Orchestrating Edge- and Cloud-based Predictive Analytics Services

Venkatarami Reddy Chintapalli[#], Koteswararao Kondepu[•], Andrea Sgambelluri[•], Antony Franklin A[#],
Bheemarjuna Reddy Tamma[#], Piero Castoldi[•], Luca Valcarenghi[•]
[#]IIT Hyderabad, India; [•]Scuola Superiore Sant'Anna, Pisa, Italy
e-mail:cs17resch01007@iith.ac.in, k.kondepu@sssup.it

***Abstract*—In the Zero-touch network and service management (ZSM) architecture, devised by ETSI, making predictions on the observed data is among the functions provided by the analytics block of the control loop cycle. Prediction performance depends on several parameters, such as the utilized computational resources, the leveraged prediction techniques, the deployment location of the prediction tools with respect to the data.**

This paper proposes a Hybrid Forecast Framework (HFF) running at the network edge or in the cloud to provide forecasting with the performance required by the control loop cycle. Forecasting at the edge might shorten the control loop cycle if resources shall be made available locally where data is collected. However, in general, edge computational resources are less abundant than the cloud ones, thus causing longer time to perform the prediction. On the opposite, forecasting in the cloud might require more time for the data to reach the utilized tools but more computational resources could be exploited. The HFF is based on utilizing traditional time series analysis prediction algorithms to minimize the utilized resources and energy at the edge while it exploits AI/ML tools to make predictions in the cloud.

Results show that for short lead time (i.e., the time, in the future, at which the status of the considered parameter is predicted) edge-based prediction exploiting time series analysis provides better accuracy, requires less resources and time (thus energy) than cloud-based prediction. However, if the lead time is long, cloud-based prediction exploiting Artificial Intelligence/Machine Learning (AI/ML) provides better accuracy. Thus, if the lead time is long, it is preferable because the long lead time compensates for the higher time for prediction, mainly due to data transfer.

***Index Terms*—Forecasting, AI/ML, Time series analysis, Edge, Cloud.**

I. Introduction

The foreseen complexity in operating and managing 5G and beyond networks has fostered the approach to closed-loop automation of network and service management operations. The Zero-touch Network and Service Management (ZSM), proposed by ETSI [1], is among the emerging architectures [2] (e.g., TM Forum's Zero-touch Orchestration, Operations and Management (ZOOM) and Smart BPM (Business Process Management)). The ZSM broad design goal is to enable zero-touch automated network and service management in a multi-vendor environment. The ZSM relies on closed-loop management operation. To achieve a closed-loop operation, a management framework needs to provide means for the ordered invocation of the phases of the closed-loop (e.g., the Observe, Orient, Decide, and Act steps of the OODA loop proposed by J.R. Boyd [3]). The management functions contribute with their respective management service capabilities to achieve the functionality of different steps of the closed loop. Specifically, Data Collection contributes to Observe, Analytics contributes to Orient, Intelligence contributes to Decide, and Orchestration and Control contribute to the Act steps.

In general, the control loop duration impacts the Service Level Agreement (SLA) between service providers and customers because it impacts the reaction time of the system to state changes. The control loop duration, in turn, is impacted not only by the time required to perform the control loop functions but also by the time required to transfer data between the control loop functional elements. Thus, making fast and accurate predictions might improve the provider capability of fulfilling the SLAs but attention also has be paid to the communication between the functional elements.

ETSI provides only the architectural view of the ZSM, but it does not provide possible implementations and the related performance evaluations. This paper focuses on the predictive analytics function of the ZSM analytics block. Forecasting/predictive analytics has been already proposed for scaling 5G core network resources by anticipating traffic load changes [4]. The approach proposed in [4] is based on two Artificial Intelligence/Machine Learning (AI/ML) techniques: Recursive Neural Networks (RNN), more specifically Long Short-Term Memory (LSTM) and Deep Neural Network (DNN). The results show that forecast-based scalability mechanism outperforms the threshold-based solutions in terms of latency to react to traffic changes. However, in [4] all the control loop functions are co-located, thus propagation delay of data between control loop functional elements is not taken into account. In more distributed scenarios where Virtual Network Function (VNF) and Software Defined Network (SDN) technologies are exploited, the placement of predictive analytics services, whether at the edge or in the cloud, might impact the control loop cycle time.

In addition, exploiting edge resources offers delay-sensitive and cost-efficient services, because the edge is closer to the end users. Conversely, the cloud is far from the end users, thus an additional communication delay [5] is to be considered. Although the edge brings several advantages over the cloud, the computation and storage resources of edge servers are limited and less powerful when compared to the cloud servers.

For what concerns the prediction/forecast techniques and, more specifically, for time series data, either traditional time series analysis or ML-based methods can be exploited. Traditional time series analysis methods such as Error, Trend, Seasonality forecast (ETS), Auto Regressive Integrated Moving Average (ARIMA), and Exponential Smoothing are the most popular and effective time series predictors [6]. Traditional methods might also outperform several other ML-based methods, including LSTM and RNNs, depending on the considered dataset and the forecast lead time (i.e., the time, in the future, at which the status of the considered quantity is predicted) [7]. Traditional prediction techniques are fast to train and forecast (i.e., testing), but are neither very accurate nor flexible to adapt to complex data.

On the other end, ML-based methods, such as LSTM, can forecast accurately but they require long training. In addition, ML-based methods require a large amount of data, which is a computationally expensive. Thus, it is not always feasible to use such powerful computing tools.

To reduce such complexity and also concerns related to data privacy while transmitting over the network, edge analytics can be exploited depending on the prediction accuracy and the availability of the data. Note that, severe resource scarcity issues may exist if ML-based methods are supported at the edge, especially when a large amount of data has to be processed. Thus, applying traditional prediction methods at the edge, while applying ML-based methods in the cloud, can provide a trade off between the achievable performance and the available resources.

This paper proposes a Hybrid Forecasting Framework (HFF) running at the network edge and in the cloud. HFF considers two different traditional time series analysis prediction approaches such as Double Exponential Smoothing (DES) and Triple Exponential Smoothing (TES) running at the edge and one ML-based approach such as Long Short-Term Memory (LSTM) running in the cloud. These methods are utilized to forecast the number of VNFs/Virtual Machines (VMs) necessary to support an automotive application as a function of the road traffic while maintaining an agreed SLA, such as the elaboration time in Advanced Driving Assistance (ADA) services. Although the considered time series is the number of cars passing through a specific road, the considered framework is general and can be applied to any time-varying request (e.g., VNFs for packet inspection, lightpath dynamic demands).

Results show that traditional time series analysis methods based on exponential smoothing outperform ML methods, such as LSTM, in terms of accuracy (measured as root mean square error) for short lead time forecasts (i.e., the time, in the future, at which the status of the considered quantity is predicted).

Moreover they require less resources and time, and thus energy. However, if the lead time is long, cloud-based prediction exploiting AI/ML provides better accuracy. Thus, if the lead time is long, cloud-based prediction is preferable because the long lead time compensates for the longer time for prediction, mainly due to data transfer. In the considered scenario, as the time for activating/scaling Virtual Machines (VMs) is long (usually minutes), thus requiring a long lead time, cloud analytics is preferable.

II. Hybrid Forecast Framework Architecture and Implementation

Figure 1 reports the control loop described in [1]. The overall control loop time, depends not only on the time taken to perform the control loop functions (i.e., observe, orient, decide, and act) by the respective management functions (i.e., data collection, analytics, intelligence, orchestration and control), but also on the time taken by the communication among the functional elements.

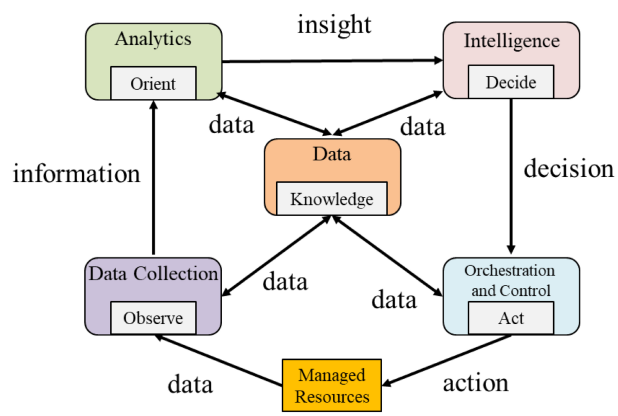

Figure 1. Mapping between ZSM architecture building blocks and closed loop functions as in [1]

This paper focuses only on the part of the control loop between the data collection, from the managed resources, and the analytics block, as depicted in Figure 2, even though a comparable contribution to the overall control cycle time can be expected to be provided by the remaining part of the control loop.

Figure 2 shows the functions of the control loop applied to the considered scenario. Data is collected at the edge. With respect to traditional application, the proposed Hybrid Forecast Framework (HFF) features the deployment of the data analytics function both at the edge and in the core. Edge data analytics is based on traditional time series analysis prediction approaches such as DES and TES. Core data analytics is based on ML-based approach such as LSTM. The selection among the two possible analytics functions depends on the required control cycle time, the available resources and the lead time, defined as period for which forecasts are needed (i.e., the future time for which the data need to be predicted).

III. Considered Forecasting Methods

Exponential smoothing and ML-based methods are considered for implementing edge analytics and cloud analytics, respectively. Exponential smoothing is a time series forecasting method for uni-variate data where the prediction is a weighted linear sum of recent past observations or lags [8]. In this paper, two exponential smoothing techniques are considered: Double Exponential Smoothing (DES) and Triple Exponential Smoothing (TES).

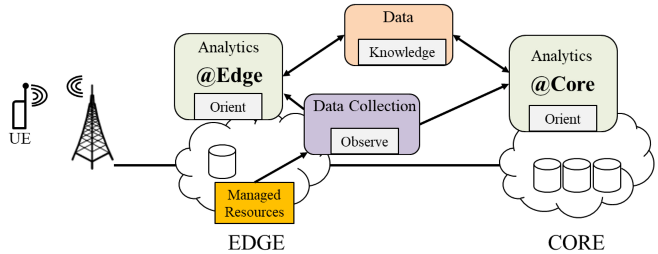

Figure 2. Scope of the paper within the ZSM and edge and core analytics

A. Double Exponential Smoothing (DES)

DES uses a level smoothing L_t with a level factor $\alpha \in [0,1]$ and trend smoothing T_t with a trend factor $\beta \in [0,1]$, as described in Eqs. (2) and (3), to compute the k-step ahead (namely *lead time*) forecast y_{t+k} through Eq. (1).

$$\widehat{y}_{t+k} = L_t + k \cdot T_t \tag{1}$$
$$L_t = \alpha \cdot y_t + (1-\alpha) \cdot (L_{t-1} + T_{t-1}) \tag{2}$$
$$T_t = \beta \cdot (L_t - L_{t-1}) + (1-\beta) \cdot T_{t-1} \tag{3}$$

The level smoothing L_t is obtained based on the previous experienced time interval value of level smoothing L_{t-1} and trend smoothing T_{t-1}. Note that, in Eq. (2), the current value of time series (i.e., y_t) is used to estimate L_t. Similarly, the trend smoothing T_t is obtained from previous values of the level smoothing L_{t-1} and trend smoothing T_{t-1}. However, instead of the current value of the time series the current values of the level smoothing L_t is utilized. The main drawback of DES is the inability to account for seasonality of demands when the data show both trend and seasonality.

B. Triple Exponential Smoothing (TES)

As shown in Eqs. (5)-(7), TES exploits three different forecasting factors such as *level* L_t, *trend* T_t, and *seasonality* S_t. Eq. (4) forecasts the value of the observed quantity $\widehat{y}_{t+k}$ at time $t+k$, given all the data points up to time t and the seasonality constant s (i.e., the number of observations per season). TES can be performed in two ways, namely *additive* and *multiplicative* methods, depending on the seasonality effect. The *additive* method is considered when the seasonality effect is constant, whereas, the *multiplicative* method is used when the size of seasonality effect is proportional to the mean [9]. Note that, the following equations are defined based on the *additive* method.

$$\widehat{y}_{t+k} = L_t + k \cdot T_t + S_{t+k-s} \tag{4}$$
$$L_t = \alpha \cdot (y_t - S_{t-s}) + (1-\alpha) \cdot (L_{t-1} + T_{t-1}) \tag{5}$$
$$T_t = \beta \cdot (L_t - L_{t-1}) + (1-\beta) \cdot T_{t-1} \tag{6}$$
$$S_t = \gamma \cdot (y_t - L_t) + (1-\gamma) \cdot S_{t-s}, \tag{7}$$

where s is the length of the seasonal cycle, $\alpha \in [0,1]$, $\beta \in [0,1]$, and $\gamma \in [0,1]$.

C. Long Short-Term Memory (LSTM)

LSTM is a special form of Recurrent Neural Network (RNN) that can learn long-term dependencies based on the information remembered in previous steps of the learning process. LSTM consists of a set of recurrent blocks (i.e., memory blocks) where each block contains one or more memory cells and multiplicative units such as *input*, *output* and *forget gate*.

LSTM is one of the most successful model for forecasting long-term time series. The LSTM can be characterized by different hyper-parameters, specifically the number of hidden layers, the number of neurons, and the batch size. Details of LSTM parameters and their impact on prediction accuracy can be found in [10]. However, the process of finding optimal hyper-parameters which minimize the forecasting error could be time and resource consuming.

In the proposed approach, the LSTM input vector corresponds to the n previous data points and the output vector corresponds to k-steps ahead with respect to the current time t of the considered time series. In this work, a *stacked LSTM model* is exploited with a single-step (i.e., $k = 1$) and a multi-step (i.e., $k > 1$) forecasting.

In *LSTM single-step forecasting (LSTM-SSF)*, a single data point is predicted based on n previous data points considered for forecasting (i.e., the size of the monitoring window):

$$P(t) = model(O(t-1), O(t-2), ..., O(t-n)), \tag{8}$$

where P is the prediction of the single data point at time t and O is the observed value in the n previous data points.

In *LSTM multi-step forecasting (LSTM-MSF)*, LSTM predicts k number of data points by considering n previous observed data points.

$$P(t+k-1, t+k-2, ..., t) = model(O(t-1), O(t-2), \dots, O(t-n)), \tag{9}$$

where k > 1.

The LSTM-MSF is exploited in two ways: one approach is forecasting k data points at a time from n data points as described in Eq. (9); the second approach is realizing a multi-step forecast by using a recursive single-step forecast, where the forecast data value is used as an input to the model by replacing $t-n$ data point as defined in Eq. (10). The latter case is referred to as *LSTM-MSF-recursive*.

$$\begin{aligned} P(t) &= model(O(t-1), O(t-2), ..., O(t-n)) \\ P(t+1) &= model(P(t), O(t-1), ..., O(t-n+1)) \\ &\dots \\ P(t+k-1) &= model(P(t+k-2), P(t+k-3), \\ &\dots, O(t-n+k-1)) \end{aligned} \tag{10}$$

Note that, *DES-recursive* and *TES-recursive* are also considered in this paper and implemented in the same way, by updating the level smoothing L_t and the trend smoothing T_t with the predicted data points while calculating $\widehat{y}_{t+k}$ in Eqs. (1) and (4).

IV. Performance Evaluation

The considered forecasting techniques are applied to predict the number of VMs needed by an automotive application (e.g., Advanced Driving Assistance (ADA)) without impacting its performance (e.g., response time) as function of the variable number of cars that are passing through a street. Despite the specificity of the considered application the considered framework is general and can be applied to any time-varying request (e.g., VNFs for packet inspection, lightpath dynamic demands). Each VNF/VM is assumed to support the service required by a fixed number of cars.

The considered dataset is obtained from [11], where the number of vehicles of a specific street (Corso Belgio) in the city of Torino (in Italy) is reported every sixty seconds. In the paper, a dataset of forty-eight hours (two days) is considered.

The considered performance parameter is the prediction accuracy, represented by the Root Mean Square Error (RMSE) of the predicted values versus the time series real values. The performance is measured as function of the *lead time* (i.e., the future time for which the data need to be predicted). The *lead time* can be a function of the time required to (de)allocate the necessary resources and it depends upon a number of factors, such as type of service and utilised virtualization mechanism. The *window size* (n) is defined as the length of n previous observed data points considered to predict k number of data points (i.e., $k \geq 1$).

In DES and TES, the hyper-parameter values such as α, β and γ are selected to minimize RMSE, as summarized in Table I. The seasonality of the TES method is set to twenty-four hours. LSTM is implemented by using Google's TensorFlow library, accessed through the Keras high-level front-end. Table I reports the set of parameters that are used to evaluate the considered forecasting methods. The experiments are carried out on a workstation equipped with 8 cores Intel(R) i7-6820HQ 2.70GHz CPU with 16GB RAM, and running on Ubuntu 16.04 LTS 64-bit operating system.

Table I
Evaluation Parameters

Parameter	Forecasting Method	Value
Level factor (α)	DES, TES	0.9, 0.9
Trend factor (β)	DES, TES	0, 0.01
Seasonality factor (γ)	TES	0.9
Number of hidden layers	LSTM	2
Neurons in hidden layer	LSTM	100
Epochs	LSTM	100
Window size (n)	LSTM	10, 15, 20, 25
Batch size	LSTM	5
Dataset split (Train:Test)	LSTM	(70:30)

A. Impact of the dataset split ratio

Figure 3 shows the RMSE as function of LSTM-MSF and LSTM-MSF-recursive forecast methods with different dataset split ratios. The *window size* is set to 10 samples and the *lead time* is set to 5 minutes. For example, if the training versus testing (i.e., forecasting) proportion is set to $x : y$, it means that x% of the collected data are used for training while y% of the collected data are used for forecasting performance evaluation. Here, different training:testing ratios are considered to observe how accurate, in terms of RMSE, is the prediction. As shown in Figure 3, the prediction accuracy increases with an increase in training data size, however, at $70 : 30$ split ratio, the considered dataset provides minimum RMSE values. Hence, $70 : 30$ split ratio is an inflection point for the considered dataset. The best split ratio mainly depends on the total number of samples in the considered dataset and the model used for training.

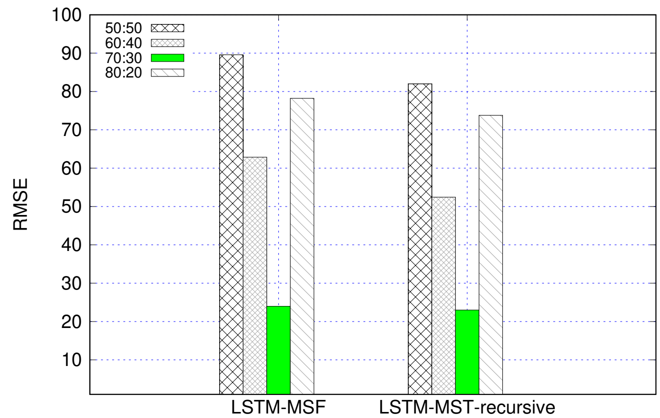

Figure 3. Impact of training and testing split ratio on the RMSE

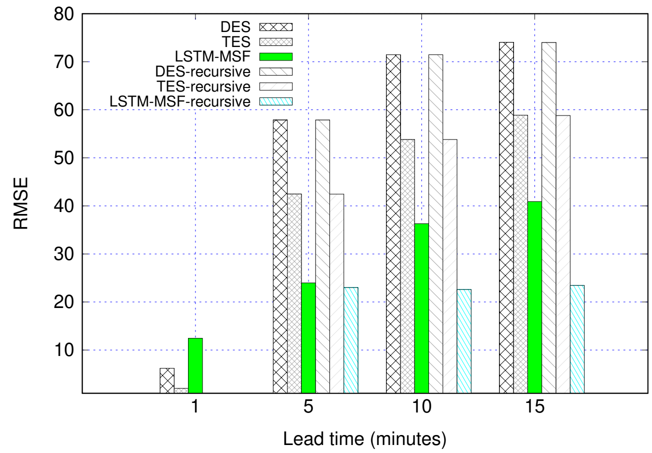

Figure 4. Impact of lead time on the RMSE with 70:30 split ratio

B. Impact of the lead time

Figure 4 shows the RMSE as a function of the *lead time* k with the six considered forecasting methods. The *window size* is set to 10 samples and the split ratio is set to $70 : 30$. For the considered dataset, when the lead time is set to one minute, the DES and TES methods outperform LSTM-MSF. However, when the lead time is long (i.e., 15 minutes), LSTM-MSF performs well compared to time series methods. In addition, LSTM-MSF-recursive method achieves the minimum RMSE value compared to LSTM-MSF method. Moreover, no significant changes are observed in case of *DES-recursive* and *TES-recursive* with DES and TES, due to the optimal selection of the α, β, and γ parameter values. Thus, for the considered dataset, if the lead time is short, methods based on exponential smoothing running at the edge can provide the best RMSE with a small contribution to the control loop cycle time.

In this way, the edge analytics is capable of providing a quick reply, meeting the low latency requirement of the service. Indeed, the prediction time for DES is around one μs, for TES is about 19 μs while for LSTM is about 3000 μs.

C. Impact of the window size

Figure 5 shows the RMSE as a function of the *window size* n with LSTM-MST and LSTM-MST-recursive forecasting methods. The split ratio is set to 70 : 30. Figure 5 depicts the RMSE values for three different lengths of lead time: 5 minutes (top), 10 minutes (middle), and 15 minutes (bottom). If the lead time is set to 5 minutes, the RMSE values of LSTM-MST-recursive and LSTM-MST are similar for all the considered window sizes. However, LSTM-MST-recursive outperforms LSTM-MST as the lead time increases. Moreover, LSTM-MST-recursive provides an almost stable RMSE value independent of window sizes.

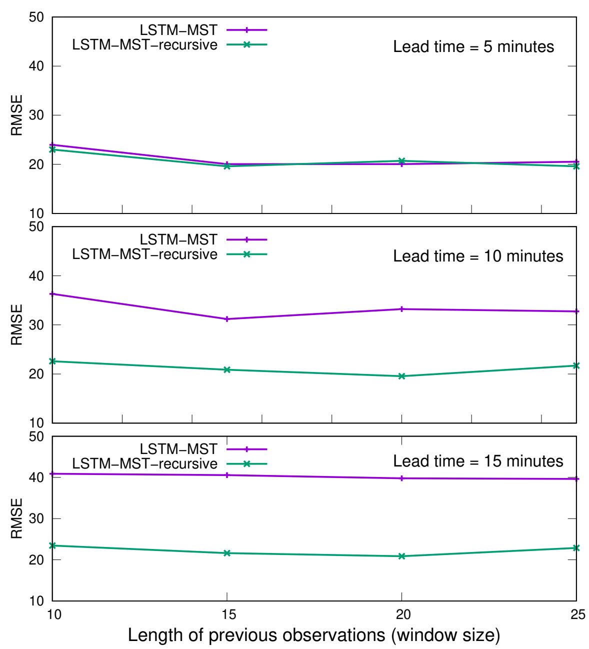

Figure 5. Impact of lead time and window size on the RMSE with different forecasting methods

V. CONCLUSIONS AND FUTURE WORK

This paper proposes a Hybrid Forecasting Framework (HFF) to implement the Analytics block of zero touch network and service management architecture proposed by ETSI. The HFF is based on implementing analytics/forecasting at both the network edge and the network core. Edge analytics is based on double and triple exponential smoothing while cloud analytics is based on a Recursive Neural Networks (RNN) method, more specifically Long Short-Term Memory (LSTM).

Performance evaluation results showed that edge analytics is capable of achieving a better forecast accuracy, measured in terms of root-mean-square error, than cloud analytics if the forecast value is in the near future (i.e., short lead time). Conversely, cloud analytics achieves better performance if the forecast value is in the far future (i.e., long lead time).

Thus, edge analytics is preferable when the lead time is short because it provides a better accuracy, at least with the considered dataset, and it allows for a short control loop cycle time by performing local forecast (i.e., short time for data transfer). Moreover, because it requires less resources and short prediction time, it is also energy efficient. If, instead, the lead time is long, core analytics is preferable because the longer prediction time, mainly due to the time to transfer the data, is compensated by a better accuracy. Moreover, because the lead time is long, a larger control loop cycle time is admissible. Thus, in general, if the time for (de)allocating resources is long (e.g., activating/scaling VMs), which implies a long lead time, cloud analytics is preferable.

The future work of this paper will focus on how multiple stages of the closed-loop blocks could impact on edge and cloud analytics. In addition, the dynamic (on-the-fly) selection of edge or cloud prediction will be included that helps intelligently scaling network functions and resource deployments based on the traffic fluctuations. Moreover, other techniques that incorporate the spatial features will also be considered to exploits the accuracy of the forecasting techniques.

ACKNOWLEDGEMENT

This work has been partially supported by the project "Scheme for Promotion of Academic and Research Collaboration (SPARC)", MHRD, Govt. of India and the EU Commission through the 5GROWTH project (grant agreement no. 856709).

REFERENCES

[1] ETSI GS ZSM 002, "Zero-touch network and Service Management (ZSM); Reference Architecture", V1.1.1, 2019.

[2] C. Benzaid and T. Taleb, "AI-driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions," in IEEE Network. doi: 10.1109/MNET.001.1900252

[3] J. R. Boyd, "The essence of winning and losing. Unpublished lecture notes", 12(23), 123-125, 1996.

[4] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," in IEEE Network, vol. 32, no. 6, pp. 42-49, November/December 2018.

[5] R. M. Shukla, S. Sengupta, and M. Chatterjee, "Software-defined network and cloud-edge collaboration for smart and connected vehicles," in Proc. of 19th ACM workshop ICDCN, pp. 6:1–6:6, 2018.

[6] Makridakis S, Spiliotis E, Assimakopoulos V, "Statistical and machine learning forecasting methods: concerns and ways forward", PLoS ONE 13(3):e0194889.

[7] T. Jirsik, Š. Trčka and P. Celeda, "Quality of Service Forecasting with LSTM Neural Network," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 2019, pp. 251-260.

[8] R.G. Brown, "Smoothing, forecasting and prediction of discrete time series", Dover Phoenix Editions, 1963

[9] Anne B. Koehler, Ralph D. Snyder, J.Keith Ord, "Forecasting models and prediction intervals for the multiplicative Holt–Winters method", International Journal of Forecasting,Volume 17, Issue 2, 2001.

[10] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] http://opendata.5t.torino.it/get_fdt