Design of AI-based Resource Forecasting Methods for Network Slicing

Juan Sebastian Camargo[*†], Estefanía Coronado[*], Blas Gómez[‡], David Rincón[†], and Shuaib Siddiqui[*]
[*]i2CAT Foundation, Barcelona, Spain. Email: {juan.camargo, estefania.coronado, shuaib.siddiqui}@i2cat.net
[†]Department of Network Engineering, Universitat Politècnica de Catalunya (UPC), Castelldefels, Barcelona, Spain.
Email: juan.sebastian.camargo.barraga@estudiantat.upc.edu, david.rincon@upc.edu
[‡]High-Performance Networks and Architectures, Universidad de Castilla-La Mancha, Albacete, Spain.
Email: blas.gomez@uclm.es

***Abstract*—With the forthcoming of 5G networks, the underlying infrastructure needs to support a higher number of heterogeneous services with different QoS needs than ever. For that reason, 5G inherently provides a way to allocate these services over the same infrastructure through the concept of Network Slicing. However, to maximize revenue and reduce operational costs, a method to proactively adapt the resources assigned to each slice becomes imperative. For that reason, this work presents two Machine Learning (ML) models, leveraging Long-Short Term Memory (LSTM) and Random Forest algorithms, to forecast the throughput of each slice and adapt accordingly the amount of resources needed. The models are evaluated using NS-3, which has been integrated with the ML models through a shared memory framework. This enables a closed loop in which the predictions of the models can be used at run time to introduce changes in the network. Consequently, it makes it able to cope with the forecasted requirements, eliminating the need for off-line training and resembling better a real-life scenario. The evaluation performed shows the ability of the models to predict the slices' throughput under various settings and proves that Random Forest provides up to 26% better results than LSTM.**

***Index Terms*—Network Slicing, Machine Learning, Network Simulation, Random Forest, LSTM**

I. Introduction

As 5G becomes widely available, the industry is moving towards new ways of operating the networks. The new applications enabled by 5G demand that the network is able to allocate different types of services over the same infrastructure under strict Service Level Agreements (SLAs). For that reason, 5G inherently provides a way to grant the needed resources to the different coexisting services through the concept of network slicing. Network slicing creates different logical networks over the same physical infrastructure, assigning to each service the necessary resources. This allows the Mobile Network Operators (MNOs) to make a better use of their infrastructure, which increases profit. However, the orchestration of such slices remains a challenge on current research. Efficient resource allocation, maximizing profit and minimizing operational expenditures seems imperative. This can be achieved by fitting the maximum number of services in the network while preserving the SLAs and, in the same way, withholding the previously allocated resources when they are not being used. For this purpose, it seems clear that MNOs need to be able to anticipate the changes in the throughput that each slice is handling to be able to proactively allocate or withhold new resources for each slice.

At this point, there is a global trend that Artificial Intelligence (AI) and, in particular, Machine Learning (ML) needs to be incorporated into the future mobile networks to approximate such complex optimization functions. Current research has shown that ML models can be used to predict the behavior of different Key Performance Indicators (KPIs) based on past experiences and take autonomous decisions based on those predictions [1], [2]. In the case of slicing, ML can be used to predict the throughput that a slice is going to handle. This can be used to scale the resources assigned to that particular slice accordingly. However, to make accurate predictions, the models need to be trained with a large amount of data.

On this basis, network simulation becomes an essential tool, as it allows creating virtual models of the network to be deployed in reality. Moreover, they can be used to generate big amounts of input data in a much shorter period of time, which is paramount given the difficulties to acquire training data from operational networks. However, the existing approaches do not take advantage of the full potential of the simulators, as they are used only to generate data which is then used offline. For this reason, classic network simulators, such as NS3 [3] or OMNeT++ [4], need to introduce ML in their workflow. In this context, NS3-AI [5] has been developed to allow the interconnection between the simulator and the most widely used state-of-the-art ML libraries, such as Keras or Sklearn.

In this work, we present two ML models to forecast the expected throughput on a network slice at a future point: Long-Short Term Memory (LSTM) and Random Forest. The models base their predictions on past experiences of throughput and Radio Access Network (RAN) metrics.

LSTM can capture previous trends from the data it is fed upon and then use such tendencies in future predictions which fits this work's use case. On the other hand, Random Forest is a classifier method that needs data preprocessing to be used as a model with time persistence. This work seeks to compare the efficiency and precision of these two models. Based on the obtained prediction, we use the shared memory environment provided by NS3-AI to adapt the resource blocks in the radio nodes assigned to each slice in run time in contrast to the existing literature. This setting facilitates online training and provides a better affinity with real life.

The rest of the paper is organized as follows. Section II discusses the related work. In Sec. III the design of the forecasting methods and the implementations details are introduced. Sec. IV presents the performance evaluation and finally, in Sec. V the conclusions are presented.

II. Related Work

A. *Slicing Resource Prediction*

ML approaches applied to network slicing have received the focus of academic and industrial research due to its high prediction accuracy and generalization capabilities, allowing them to operate efficiently in mobile networks. However, the traffic being transmitted through the slices is dynamic but the capacity that lies underneath is static, generating a conflict of under or overprovisioned capacity. Several authors have worked over this issue thoroughly [1], [2], [6]–[11].

The works presented in [6] and [7] use a Deep Neural Network (DNN) and LSTM as prediction models. They use the sliding window technique in a fully-connected hidden layer topology with an output of the sliding's size. In particular, the work in [6] uses the historical service provider data (i.e., traffic demand, time, resources assigned) to forecast the amount of resources needed to guarantee the SLA. On the other hand, the authors of [7] modify the loss function so it includes multi-domain KPIs input for the predictor, and allows the incoming traffic prediction to be related to all the involved domains in the network. While the work in [6] compares the ML models with an Autoregressive Integrated Moving Average (ARIMA) model, the approach presented in [7] creates an additional Convolutional Neural Network (CNN). Additionally, the objective function of the ML models in [7] is modified to use a multi-domain capacity that needs to be minimized. In the same line, the authors of [8] use ARIMA as a non-ML model that is compared with an LSTM and a Support Vector Regressor, with the LSTM overperforming the other two. This approach is based on a spatiotemporal model that seeks a co-relation between the incoming traffic and the physical location and time of the UE. Additionally, LSTM, Time-Delay Neural Networks and Support Vector Machine models are used in [9] to forecast the needed resources of a virtualized base station with the RAN parameters as input of the models.

On a different line, the authors of [1] and [2] use per-slice throughput prediction to assign radio network resources in the form of the Physical Resource Blocks (PRB) to predict the best wireless resource allocation. However, there is a considerable difference in the approach in terms of ML selected. In [1] a web-based algorithm is presented, taking as input the number of available time-slots of the network and their renting costs. By contrast, in [2], the authors use a regressor tree with the PRB historically used per equipment, per slice type, and the currently requested PRB. However, both works perform the prediction offline. Similarly, the approach introduced in [10] leverages Reinforcement Learning to minimize the probability to incur a violation of the SLA due to underprovisioning, avoiding penalties that can considerably affect the MNO's reputation and budget. Finally, using a three-dimensional CNN, the work in [11] forecasts the network capacity needed to cope with the incoming traffic. The three CNNs are known for achieving a high prediction level with a minimum training, helping reduce the information gathering and training time and resources.

The body of literature using LSTM to predict throughput in mobile networks is extensive due to its native capabilities for time-series analysis and sequential inputs. However, Random Forest is seldom used for network slicing predictions as, opposite to LSTM, it is not naturally suited for sequential information. This work seeks to compare the results of two opposite models and find the settings on which they provide a greater performance. Notice that it is not a common practice to use the output obtained from the ML models back into the simulation process. Instead, in most of the literature, the predictions are performed offline. For that reason, the simulation processes rarely take full advantage of the prediction value, if any. A key part of this work is the use and fine-tuning of a common framework that allows the interconnection of the simulation with the prediction models, generating a control loop that more closely resembles reality as it allows the use of the predictions back into the simulation during its execution.

B. *Support for AI Pipelines in Network Simulators*

NS3 is a de-facto standard simulator for academic research in networking technologies. With the landing of ML in next-generation networks, there is a need to integrate these algorithms in network simulators. To provide an answer to this need, the authors of [12] presented a framework that integrates OpenAI Gym and NS3. This framework has enabled the use of Reinforcement Learning (RL) in many research works, such as the work in [13], which introduces an RL agent that adapts transmission rate in Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) wireless networks, and the approach in [14], which presents a deep RL approach for congestion control in Centralized RAN (C-RAN) scenarios.

Inspired by [12], the authors of [5] present NS3-AI, which provides a faster and more efficient solution to enable data interaction between NS3 and Python-based AI frameworks. It presents an enhancement of ns3-gym as it exploits shared memory to allow a seven times faster transmission of the data between NS3 and the AI frameworks, which makes it perfect for the implementation of the models used in this work.

Contributions: Based on the above, this work has two main contributions: *(i)* to compare one ML model that natively handles time-series data for slice throughput forecasting against one ML model not natively suited to do it; and *(ii)* to create a bi-directional information exchange system that actively shares information between the prediction framework and the simulation framework, to use the said prediction as input in the simulation and apply changes into the network in run time.

III. Design of ML-based Per-slice Throughput Forecasting Methods

This work presents two ML models to forecast the throughput expected in network slices and validates them using a simulation environment that changes the network capacity at execution time according to the model's predictions. Given the over-costs generated by resource overprovisioning and the SLA penalties due to underprovisioning, predicting the traffic that is going to traverse an slide is of utter importance to optimize network resources and decrease operational costs. The data obtained from the network which is going to allow this prediction is presented as a series of sequential time samples (time series). LSTM naturally masters this structure and is capable of generating long-term dependencies [15]. As the releated work reveals, it is clear that Recurrent Neural Networks (RNNs) are the way to go in time-series forecasting in networking. However, models out of this category are seldom used. For this reason, this work also studies Random Forest, given its high resistance to overfitting and generalization characteristics, which adapt to the dynamic nature of the mobile network that is evaluated.

Both ML design's used in this work represent a multivariate time-series model $H(X)$ that, given an ordered input matrix $X = (X_1, ..., X_n) \in \mathbb{R}^{mXn}$ of n features, and m entries, produces an ordered output sequence y at time t. Therefore, denoting $X_j^{(i)}$ as the element at row i and column j in X, each entry row is a vector $X^{(i)} = (X_1^{(i)}, ..., X_n^{(i)})$, while a specific value of a feature j represents a sequence of rows $(X_j^{(1)}, ..., X_j^{(m)})$. In particular, the model $H(X)$ uses X to predict the throughput per slice, y, for a future time $t + 1$. The chosen models are trained using the dataset in [16]. This dataset comprises a set of time series describing channel indicator parameters and the throughput generated by a hundred UEs (equally divided into uplink and downlink transmissions), during 490 seconds, with a sampling frequency of 250 ms. The data comes from a cellular network with three UE mobility patterns (pedestrians, cars, and trains) and 21 individual cell sectors.

The training dataset needs pre-processing to clean values not useful for this use-case. First, all the identifiers are deleted as they do not provide meaningful information for training. Additionally, as the models focus on predicting the UE's downlink traffic, only the downlink UEs are considered. Finally, since the ML models measure the distance between the variables to infer the output values, the data has been scaled in the range [0, 1] to avoid higher numbers generating a bias in the prediction. After this, a 10-feature vector, X, is ready to be used as the training's input. Namely: total block size, channel quality indicator, competitive throughput CQI, new data indicator, reference signal received, power, signal interference noise ratio, competing for SINR, delay, and total block error rate. With this input, we aim to predict the throughput expected on a network slice in the next time-step, $t + 1$. Although it is possible to choose any future time-step, the analysis has shown that the prediction error propagates along with the following estimations because the prediction at time t+2 takes as input the $t + 1$ result. Finally, the prediction y shows the output scenarios: increase, decrease or maintain the PRBs of the radio nodes, with the final goal of employing only the necessary resources to handle the expected throughput in the slices.

A. Random Forest

Random Forest is based on several decision trees using different randomly-selected samples of the input data to prevent overfitting in the predictions. Each decision tree is based on a subdivision of all the possible pairs of features, X_i, and predictions, y_i, defined as the instance space. The first node divides the space into two concrete sub-spaces based on the feature with the higher division possibility, being it a range of said feature. The process is then repeated for the remaining features. Random Forest provides a high capacity for adaptability and generalization, which works with the changing environments of mobile networks. In this work, the implementation of Random Forest is formed by 100 individual regression trees, with a 25% dropout possibility. Bootstraping is used during the training process to allow each decision tree to be trained with a random sampling of the original dataset. This prevents overfitting and eliminates any existing bias in the sequential data.

By default, Random Forest does not capture historical dependencies of the input data. To solve this issue we have used the sliding window technique. This method includes previous values of the variables as a way to track trends and give relevance to historical values within the predictions. The previous values of the target feature, y, are included into the multi-variate time-series as additional variables, representing $t - 1, t - 2, ..., t - l$ where l is the size of the sliding window. Random Forest allows calculating the variable importance in the regression process by weighting how much each variable affects the variance of the regression. Initially, a sliding window of $l = 10$ was chosen, but the variable importance decreased considerably after the third time step. For that reason, only the three previous steps and the current throughput value are considered and used as the final window size. These values are incorporated to the vector X. Moreover, Reference Signal Received Power (RSRP) and Signal Interference Noise Ratio (SINR), show marginal variable importance. For that reason, only RSRP and SINR, together with the throughput at time t, $t - 1$, $t - 2$ and $t - 3$, are selected in the final training dataset composed of 6 features.

B. LSTM

LSTM is based on a RNN topology with modified neurons that uses gates. The gates create long-term dependencies in the system that enables the model to interpret historical patterns and apply that knowledge to future predictions. Gates are mathematical models that allow the neuron to regulate what the system can learn, forget or maintain in every iteration. Traditionally, LSTM topology only has one LSTM layer connected with one dense layer to provide the forecasted value. However, in order to design a model that is able to adapt to the variable throughput patterns, a stacked LSTM structure is used. This type of model has additional LSTM layers that are connected among them in a topology where the input of a layer is the output of the preceding one. This increases the ability of the model to analyze complex input patterns, such as the ones present in mobile networks. The topology used for this work consists of three LSTM hidden layers and one dense layer acting as the output. Each of the three LSTM layers are formed by an array of 128 neurons, with a 25% dropout layer in-between. The input of the model is formed by a bi-dimensional 6x3 array, where the first dimension represents the features (SINR, RSRP and throughput in times $t, t-1, t-2$ and $t-3$) and the second dimension represents the values of those features in three previous time steps.

Finally, the output of the third layer is the input of a fully connected dense layer, providing the predicted throughput. The prediction model has the possibility of using a three-step prediction in the future ($t+1, t+2$ and $t+3$), but for this work, only the first step ($t+1$) was considered. Usually, on a multi-step predictor, the error increases with the increment of multi-steps in the future, reducing the prediction's accuracy.

The additional layers let the model abstract the characteristics of the network and make connections that adapt one specific layer to one single characteristic. The model uses the LSTM layers as a way to particularize each of the characteristics of the previous time-steps ($t-1, t-2, t-3$) into each of the three layers, respectively. The dataset is divided into 25% for test and 75% for training, with a batch size of 32, 15 epochs and a learning rate of 0.1%. Notice that the fine-tuning process of finding the optimal hyperparameter values is frequently done experimentally, following a trial and error cycle, choosing the parameters that better suit the model.

C. Implementation Details

LSTM is implemented in Python using Keras, while Random Forest is implemented using Scikit-learn. When the training and validation of the prediction models has finished, they are tested in an NS3 simulation. NS3 is a discrete event simulator capable of deploying mobile network architectures in an end-to-end format. However, NS3 runs as a closed framework, and it does not provide a native interface to use the most extended ML libraries, such as Keras, Scikit-learn and Pytorch. For that reason, NS3-AI [5] is used to enable the interconnection of NS3 with the aforementioned libraries through the use of a shared memory pool, as shown in Figure 1. This process enables the exchange of information among the prediction and the simulation frameworks in run time, creating a closed-loop cycle in-between.

The same chosen variables used as input for the training dataset (RSRP, SINR, and throughput) are periodically sent (every 50 milliseconds) to the prediction model using the aforementioned shared memory pool. The prediction model gathers and transforms the information into a multi-variate time-series with three steps sliding window that acts as an input of the models. At the same time, the model provides a prediction that is used to determine an action, i.e., increasing, decreasing, or maintaining the number of PRBs assigned to the radio nodes. This action is then sent back to NS3 through the same memory pool to adjust the network accordingly. This complete cycle is repeated until the simulation time is over.

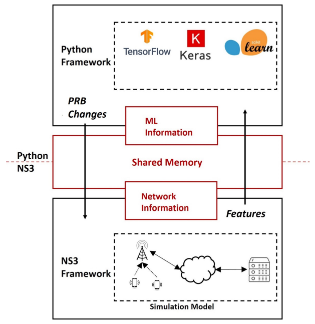


Fig. 1: NS3 and Python shared memory pool.

TABLE I
SCENARIOS DESCRIPTION FOR EVALUATION CAMPAIGN

Parameter	Sce. 1	Sce. 2	Sce. 3	Sce. 4
Active UE	15	10	10	15
Intermittent UE	0	10	10	0
1st interval	0	5	5	0
2nd interval	0	3	3	0
On-Off cycles	0	3	3	0
Radio nodes	3	3	3	6
Mobility Pattern	GM	GM	S	GM
Initial Speed	15 m/s	15 m/s	0	15 m/s
DL/UL PRB	D	D	D	F

IV. Performance Evaluation

A. Methodology

Four scenarios are used to test the designed ML models. Each scenario contains an area of 750 x 750 meters where 15 UEs (for scenarios 1 and 4) and 20 UEs (for scenarios 2 and 3) are randomly placed. For scenarios 1, 2 and 4, the UEs follow a Gauss-Markov mobility pattern, whereas in scenario 3 the UEs are stationary. The radio nodes are connected to a Packet Gateway Core, which in turn is linked to the Internet and then to a remote host acting as the content provider. Afterwards, a UDP server is installed in all UEs and configured in a point-to-point network with a 20 Mbps throughput and a Maximum Transfer Unit (MTU) of 1200 bytes to the remote host. The radio nodes are configured with a default PRB value (50 PRBs) at the beginning of the simulation and are adapted through the simulation according to the prediction of the ML models. Each experiment simulates 25 seconds. Six slice types are created for each scenario and each UE is assigned to a particular slice. Additionally, to guarantee a minimum level of service for all users, a Proportional-Fair Scheduler (PFS) is used. A PFS is a radio scheduler in charge of managing the RAN among the UEs offering a trade-off between having a high throughput and providing the required service to all the UEs. The scenarios are described in Table I, where *GM* stands for Gauss Markov,

S for Stationary, D for Dynamic (i.e., PRBs changes based on the predictions) and F for Fixed (i.e., PRBs are fixed).

Scenarios 2 and 3 have an intermittent traffic pattern with the following behavior: the UEs delivering intermittent traffic are activated during five seconds, followed by an idle state of three seconds and then activated again for the next five seconds. The first intermittent UE starts at second 3 of the simulation and every other UE have an additional start-delay of 0.5 seconds among each other. In order to acquire meaningful data from the experiments, each scenario is repeated 25 times. In the following section the average of these 25 executions is shown.

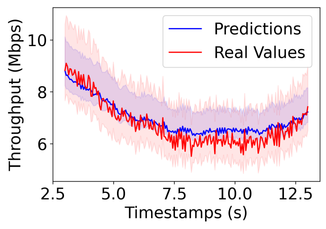
B. Results Discussion

To evaluate the accuracy of the predictions, the Mean Squared Error (MSE), the Mean Absolute Error (MAE), the Root Mean Squared Error (RMSE), the Explained Variation Score (EVS), the Median Absolute Deviation (MAD) and R^2 scored are computed. Each ML model is provided with the same data to guarantee that the results are compared under the same conditions. A summary of the error metrics is provided in Table II. Moreover, for all scenarios a comparison between the real throughput of the slices and the forecasted one is presented.

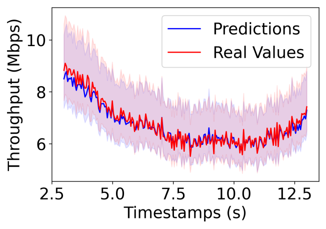
TABLE II
ERROR METRICS FOR ALL SCENARIOS

Error	Model	Sce. 1	Sce. 2	Sce. 3	Sce. 4
MSE	*LSTM*	0.154	0.162	4.063	0.133
	RF	0.045	0.038	0.128	0.036
MAE	*LSTM*	0.321	0.335	1.716	0.314
	RF	0.170	0.162	0.302	0.156
RMSE	*LSTM*	0.396	0.402	2.015	0.365
	RF	0.212	0.194	0.358	0.192
EVS	*LSTM*	0.846	0.430	0.454	0.757
	RF	0.956	0.849	0.981	0.915
MAD	*LSTM*	0.260	0.301	1.690	0.306
	RF	0.142	0.145	0.290	0.132
R2	*LSTM*	0.784	0.141	0.422	0.589
	RF	0.937	0.732	0.981	0.886

The results of scenario 1 are shown in Figure 2 comparing the ground-truth of the throughput with the value forecasted by the ML models. From a graphical analysis, it is possible to observe that LSTM underperforms Random Forest's predictions. LSTM's predictions follow the traffic trend properly, but Random Forest generates a smoother predicted signal, being able to follow the spikes of the throughput faster and more precise. This is confirmed in Table II, which shows that Random Forest's error metrics are, overall, better. However, LSTM's MAE and MAD metrics are low when compared with the absolute traffic in the scenarios, making it still a viable predictor. This final point also holds for scenario 2, for which in Table II can be seen that the MAE and MAD metrics are similar to scenario 1. Scenario 2 includes the intermittent traffic pattern, forcing the models to predict based on previously unknown information. Because of that, the EVS and R^2 metrics in both models decreased compared with the initial scenario. Additionally, it is possible to see a biased error in scenario 2 due to the EVS value being almost half the R^2 value. This behavior is linked with the model's inability to generalize, meaning that the models are memorizing the training information and are not able to correctly predict any related information. This fact can be verified in the results in terms of forecasted traffic depicted in Figure 3.

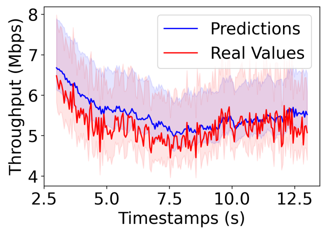


(a) LSTM.

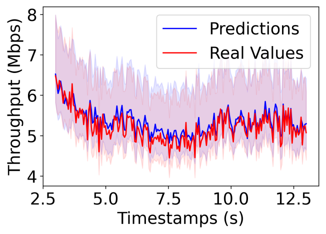


(b) Random Forest.

Fig. 2: Global throughput prediction vs. real value in scenario 1.



(a) LSTM.

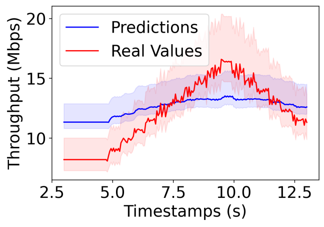


(b) Random Forest.

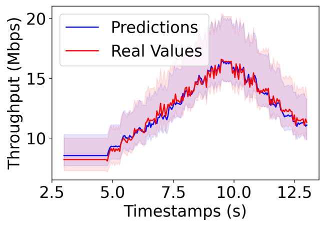
Fig. 3: Global throughput prediction vs. real value in scenario 2.

Regarding scenario 3, both EVS and R^2 error metrics plummeted compared with the previous two cases for LSTM, as shown in Table II. This could happen because scenario 3 is the one that changes the most of all the four scenarios. The UEs in scenario 3 do not move, compared with the always moving UEs in the other scenarios. This behavior could affect the traffic pattern and the way the prediction models provide the forecasted throughput, directly affecting the error metrics. Additionally, for LSTM, the MAE and MAD values increase, being up to 25% of the total forecasted value. These absolute errors, in hand with the lower EVS and R^2 values show that LSTM is not suitable to be used under these circumstances. Conversely, Random Forest's metrics show that the quality of the prediction stayed the same, with low MAE and RMSE and high values of EVS and R^2, proving to be a valid model. Furthermore, in this scenario, EVS and R^2 are the same, which corresponds to a prediction with minimal or zero bias error, as shown in Figure 4, which depicts the great difference in the ability of both models to generalize the predicted throughput on more diverse network settings.

Finally, the accuracy results of scenario 4 are shown in Table II. By analyzing these values, it is possible to see that again the numbers are on the side of Random Forest, but not excluding LSTM as a prediction model. Additionally, it can be noted that both models have a small bias error, based on



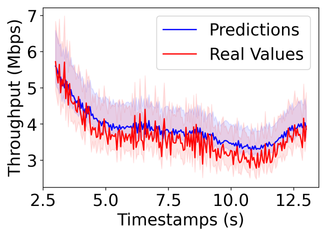
(a) LSTM.



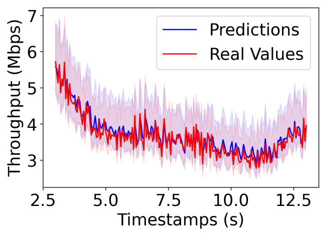
(b) Random Forest.

Fig. 4: Global throughput prediction vs. real value in scenario 3.

the differences of the EVS and R^2. The same conclusions can be drawn from the throughput comparison depicted in Figure 5. This scenario in particular has a higher network capacity, by using double of the radio nodes, but this does not have any effect over the prediction models as the MAE, MAD and RMSE metrics are similar to scenarios 1 and 2, providing a predicted value in the same order of magnitude as the throughput value. The absolute errors are good ways of determining the prediction quality but finally the error values need to be analyzed by comparing them with the application in which the model is going to be deployed, as the margins of acceptance changes within different deployments.



(a) LSTM.



(b) Random Forest.

Fig. 5: Global traffic prediction vs. real value in scenario 4.

V. Conclusions

In this paper, we have presented two ML-based throughput forecasting methods for network slicing that proactively improves wireless resource allocation for each network slice, reducing costs while preserving the SLAs. Using NS3 network simulator and NS3-AI framework, a closed loop where the actions predicted by the ML models could be used in run time has been created. In this setting, an LSTM and a Random Forest model have been designed, being the second the one that has shown a better performance for the use case presented in this paper. The evaluation carried out has shown that Random Forest fits the observed data 26% better than the other model. LSTM lacks generalization and its performance drops for those scenarios that differ the most from the training benchmark. As future work, LSTM requires further analysis as it has not shown the expected capacity for generalization, requiring tests with bigger datasets and other neural network models. Moreover, we plan to validate the results on a real-world testbed using various traffic and UEs types.

Acknowledgment

This work has been supported by the EU's H2020 project Affordable5G (957317) and PID2019-108713RB-C51 MCIN/ AEI /10.13039/501100011033. This work is part of the R&D project RTI2018-098156-B-C52 supported by the MCIN and the European Regional Development Fund: "a way of making Europe". This work was also supported by the Government of Castilla-La Mancha (project SBPLY/17/180501/000353), and Universidad de Castilla-La Mancha and the European Union's Social Fund (Grant 2019-PREDUCLM-10921). The authors would also like to acknowledge CERCA Programme / Generalitat de Catalunya for sponsoring this work.

References

[1] R. Abozariba, M. Kamran Naeem, M. Asaduzzaman, and M. Patwary, "Uncertainty-aware RAN Slicing via Machine Learning Predictions in Next-Generation Networks," in *Proc. of IEEE VTC2020-Fall*, Victoria, BC, Canada, 2020.
[2] N. Salhab, R. Langar, R. Rahim, S. Cherrier, and A. Outtagarts, "Autonomous Network Slicing Prototype Using Machine-Learning-Based Forecasting for Radio Resources," *IEEE Communications Magazine*, vol. 59, no. 6, pp. 73–79, 2021.
[3] G. F. Riley and T. R. Henderson, "The NS-3 Network Simulator," in *Modeling and Tools for Network Simulation*, 2010, pp. 15–34.
[4] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Marseille, France, 2008.
[5] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "Ns3-Ai: Fostering Artificial Intelligence Algorithms for Networking Research," in *Proc. of WNS3*, Gaithersburg, MD, USA, 2020.
[6] J.-B. Monteil, J. Hribar, P. Barnard, Y. Li, and L. A. DaSilva, "Resource Reservation within Sliced 5G Networks: A Cost-Reduction Strategy for Service Providers," in *Proc. of IEEE ICC Workshops*, Dublin, Ireland, 2020.
[7] L. A. Garrido, P.-V. Mekikis, A. Dalgkitsis, and C. Verikoukis, "Context-Aware Traffic Prediction: Loss Function Formulation for Predicting Traffic in 5G Networks," in *Proc. of IEEE ICC*, Montreal, Canada, 2021.
[8] 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. of IEEE INFOCOM*, Atlanta, GA, USA, 2017.
[9] R. Guerra-Gómez, S. R. Boqué, M. García-Lozano, and J. O. Bonafé, "Machine-Learning based Traffic Forecasting for Resource Management in C-RAN," in *Proc. of EuCNC*, Dubrovnik, Croatia, 2020.
[10] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proc. of IEEE INFOCOM*, Atlanta, GA, USA, 2017.
[11] 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. of IEEE INFOCOM*, Paris, France, 2019.
[12] P. Gawłowicz and A. Zubow, "NS-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research," in *Proc. of ACM MSWiM*, Miami Beach, USA, 2019.
[13] S. Cho, "Reinforcement Learning for Rate Adaptation in CSMA/CA Wireless Networks," in *Advances in Computer Science and Ubiquitous Computing*. Springer, 2021.
[14] I. Nascimento, R. Souza, S. Lins, A. Silva, and A. Klautau, "Deep Reinforcement Learning Applied to Congestion Control in Fronthaul Networks," in *Proc. of IEEE LATINCOM*, Salvador, Brazil, 2019.
[15] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, J. Kolen, and S. Kremer, "A field guide to dynamical recurrent neural networks," *chapter Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*, pp. 237–243, 2001.
[16] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics," in *Proc. of ACM MSC*, Amsterdam, the Netherlands, 2018.