.	Ethio Telecom.	APE; MAPE.
	Zhang et al. [79]	2017	Combined SARIMA (SARIMA, Decision Tree, Spreading Model, and Top-K Regression Tree).	Traffic load of base stations from a big city of China.	ARMSE.
	Madariaga et al. [80]	2018	ARIMA; SARIMA.	Android App Adkintun Mobile.	RMSE; MAE.
	Zhang et al. [35]	2019	SARIMA.	6400 cellular towers from Shanghai.	MSE.

	Bastos [81]	2019	Random Walk;
			Linear Trend;
			Theta;
			Exponential Smoothing;
			ARIMA.
				Radio network controllers in a 3G network of an operator from northern Europe.	RMSE;
					MAPE.

	Arifin and Habibie [82]	2020	ARIMA with disruptive formula.	2G, 3G, and LTE mobile network traffic from a site in Indonesia.	PE.

	Yang et al. [83]	2021	SA ARIMA-BP.	WIDE project.	MAPE;
					MAE;
					RMSE.

TABLE 6. ***(Continued.)*** **Traffic network prediction: detailed bibliography.**

RNN, LSTM, GRU, or CNN based approaches	Oliveira et al. [84]	2016	MLP;
			SAE;
			RNN.
				Private ISP from 11 European cities.	MSE;
					NRMSE.

	Fu et al. [60]	2016	GRU; LSTM.	15000 sensors deployed statewide California: PeMS dataset.	MSE; MAE.

	Wang et al. [55]	2017	LSTM.	2844 BSs from China Mobile at Suzhou.	MSE;
					MAE;
					Log loss.

	Huang et al. [3]	2017	LSTM;
			3D-CNN;
			CNN-LSTM.
				Telecom Italia 2015.	RMSE;
					MAPE;
					MAE.

	Vinayakumar et al. [85]	2017	FFNN;
			RNN;
			LSTM;
			GRU;
			IRNN.
				1200 traffic matrices from GÉANT network.	MSE.

	Ramakrishnan and Soni [54]	2018	RNN;
			LSTM;
			GRU.
				Abilene network;
				GEANT network.
					MSE.

	Qiu et al. [86]	2018	LSTM.	16 base stations from a big city in Asia: 15 days period.	MSE.
	Hua et al. [87]	2018	LSTM; RCLSTM.	GÉANT network data set.	MSE; MAE.
	Trinh et al. [88]	2018	LSTM.	LTE scheduling information of users connected to a eNodeB.	NRMSE.
	Feng et al. [89]	2018	DeepTP (LSTM based).	9600 BS from a mobile cellular network from Shanghai, China.	NRMSE.
	Li et al. [56]	2019	LSTM.	Transportation research data laboratory from University of Minnesota.	RMSE; ACC.

	Andreoletti et al. [90]	2019	Diffusion Convolutional-RNN.	Backbone Abilene network.	MAE;
					MAPE;
					RMSE;
					Convergence Epoch;
					Convergence Time.

	Shihao et al. [91]	2019	LSTM-DNN.	Network service from 11 European cities;
				UKERNA website;
				Beijing University.
					MAPE.

	Bega et al. [16]	2019	3D-CNN.	Network datacenter with 470 4G eNodeBS;
				Mobile Edge Computing datacenters;
				C-RAN datacenters.
					Monetary cost;
					MAE;
					Overprovisioning/SLA.

	Sone et al. [92]	2020	Holt-Winters;
			SARIMA;
			LSTM;
			GRU;
			CNN;
			CNN-LSTM;
			CNN-GRU.
				470 APs in the Linnanmaa campus of the University of Oulu, Finland.	RMSE;
					MAE;
					NRMSE;
					R^2 score.

	Li et al. [93]	2020	LA-ResNet (LSTM).	Telecom Italia: Milan 2013.	RMSE; Average accuracy.

	Elsherbiny et al. [94]	2020	KNN;
			SVR;
			Ridge Regression;
			Random Forest;
			ARIMA;
			LSTM.
				Multiple network parameters from a public transit bus in Canada.	RMSE;
					R^2 score.

	Deng et al. [63]	2021	RNN	2020 MathorCup (upstream and downstream traffics).	MSOE.

	Patil et al. [59]	2021	ARIMA;
			GRU;
			LSTM.
				IoT basic-traffic.	MAE;
					RMSE;
					MSE;
					MRE.

	Alsaade and Al-Adhaileh [95]	2021	SES-LSTM.	3 months of 4G LTE network traffic in 2018.	MSE;
					RMSE;
					NRMSE;
					R^2.

	Chien and Huang [96]	2021	CNN.	Telecom Italia: Milan 2013.	MAE;
					RMSE;
					MAPE.

	Gao et al. [97]	2021	Ensemble Model; CNN.	Telecom Italia: Milan 2013.	ANRMSE.

	Wu et al. [98]	2021	GS-STN (CNN-LSTM).	Telecom Italia: Milan 2013.	NMAE;
					NRMSE;
					Training time.

TABLE 6. *(Continued.)* Traffic network prediction: detailed bibliography.

	Zhan et al. [99]	2021	CNN.	Telecom Italia: Milan 2013.	RMSE;
					MAE;
					MAPE;
					DirAcc.

	Shen et al. [100]	2021	TWACNet (CNN).	Telecom Italia: Milan 2013.	MAE;
					RMSE;
					Training time.

	Kuber et al. [101]	2021	LSTM.	Telecom Italia: Milan 2013.	RMSE; MAE.

	Lo Schiavo et al. [102]	2022	TES-RNN.	More than 400 4G/LTE base stations in a metropolitan area.	Training time;
					Average MSE;
					F1 score;
					ROC curve.

Other approaches	Hasegawa et al. [103]	2001	LLA;
			RBF;
			SVM.
				Packet trace taken from US-Japan link of WIDE backbone.	CC.

	Cortez et al. [104]	2006	Ensemble NN.	Private ISP from 11 European cities; UK academic network.	MAPE.
	Tikunov and Nishimura [105]	2007	Holt-Winters.	Commercial GMS/GPRS network.	NRMSE.
	Filho and Maia [106]	2010	PCA with K-means.	GÉANT network.	MAPE.
	Shafiq et al. [107]	2011	Zipf-like; Markov model.	Mobile device traffic data from cellular operator's core network in US.	MSE; Fit quality.
	Li et al. [108]	2014	Spatial-temporal compressive sensing.	2 weeks of traffic load of 64 BSs from Hangzhou.	NRMSE.

	Nikravesh et al. [109]	2016	MLP;
			MLPWD;
			SVM.
				5840 cells from a commercial trial mobile network.	MAPE;
					RMSE.

	Zhang and Patras [110]	2017	STN; D-STN.	Telecom Italia for Milan.	NRMSE.
	Wang et al. [111]	2017	Graph Neural Network-D; Graph Neural Network-A.	5959 cell towers covering 1.5 million users in a major city of China.	MAE; MARE.
	Li et al. [112]	2017	ADM-LARS; ADM-OMP.	Traffic records from China Mobile in Hangzhou.	NMAE.
	Narejo and Pasero [113]	2018	DBN.	UK academic internet traffic from 2004 to 2005.	MSE; RMSE.

	Stepanov et al. [19]	2020	Bagging;
			Random Forest;
			SVM.
				Traffic of a LTE network.	RMSE;
					MAE;
					R^2.

	Vinchoff et al. [114]	2020	Adversarial NN.	Complex Elastic Optical Network Simulator; Short-Term Real-life Fiber Network Data.	MSE.
	Yang [115]	2021	STFT.	Generated network traffic.	AE; RE.
	Sun and Guo [116]	2021	Feature Embedding Gaussian Process.	4G BS from a metropolitan area.	ACE.

	Ale et al. [117]	2021	Bayesian Hierarchical Learning.	China Telecom: Shanghai.	PMCC;
					MSE;
					MAE;
					MAPE;
					RMSEP;
					RMSE.

	Tomic et al. [118]	2022	Naive Model;
			Linear Regression;
			XGBoost;
			FFNN.
				Tier 1 Mobile Operator: 3000 BS operating in various LTE frequency bands.	MAPE;
					Training time.

	Wang et al. [66]	2022	Graph Neural Network.	Telecom Italia: Milan 2013.	MAE;
					MAPE;
					RMSE.

	Zhou et al. [67]	2022	Spatio-temporal Graph Convolutional Neural Network.	Telecom Italia: Milan 2013;
				Caltrans Performance Measurement System.
					MAE;
					RMSE;
					Epoch efficiency.

C. CONVOLUTIONAL NEURAL NETWORKS

The data set contains the geographical positions of all antennas, which are distributed in a matrix $M \in \mathbb{R}^{100\times100}$. As previously mentioned, CNNs are able to capture local patterns in a feature space of a matrix. Hence, one could consider all adjacent towers to the one in cell ID 1051 to predict the traffic load for one week of CDRs. In this case, the input of the CNNs is a tensor presented in Figure 20, where each tensor position is the traffic load of a specific cell ID at a time step. Notice that we have both geographical and time information as input.

For the CNN simulations, we used the same number of epochs, batch size, optimizer, loss function, and learning rate

TABLE 7. RNN forecasting errors for one week of CDRs predictions.

n_n		Number of RNN hidden layers			
		1	2	3	4
32	MAE	4.424	4.258	4.976	3.626
	MSE	26.151	33.939	42.344	22.303
64	MAE	4.622	5.958	5.126	5.370
	MSE	30.537	48.103	38.203	42.261
128	MAE	3.716	4.087	5.480	3.743
	MSE	21.320	24.592	42.603	21.744
256	MAE	2.982	4.842	**2.705**	3.863
	MSE	14.275	32.687	**10.887**	22.700

TABLE 8. LSTM forecasting errors for one week of CDRs predictions.

n_n		Number of LSTM hidden layers			
		1	2	3	4
32	MAE	3.240	3.829	3.262	3.504
	MSE	14.456	19.818	17.861	18.108
64	MAE	**2.400**	3.104	4.946	5.296
	MSE	**9.796**	15.046	46.799	39.411
128	MAE	3.700	4.923	4.898	4.229
	MSE	19.017	34.720	35.358	27.686
256	MAE	3.642	2.820	4.163	3.552
	MSE	20.644	14.511	24.801	20.591

TABLE 9. GRU forecasting errors for one week of CDRs predictions.

n_n		Number of GRU hidden layers			
		1	2	3	4
32	MAE	3.158	2.863	3.152	3.149
	MSE	15.048	12.748	15.350	14.927
64	MAE	3.170	3.583	4.034	**2.484**
	MSE	14.949	18.122	21.723	**10.254**
128	MAE	2.845	4.490	2.563	3.737
	MSE	12.123	32.631	11.034	23.611
256	MAE	3.403	2.797	3.663	3.286
	MSE	18.469	12.363	20.699	16.782

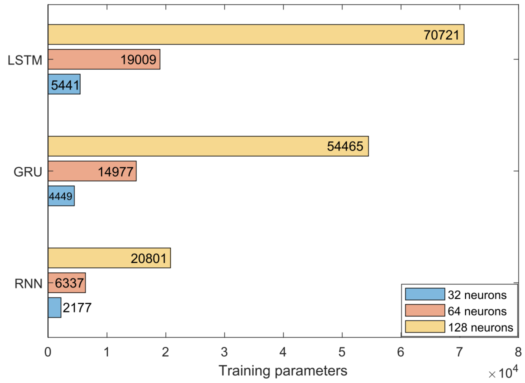


FIGURE 19. Training parameters for the three types of recurrent neural network considering different numbers of neurons, with 1 hidden layer.

from the RNNs neural networks. Besides, we consider the following network architecture:

- 3D CNN layer with 32 filters, Kernel size of $\{k, 2, 2\}$, with $k = \{1, 2, 3\}$ being the time-dimension and padding set to "same";
- 3D Pooling layer of dimension $\{2, 2, 2\}$ and padding set to "same";

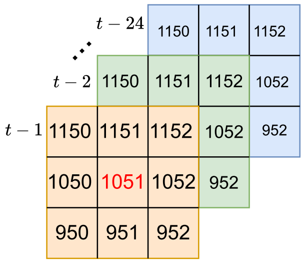


FIGURE 20. Spatio-temporal input tensor.

TABLE 10. Results obtained via 3D-CNN considering MaxPooling layers.

k		Number of 3D-CNN layers and Pooling layers	
		1	2
1	MAE	3.186	3.617
	MSE	15.020	18.893
	T.P.	49,641	10,761
2	MAE	**2.527**	3.337
	MSE	**10.421**	15.159
	T.P.	**49,769**	13,961
3	MAE	2.668	4.039
	MSE	10.959	24.857
	T.P.	49,897	18,185

TABLE 11. Results obtained via 3D-CNN considering Average Pooling layers.

k		Number of 3D-CNN layers and Pooling layers	
		1	2
1	MAE	2.895	**2.096**
	MSE	13.213	**6.718**
	T.P.	49,641	**10,761**
2	MAE	2.872	3.676
	MSE	12.984	20.807
	T.P.	49,769	13,961
3	MAE	3.159	3.458
	MSE	16.191	19.804
	T.P.	49,897	18,185

- for the simulations with 2 sequences of feature extractions (Figure 16), after the layers above, we have:
  - 3D CNN layer with 32 filters, Kernel size of $\{k, 2, 2\}$ and padding set to "valid";
  - 3D Pooling layer of dimension $\{2, 1, 1\}$ and padding set to "valid";
- flatten layer followed by dense layer with 32 neurons;
- output dense layer with 9 neurons;

Note that the output layer has 9 neurons because each one is the prediction for a position of the spatio-temporal input tensor in Figure 20 at time t. Then, the input tensor is updated with these new values and the last 24 hours are used for a new prediction. The results considering MAE, MSE, and the number of training parameters (T.P.) are presented in Tables 10 and 11.

D. COMPARISON BETWEEN TECHNIQUES

The time-series predictions for hourly aggregated CDRs considering the previous approaches are presented in Figure 21.

TABLE 12. Smallest MAE and MSE obtained with previous approaches.

Approach	MAE	MSE
SARIMA	2.911	12.827
Decomposed SARIMA	2.733	11.452
Simple RNN	2.705	10.887
3D-CNN (MaxPool)	2.527	10.421
GRU	2.484	10.254
LSTM	2.400	9.976
3D-CNN (AvgPool)	2.096	6.718

Comparing the results obtained via NN and SARIMA models, we notice that the neural network approaches provided at least one configuration with smaller MAE and MSE, see Table 12, where the best results of each model are shown. These differences in results may grow even more when the time-series to predict is highly nonlinear or the amount of data to be analyzed is very big. In such cases, SARIMA models may present degradation, leading to higher errors.

It is important to emphasize that the results obtained via neural networks approaches may vary a little, even if all parameters are kept the same, in case new training of the models is performed. Due to the highly non-convexity of such functions, different training phases may end up in different local minima. On the contrary, the SARIMA models have a closed-formula of calculation, which leads to the same results if their parameters $(p, d, q)(P, D, Q)_S$ do not change. Hence, the order presented in Table 12 may vary if new experiments are performed. Our goal, besides presenting NNs outperforming the statistical models, is to study different strategies, so they can be used in different situations or even together. In fact, many papers design prediction models based on a combination of these approaches. For instance, Huang et al. [3] presents a formulation where the model extracts spatial and time correlations based on 3d-CNN and LSTM, respectively, leading to very accurate results.

VII. OPEN RESEARCH DIRECTIONS

Our tutorial and survey let us overall highlight that, even though network traffic prediction has been extensively studied over the recent years, there are still gaps in the literature and several clear directions for improvement.

A. GAPS IN THE LITERATURE

As far as gaps are concerned, the search for models capable of forecasting the traffic demand with an accuracy as high as possible is intrinsically related to computational costs, due to model complexity and the time associated not just with its training, but also with its output predicted value. Yet, many papers do not present a formal analysis of model complexity, or results about the computational time of the solution. We argue that the community should make an effort of always including comparative analyses of new models with the state of the art not only in terms of performance but also complexity, so as to reveal the trade-off between these two key metrics.

Also, recent challenges in time series forecasting that are general purpose and not specific to network traffic have proven how many proposed models may fail to outperform simple baselines, as summarized by Makridakis et al. [120], [121]. In this regard, many studies in the literature do not provide direct comparison against naive or classical approaches, or do so with unclear parametrization of such benchmarks. Our conclusions is that the availability of a ready-to-use set of baseline models usable by the whole community would benefit the direct comparability of newly proposed solutions. We hope that the implementations developed for the tutorial and comparison parts of this document, which we provide as open-source software, can become the first core of such a toolbox.

It is also worth noting that, although the biggest part of papers cite the importance to predict the network traffic load (e.g., resource allocation and utilization, anticipatory decisions making, or network design), there is a gap on how these forecasting models could be used in a real world scenario to design optimization problems regarding energy saving, quality of service, close-loop control of the entire network, etc. We consider that there is a clear need that new solutions are not only compared to existing ones in terms of absolute or relative errors in the prediction (e.g., via MAE or MAPE metrics), but also deployed in practical applications; in this way, one can really appreciate the gain that a model would grant over another in production settings.

Another obstacle, related to benchmarking and even harder to surpass than those above, is that of the lack of a reference dataset of network traffic that is consistently adopted across performance evaluations of proposed models –opposed to what happens in other communities like the image processing one. Unfortunately, traffic data is often regarded as sensitive by operators, and only disclosed under restrictive Non-Disclosure Agreements (NDAs). Publicly available datasets are scarce and outdated. A promising option in this sense may be provided by emerging efforts on the generation of realistic synthetic spatial and temporal network traffic: early models proposed by Lin et al. [122] and Xu et al. [123] can produce vast amounts of traffic data that can be openly released, towards building a reference dataset to be commonly adopted by the community.

B. DIRECTIONS FOR IMPROVING FORECASTING MODELS

Our survey of forecasting models for network traffic evidences how, as in many other scientific and application domains, deep learning is revolutionizing the approaches for predicting future demands, thanks to significant accuracy improvements. Yet, it also outlines several emerging pathways for the improvement of existing deep learning architectures, which go beyond now rather standard RNNs, CNNs or GNNs and are aimed at further increasing the quality of predictions.

An interesting direction is that of hybrid strategies that mix deep learning and statistical modeling, jointly parametrized via a single training precess. As proven by early works, such as that of Lo Schiavo et al. [102], such combined approaches

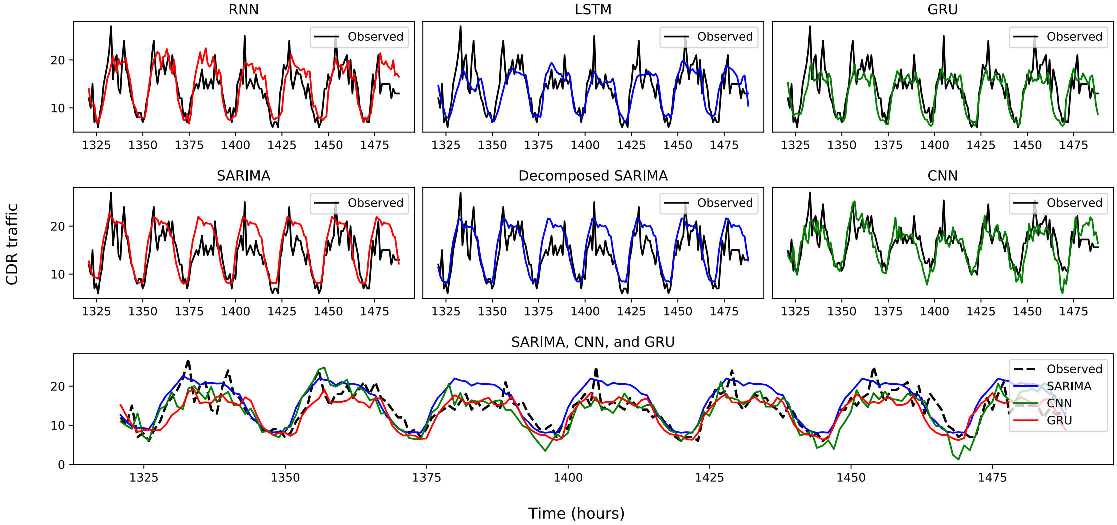


FIGURE 21. Time-series prediction of the three recurrent neural networks, CNN and SARIMA models.

shows promises of improving the performance of the forecast with respect to pure deep learning solutions. However, current studies in this direction are fairly preliminary and additional research is needed to unveil the potential of hybrid models.

In addition, the vast majority of network traffic predictors employ legacy loss functions, such as MAE or MSE, which are an ideal choice when the target of the forecast is anticipating the future traffic. Yet, in many practical applications, such as for network management, the value of the prediction is in how it can support effective decision-making: in these settings a predictor that minimizes the error to the future demand in not necessarily the optimal choice. On a related note, seminal studies, such as those of Bega et al. [16] have started exploring how to cope with the limitation above directly at model design stage. The concept is introducing loss functions that are designed by network experts and are tailored to the downstream application: such custom functions can train the predictor model to produce an output that is aligned with the requirements of the end user. Loss meta-learning frameworks such as that by Collet et al. [124] also start being proposed, which automate the definition of the most appropriate loss function to the target performance objective prediction.

Another relevant direction concerns the fact that forecast models are today evaluated in non-real-time settings. This implies that the current practice in performance assessment involves (*i*) off-line training with historical data; (*ii*) testing with limited re-played historical data typically recorded immediately after the training data; and, (*iii*) little or no attention to inference latency. Together, these habits lead to predictor implementations that are hardly validated in production-like settings. The situations calls for forecasting frameworks that can operate in on-line settings, e.g., by ensuring inference delays that are consistent with the requirements of the downstream applications, which can be down to milliseconds in radio access operations. It also requires in=depth studies of the prediction accuracy decrease over time, and need for re-training or, better, continual learning. Finally, evaluations should be designed so that it is ensured that models can correctly operate on live streaming data.

Finally, generalization is also a widely open research subject. Most models proposed in the literature are tested in specific scenarios, which, even when covering large urban areas, are still representative of a given network deployment. There are unanswered questions about how models trained in specific regions generalize to other areas. Recent studies of transfer learning approaches, such as those by Wu et al. [125] are promising in this sense, and pave the way for further investigations.

VIII. CONCLUSION

In this survey, we presented distinct approaches used to design network traffic prediction models. Different mathematical formulations were discussed and the most relevant were fully explained, so the reader is able to gain a deep understanding of each one. Based on the state of art, the network traffic prediction task was divided into three main areas (clustering, prediction model, and time-series decomposition) that were separately explained. A detailed list of papers addressing such a problem was also presented, allowing the reader to find some of the most relevant papers in this field of research. We also performed numerical experiments in a real world scenario data set to compare some of the explained mathematical models in terms of accuracy and computational cost. All the codes are publicly available, so other researchers can compare their approaches with the ones we provided or even use them as a foundation for more advanced solutions. Gaps in the literature were also listed, allowing the readers not just to use our codes and have a full understanding of network traffic prediction, but also giving possible directions for future works.

APPENDIX

In the numerical experiments, six different prediction models were designed and tested in a real world scenario data set. We provide the codes for all approaches, so other researches can use these models and perform a direct comparison in terms of computational cost and prediction accuracy. All the codes, written in Python 3, have comments to allow their easy use/understanding and are available in GitHub.[2] We also made available a small pre-processed subset of the data used to train the models. A description of the data set is provided by Barlacchi et al. [17] and its full version can be downloaded from the Harvard Dataverse Italia [126].

REFERENCES

[1] 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, 3rd Quart., 2017.

[2] P. Cao, F. Dai, G. Liu, J. Yang, and B. Huang, "A survey of traffic prediction based on deep neural network: Data, methods and challenges," in *Cloud Computing*. Cham, Switzerland: Springer, 2022, pp. 17–29.

[3] C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *Proc. IEEE 28th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Oct. 2017, pp. 1–6.

[4] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Exp. Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117921. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422011654

[5] M. Joshi and T. H. Hadi, "A review of network traffic analysis and prediction techniques," 2015, *arXiv:1507.05722*.

[6] P. Hendikawati, "A survey of time series forecasting from stochastic method to soft computing," *J. Phys., Conf.*, vol. 1613, no. 1, Aug. 2020, Art. no. 012019.

[7] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Exp. Syst. Appl.*, vol. 201, Sep. 2022, Art. no. 117163. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742200553X

[8] K. Lin and J. Zhang, "Energy efficiency enhancement via power modification with simulated annealing cell selection," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 526–531.

[9] P. Kanani, K. Shah, and M. Kaul, "A survey on evolution of mobile networks: 1G to 4G," *Int. J. Eng. Sci. Res.*, vol. 3, pp. 803–810, Feb. 2014.

[10] D. Abed, M. Ismail, and K. Jumari, "The evolution to 4G cellular systems: Architecture and key features of LTE-advanced networks," *Int. J. Comput. Netw. Wireless Commun.*, vol. 2, pp. 2250–3501, Jan. 2012.

[11] N. Seddigh, B. Nandy, R. Makkar, and J. F. Beaumont, "Security advances and challenges in 4G wireless networks," in *Proc. 8th Int. Conf. Privacy, Secur. Trust*, Aug. 2010, pp. 62–71.

[12] N. Al-Falahy and O. Y. Alani, "Technologies for 5G networks: Challenges and opportunities," *IT Prof.*, vol. 19, no. 1, pp. 12–20, Jan./Feb. 2017.

[13] J. Su, M. Beshley, K. Przystupa, O. Kochan, B. Rusyn, R. Stanislawski, O. Yaremko, M. Majka, H. Beshley, I. Demydov, J. Pyrih, and I. Kahalo, "5G multi-tier radio access network planning based on Voronoi diagram," *Measurement*, vol. 192, Mar. 2022, Art. no. 110814.

[14] 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, no. 1, pp. 44–51, Jan./Feb. 2016.

[15] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2014.

[16] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive network management in sliced 5G networks with deep learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 280–288.

[17] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of Milan and the province of trentino," *Sci. Data*, vol. 2, no. 1, pp. 1–15, Oct. 2015.

[18] P. Lechowicz, "Regression-based fragmentation metric and fragmentation-aware algorithm in spectrally-spatially flexible optical networks," *Comput. Commun.*, vol. 175, pp. 156–176, Jul. 2021.

[19] N. Stepanov, D. Alekseeva, A. Ometov, and E. S. Lohan, "Applying machine learning to LTE traffic prediction: Comparison of bagging, random forest, and SVM," in *Proc. 12th Int. Congr. Ultra Modern Telecommun. Control Syst. Workshops (ICUMT)*, Oct. 2020, pp. 119–123.

[20] H. Lu and F. Yang, "A network traffic prediction model based on wavelet transformation and LSTM network," in *Proc. IEEE 9th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Nov. 2018, pp. 1–4.

[21] Z. Tian, "Network traffic prediction method based on wavelet transform and multiple models fusion," *Int. J. Commun. Syst.*, vol. 33, no. 11, p. e4415, Jul. 2020.

[22] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, "Contemporary sequential network attacks prediction using hidden Markov model," in *Proc. 17th Int. Conf. Privacy, Secur. Trust (PST)*, Aug. 2019, pp. 1–3.

[23] A. Chen, J. Law, and M. Aibin, "A survey on traffic prediction techniques using artificial intelligence for communication networks," *Telecom*, vol. 2, no. 4, pp. 518–535, Dec. 2021.

[24] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering—A decade review," *Inf. Syst.*, vol. 53, pp. 16–38, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437915000733

[25] M. Chiş, S. Banerjee, and A. E. Hassanien, "Clustering time series data: An evolutionary approach," in *Foundations of Computational, Intelligence*. Berlin, Germany: Springer, 2009, pp. 193–207, doi: 10.1007/978-3-642-01091-0_9.

[26] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering validity checking methods: Part II," *ACM SIGMOD Rec.*, vol. 31, no. 3, pp. 19–27, 2002, doi: 10.1145/601858.601862.

[27] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd ed. New York, NY, USA: Springer, 2002.

[28] P. K. Hoong, "BitTorrent network traffic forecasting with ARMA," *Int. J. Comput. Netw. Commun.*, vol. 4, no. 4, pp. 143–156, Jul. 2012.

[29] I. K. T. Tan, P. K. Hoong, and C. Y. Keong, "Towards forecasting low network traffic for software patch downloads: An ARMA model forecast using CRONOS," in *Proc. 2nd Int. Conf. Comput. Netw. Technol.*, 2010, pp. 88–92.

[30] G. Calafiore and L. El Ghaoui, *Optimization Models* (Control systems and optimization series). Cambridge, U.K.: Cambridge Univ. Press, Oct. 2014.

[31] S. Jung, C. Kim, and Y. Chung, "A prediction method of network traffic using time series models," in *Proc. Int. Conf. Comput. Sci. Appl.*, vol. 3982, Nov. 2006, pp. 234–243.

[32] M. Alsharif, M. Younes, and J. Kim, "Time series ARIMA model for prediction of daily and monthly average global solar radiation: The case study of Seoul, South Korea," *Symmetry*, vol. 11, no. 2, p. 240, Feb. 2019.

[33] Y. Yu, J. Wang, M. Song, and J. Song, "Network traffic prediction and result analysis based on seasonal ARIMA and correlation coefficient," in *Proc. Int. Conf. Intell. Syst. Design Eng. Appl.*, Oct. 2010, pp. 980–983.

[34] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big data driven mobile traffic understanding and forecasting: A time series approach," *IEEE Trans. Services Comput.*, vol. 9, no. 5, pp. 796–805, Sep./Oct. 2016.

[35] M. Zhang, H. Fu, Y. Li, and S. Chen, "Understanding urban dynamics from massive mobile traffic data," *IEEE Trans. Big Data*, vol. 5, no. 2, pp. 266–278, Jun. 2019.

[36] B. Vujicic, H. Chen, and L. Trajkovic, "Prediction of traffic in a public safety network," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 1–4.

[37] H. Chen and L. Trajkovic, "Trunked radio systems: Traffic prediction based on user clusters," in *Proc. 1st Int. Symp. Wireless Commun. Syst.*, 2004, pp. 76–80.

[38] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucl. Acids Res.*, vol. 16, no. 22, pp. 10881–10890, 1988. [Online]. Available: https://europepmc.org/articles/PMC338945

[39] B. Cici, M. Gjoka, A. Markopoulou, and C. T. Butts, "On the decomposition of cell phone activity patterns and their connection with urban ecology," in *Proc. 16th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.* New York, NY, USA: Association for Computing Machinery, 2015, pp. 317–326.

[2] https://github.com/Ferreira-O-Gabriel/Network-Traffic-Prediction.git

[40] D. Miao, X. Qin, and W. Wang, "The periodic data traffic modeling based on multiplicative seasonal ARIMA model," in *Proc. 6th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2014, pp. 1–5.

[41] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[42] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 262–270, doi: 10.1145/2339530.2339576.

[43] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Proc. IEEE 18th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2006, pp. 1135–1138.

[44] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining Knowl. Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[45] J. Paparrizos and L. Gravano, "K-shape: Efficient and accurate clustering of time series," *ACM SIGMOD Rec.*, vol. 45, no. 1, pp. 69–76, Jun. 2016, doi: 10.1145/2949741.2949758.

[46] T. W. Liao, "Clustering of time series data—A survey," *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320305001305

[47] C. A. H. Suarez, O. J. S. Parra, and A. E. Díaz, "An ARIMA model for forecasting Wi-Fi data network traffic values," *Ingeniería e Investigación*, vol. 29, no. 2, pp. 65–69, May 2009.

[48] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal ARIMA model with limited input data," *Eur. Transp. Res. Rev.*, vol. 7, no. 3, pp. 1–9, Jun. 2015.

[49] Y. Shu, M. Yu, J. Liu, and O. W. W. Yang, "Wireless traffic modeling and prediction using seasonal ARIMA models," in *Proc. IEEE Int. Conf. Commun.*, vol. 3, May 2003, pp. 1675–1679.

[50] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Dec. 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[51] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, "Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Commun. Process.*, Aug. 2011, pp. 95–102.

[52] A. Azzouni and G. Pujolle, "A long short-term memory recurrent neural network framework for network traffic matrix prediction," May 2017, *arXiv:1705.05690*.

[53] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, vol. 9, Y. W. Teh and M. Titterington, Eds. Sardinia, Italy: Chia Laguna, May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[54] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 187–193.

[55] 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 Conf. Comput. Commun.*, May 2017, pp. 1–9.

[56] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with KNN and LSTM," *J. Adv. Transp.*, vol. 2019, pp. 1–10, Feb. 2019.

[57] S. Hochreiter and J. J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 80–1735, 1997.

[58] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179

[59] S. A. Patil, L. A. Raj, and B. K. Singh, "Prediction of IoT traffic using the gated recurrent unit neural network-(GRU-NN-) based predictive model," *Secur. Commun. Netw.*, vol. 2021, pp. 1–7, Oct. 2021.

[60] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328.

[61] C. Chen, K. Li, S. G. Teo, X. Zou, K. Li, and Z. Zeng, "Citywide traffic flow prediction based on multiple gated spatio-temporal convolutional neural networks," *ACM Trans. Knowl. Discovery Data*, vol. 14, no. 4, pp. 1–23, Aug. 2020.

[62] Y. J. Ong, M. Qiao, and D. Jadav, "Temporal tensor transformation network for multivariate time series prediction," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2020, pp. 1594–1603.

[63] T. Deng, M. Wan, K. Shi, L. Zhu, X. Wang, and X. Jiang, "Short term prediction of wireless traffic based on tensor decomposition and recurrent neural network," *Social Netw. Appl. Sci.*, vol. 3, no. 9, pp. 1–14, Sep. 2021.

[64] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2016.

[65] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Apr. 2021.

[66] Z. Wang, J. Hu, G. Min, Z. Zhao, Z. Chang, and Z. Wang, "Spatial-temporal cellular traffic prediction for 5G and beyond: A graph neural networks-based approach," *IEEE Trans. Ind. Informat.*, early access, Jun. 20, 2022, doi: 10.1109/TII.2022.3182768.

[67] X. Zhou, Y. Zhang, Z. Li, X. Wang, J. Zhao, and Z. Zhang, "Large-scale cellular traffic prediction based on graph convolutional networks with transfer learning," *Neural Comput. Appl.*, vol. 34, no. 7, pp. 5549–5559, Apr. 2022, doi: 10.1007/s00521-021-06708-x.

[68] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of internet backbone traffic," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1110–1124, Sep. 2005.

[69] G. Mao, "Real-time network traffic prediction based on a multiscale decomposition," in *Proc. 4th Int. Conf. Netw.*, vol. 3420. Reunion Island, France: Springer, 2005, pp. 492–499, doi: 10.1007/978-3-540-31956-6_58.

[70] B. Zhou, D. He, and Z. Sun, *Traffic Modeling and Prediction Using ARIMA/GARCH Model*. Boston, MA, USA: Springer, Sep. 2006, pp. 101–121.

[71] H. Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *Proc. Int. Symp. Inf. Technol.*, vol. 4, Sep. 2008, pp. 1–6.

[72] C. Chen, Q. Pei, and L. Ning, "Forecasting 802.11 traffic using seasonal ARIMA model," in *Proc. Int. Forum Comput. Sci.-Technol. Appl.*, 2009, pp. 347–350.

[73] J. Guo, Y. Peng, X. Peng, Q. Chen, J. Yu, and Y. Dai, "Traffic forecasting for mobile networks with multiplicative seasonal ARIMA models," in *Proc. 9th Int. Conf. Electron. Meas. Instrum.*, Aug. 2009, pp. 3–377.

[74] C. Christodoulos, C. Michalakelis, and D. Varoutas, "Forecasting with limited data: Combining ARIMA and diffusion models," *Technol. Forecasting Social Change*, vol. 77, no. 4, pp. 558–565, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0040162510000235

[75] N. K. Hoong, P. K. Hoong, I. K. T. Tan, N. Muthuvelu, and L. C. Seng, "Impact of utilizing forecasted network traffic for data transfers," in *Proc. 13th Int. Conf. Adv. Commun. Technol.*, 2011, pp. 1199–1204.

[76] H. Haviluddin and R. Alfred, "Forecasting network activities using ARIMA method," *J. Adv. Comput. Netw.*, vol. 2, no. 3, pp. 173–177, 2014.

[77] W. Yoo and A. Sim, "Network bandwidth utilization forecast model on high bandwidth networks," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 494–498.

[78] S. Medhn, B. Seifu, A. Salem, and D. Hailemariam, "Mobile data traffic forecasting in UMTS networks based on SARIMA model: The case of Addis Ababa, Ethiopia," in *Proc. IEEE AFRICON*, Sep. 2017, pp. 285–290.

[79] S. Zhang, S. Zhao, M. Yuan, J. Zeng, J. Yao, M. R. Lyu, and I. King, "Traffic prediction based power saving in cellular networks: A machine learning method," in *Proc. 25th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.* New York, NY, USA: Association for Computing Machinery, 2017, doi: 10.1145/3139958.3140053.

[80] D. Madariaga, M. Panza, and J. Bustos-Jimenez, "I'm only unhappy when it rains: Forecasting mobile QoS with weather conditions," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–6.

[81] J. A. Bastos, "Forecasting the capacity of mobile networks," *Telecommun. Syst.*, vol. 72, no. 2, pp. 231–242, Oct. 2019.

[82] A. S. Arifin and M. I. Habibie, "The prediction of mobile data traffic based on the ARIMA model and disruptive formula in industry 4.0: A case study in Jakarta, Indonesia," *Telkomnika*, vol. 18, no. 2, pp. 907–918, 2020.

[83] H. Yang, X. Li, W. Qiang, Y. Zhao, W. Zhang, and C. Tang, "A network traffic forecasting method based on SA optimized ARIMA–BP neural network," *Comput. Netw.*, vol. 193, Jul. 2021, Art. no. 108102. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001821

[84] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Computer network traffic prediction: A comparison between traditional and deep learning neural networks," *Int. J. Big Data Intell.*, vol. 3, no. 1, p. 28, 2016.

[85] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 2353–2358.

[86] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 4, pp. 554–557, Aug. 2018.

[87] Y. Hua, Z. Zhao, Z. Liu, X. Chen, R. Li, and H. Zhang, "Traffic prediction based on random connectivity in deep learning with long short-term memory," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–6.

[88] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1827–1832.

[89] 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, Nov. 2018.

[90] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 246–251.

[91] W. Shihao, Z. Qinzheng, Y. Han, L. Qianmu, and Q. Yong, "A network traffic prediction method based on LSTM," *ZTE Commun.*, vol. 17, no. 2, p. 19, 2019. [Online]. Available: http://zte.magtechjournal.com/EN/abstract/article12.shtml

[92] S. P. Sone, J. J. Lehtomaki, and Z. Khan, "Wireless traffic usage forecasting using real enterprise network data: Analysis and methods," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 777–797, 2020.

[93] M. Li, Y. Wang, Z. Wang, and H. Zheng, "A deep learning method based on an attention mechanism for wireless network traffic prediction," *Ad Hoc Netw.*, vol. 107, Oct. 2020, Art. no. 102258. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870519310923

[94] H. Elsherbiny, H. M. Abbas, H. Abou-zeid, H. S. Hassanein, and A. Noureldin, "4G LTE network throughput modelling and prediction," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[95] F. W. Alsaade and M. H. Al-Adhaileh, "Cellular traffic prediction based on an intelligent model," *Mobile Inf. Syst.*, vol. 2021, pp. 1–15, Jul. 2021.

[96] W.-C. Chien and Y.-M. Huang, "A lightweight model with spatial–temporal correlation for cellular traffic prediction in Internet of Things," *J. Supercomput.*, vol. 77, no. 9, pp. 10023–10039, Sep. 2021, doi: 10.1007/s11227-021-03662-2.

[97] Y. Gao, M. Zhang, J. Chen, J. Han, D. Li, and R. Qiu, "Accurate load prediction algorithms assisted with machine learning for network traffic," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2021, pp. 1683–1688.

[98] Q. Wu, X. Chen, Z. Zhou, L. Chen, and J. Zhang, "Deep reinforcement learning with spatio-temporal traffic forecasting for data-driven base station sleep control," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 935–948, Apr. 2021, doi: 10.1109/TNET.2021.3053771.

[99] S. Zhan, L. Yu, Z. Wang, Y. Du, Y. Yu, Q. Cao, S. Dang, and Z. Khan, "Cell traffic prediction based on convolutional neural network for software-defined ultra-dense visible light communication networks," *Secur. Commun. Netw.*, vol. 2021, pp. 1–10, Aug. 2021.

[100] W. Shen, H. Zhang, S. Guo, and C. Zhang, "Time-wise attention aided convolutional neural network for data-driven cellular traffic prediction," *IEEE Wireless Commun. Lett.*, vol. 10, no. 8, pp. 1747–1751, Aug. 2021.

[101] T. Kuber, I. Seskar, and N. Mandayam, "Traffic prediction by augmenting cellular data with non-cellular attributes," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2021, pp. 1–6.

[102] L. L. Schiavo, M. Fiore, M. Gramaglia, A. Banchs, and X. Costa-Perez, "Forecasting for network management with joint statistical modelling and machine learning," in *Proc. IEEE 23rd Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2022, pp. 60–69.

[103] M. Hasegawa, G. Wu, and M. Mizuni, "Applications of nonlinear prediction methods to the internet traffic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2001, pp. 169–172.

[104] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2006, pp. 2635–2642.

[105] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using holt-winter's exponential smoothing," in *Proc. 15th Int. Conf. Softw., Telecommun. Comput. Netw.*, 2007, pp. 1–5.

[106] R. Filho and J. Maia, "Network traffic prediction using PCA and K-means," in *Proc. IEEE Network Oper. Manage. Symp. (NOMS)*, Jan. 2010, pp. 938–941.

[107] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.* New York, NY, USA: Association for Computing Machinery, Jun. 2011, pp. 305–316, doi: 10.1145/1993744.1993776.

[108] R. Li, Z. Zhao, X. Zhou, and H. Zhang, "Energy savings scheme in radio access networks via compressive sensing-based traffic load prediction," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 4, pp. 468–478, Apr. 2014.

[109] A. Y. Nikravesh, S. A. Ajila, C.-H. Lung, and W. Ding, "Mobile network traffic prediction using MLP, MLPWD, and SVM," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2016, pp. 402–409.

[110] 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. Comput.*, Jun. 2018, pp. 231–240.

[111] X. Wang, Z. Zhou, Z. Yang, Y. Liu, and C. Peng, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," in *Proc. IEEE 25th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2017, pp. 1–10.

[112] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3899–3912, Jun. 2017.

[113] S. Narejo and E. Pasero, "An application of internet traffic prediction with deep neural network," in *Multidisciplinary Approaches to Neural Computing* (Smart Innovation, Systems and Technologies). Cham, Switzerland: Springer, Aug. 2018, pp. 139–149.

[114] C. Vinchoff, N. Chung, T. Gordon, L. Lyford, and M. Aibin, "Traffic prediction in optical networks using graph convolutional generative adversarial networks," in *Proc. 22nd Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 2020, pp. 1–4.

[115] Y. Yang, "A new network traffic prediction approach in software defined networks," *Mobile Netw. Appl.*, vol. 26, no. 2, pp. 681–690, Apr. 2021.

[116] S. C. Sun and W. Guo, "Forecasting wireless demand with extreme values using feature embedding in Gaussian processes," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–6.

[117] L. Ale, N. Zhang, S. A. King, and J. Guardiola, "Spatio-temporal Bayesian learning for mobile edge computing resource planning in smart cities," *ACM Trans. Internet Technol.*, vol. 21, no. 3, pp. 1–21, Jun. 2021, doi: 10.1145/3448613.

[118] I. Tomic, E. Bleakley, and P. Ivanis, "Predictive capacity planning for mobile networks—ML supported prediction of network performance and user experience evolution," *Electronics*, vol. 11, no. 4, p. 626, Feb. 2022.

[119] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[120] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: 100,000 time series and 61 forecasting methods," *Int. J. Forecasting*, vol. 36, no. 1, pp. 54–74, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019301128

[121] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *Int. J. Forecasting*, vol. 38, no. 4, pp. 1346–1364, Oct. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207021001874

[122] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using GANs for sharing networked time series data: Challenges, initial promise, and open questions," in *Proc. ACM Internet Meas. Conf.* New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 464–483, doi: 10.1145/3419394.3423643.

[123] K. Xu, R. Singh, M. Fiore, M. K. Marina, H. Bilen, M. Usama, H. Benn, and C. Ziemlicki, "SpectraGAN: Spectrum based generation of city scale spatiotemporal mobile network traffic data," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.* New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 243–258, doi: 10.1145/3485983.3494844.

[124] A. Collet, A. Banchs, and M. Fiore, "LossLeaP: Learning to predict for intent-based networking," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2022, pp. 2138–2147.

[125] Q. Wu, K. He, X. Chen, S. Yu, and J. Zhang, "Deep transfer learning across cities for mobile traffic prediction," *IEEE/ACM Trans. Netw.*, vol. 30, no. 3, pp. 1255–1267, Jun. 2022.

[126] T. Italia, "Telecommunications-SMS, call, internet-MI," Telecom Italia, Italy, Tech. Rep. 1, 2015, doi: 10.7910/DVN/EGZHFV.

GABRIEL O. FERREIRA (Member, IEEE) received the B.S. degree in mechatronics engineering from the Federal Center for Technological Education of Minas Gerais, Brazil, in 2018, and the M.Sc. degree in electrical engineering from the Federal University of São João del-Rei, Brazil, in 2020. He is currently pursuing the Ph.D. degree with the Department of Electronics and Telecommunication, Politecnico di Torino, Italy.

His research interests include modeling and forecasting mobile network traffic, network optimization, and network closed-loop control.

CHIARA RAVAZZI (Member, IEEE) received the Ph.D. degree in mathematics for engineering sciences from the Politecnico di Torino, in 2011. She was a Visiting Member at the Massachusetts Institute of Technology (LIDS), in 2010, and a Postdoctoral Researcher at the DISMA, DET, Politecnico di Torino, from 2011 to 2016. Since 2017, she has been a Tenured Researcher of the National Research Council of Italy (CNR-IEIIT). Her current research interests include control and information theory, signal processing, optimization, and learning algorithms for network systems. She has been serving as an Associate Editor for the IEEE Transactions on Signal Processing, since 2019, and the IEEE Transaction on Control Systems Letters, since 2021.

FABRIZIO DABBENE (Senior Member, IEEE) received the Laurea and Ph.D. degrees from the Politecnico di Torino, Italy, in 1995 and 1999, respectively.

He is currently the Director of Research with the Institute IEIIT, National Research Council of Italy (CNR), Milan, Italy, where he is the Coordinator of the Information and Systems Engineering Group. He has held visiting and research positions with The University of Iowa, Iowa City, IA, USA; Penn State University, University Park, PA, USA; and the Russian Academy of Sciences, Institute of Control Science, Moscow, Russia. He has authored or coauthored more than 100 research papers and two books.

Dr. Dabbene was an Elected Member of the Board of Governors, from 2014 to 2016. He has served as the vice president for publications, from 2015 to 2016. He is currently chairing the IEEE-CSS Italy Chapter. He has also served as an Associate Editor for *Automatica*, from 2008 to 2014, and IEEE Transactions on Automatic Control, from 2008 to 2012. He is also a Senior Editor of the IEEE Control Systems Society Letters.

GIUSEPPE C. CALAFIORE (Fellow, IEEE) was a Senior Fellow at the Institute of Pure and Applied Mathematics (IPAM), University of California at Los Angeles, in 2010. He is currently a Full Professor with DET, Politecnico di Torino, where he coordinates the Control Systems and Data Science Group. He is also a Full Professor with the College of Mechanical Engineering, VinUniversity, Hanoi, Vietnam, and an Associate Fellow of the IEIIT-CNR, Italy. He held visiting positions at the Information Systems Laboratory (ISL), Stanford University, CA, in 1995; the Ecole Nationale Supérieure de Techniques Avancées (ENSTA), Paris, in 1998; and the University of California at Berkeley, in 1999, 2003, 2007, 2017, 2018, and 2019, and 2021, where he lately taught a master course on financial data science. He has over 20 years of teaching experience in master-level and Ph.D. courses in the areas of systems and control theory, convex optimization and machine learning. He is the author of about 210 journal and conference proceedings papers and eight books. His research interests include convex optimization, identification and control of uncertain systems, with applications ranging from finance and economic systems to robust control, machine learning, and data science. He received the IEEE Control System Society "George S. Axelby" Outstanding Paper Award, in 2008.

MARCO FIORE (Senior Member, IEEE) received the M.Sc. degree from the University of Illinois at Chicago, IL, USA, in 2003, the M.Sc. and Ph.D. degrees from the Politecnico di Torino, Italy, in 2004 and 2008, respectively, and the Habilitation a Diriger des Recherches (HDR) degree from the Universite de Lyon, France, in 2014.

He was a Visiting Researcher at Rice University, Houston, TX, USA, from 2006 to 2007; the Universitat Politecnica de Catalunya (UPC), Spain, in 2008; and University College London (UCL), U.K., from 2016 to 2018. He held tenured positions as a Maitre de Conferences (Associate Professor) at the Institut National des Sciences Appliquees (INSA) de Lyon, from 2009 to 2013; and a Researcher at Consiglio Nazionale delle Ricerche (CNR), Italy, from 2013 to 2019. He is currently a Research Associate Professor with the IMDEA Networks Institute and the CTO with Net AI Tech Ltd.

Dr. Fiore is a member of ACM. He was a recipient of the European Union Marie Curie Fellowship and the Royal Society International Exchange Fellowship. He leads the Networks Data Science Group, IMDEA Networks Institute, which focuses on research at the interface of computer networks, data analysis, and machine learning.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE' within the CRUI CARE Agreement