Mobile traffic forecasting using a combined FFT/LSTM strategy in SDN networks

Mohammed Lotfi Hachemi, Abdelghani Ghomari, Yassine Hadjadj-Aoul, Gerardo Rubino

Mobile traffic forecasting using a combined FFT/LSTM strategy in SDN networks

Mohammed Lotfi Hachemi[1], Abdelghani Ghomari[1], Yassine Hadjadj-Aoul[2] and Gerardo Rubino[3]

Abstract—Over the last few years, networks' infrastructures are experiencing a profound change initiated by Software Defined Networking (SDN) and Network Function Virtualization (NFV). In such networks, avoiding the risk of service degradation increasingly involves predicting the evolution of metrics impacting the Quality of Service (QoS), in order to implement appropriate preventive actions. Recurrent neural networks, in particular Long Short Term Memory (LSTM) networks, already demonstrated their efficiency in predicting time series, in particular in networking, thanks to their ability to memorize long sequences of data. In this paper, we propose an improvement that increases their accuracy by combining them with filters, especially the Fast Fourier Transform (FFT), in order to better extract the characteristics of the time series to be predicted. The proposed approach allows improving prediction performance significantly, while presenting an extremely low computational complexity at run-time compared to classical techniques such as Auto-Regressive Integrated Moving Average (ARIMA), which requires costly online operations.

Index Terms—SDN, LSTM, time-series, forecasting, Fast Fourier Transform.

I. Introduction

The continued emergence of new and constrained services is one of the major challenges facing Infrastructures' Providers (InPs) [4]. In order to cope with these new trends, InPs are currently shifting their hardware-based equipment to much more agile software-based solutions. This has been made possible through the separation of the control plane from the data plane, with the Software Defined Networking (SDN) paradigm, and the decoupling of network functions from their hardware, with the Network Function Virtualization (NFV) concept [18].

Although these technologies enable InPs to meet their objectives, they nevertheless introduce new challenges [9]. One of the major ones is the scaling up (respectively down) of functions to accommodate an increase (respectively decrease) in the workload. Indeed, scaling-up network functions involves the deployment of one or more new instances, which generally requires several seconds or even minutes. As a consequence and in order to avoid any congestion, several studies in the literature focus on load prediction to anticipate its possible increase [1].

Forecasting time-series evolution is a fairly classical field of research [8] with a vast associated literature. Initial approaches were generally based on exponential smoothing or Auto-Regressive Integrated Moving Average (ARIMA) models [20]. Although effective, these techniques have recently been superseded by deep learning-based strategies [15]. Among them, Recurrent Neural Networks (RNN) are certainly the most effective in recognizing traffic patterns, thanks to their recurrent structure. In particular, Long Short Term Memory (LSTM) networks outperform the majority of existing techniques and do not present the problem of the vanishing gradient from which RNNs typically suffer [14].

In this paper, we propose to go beyond existing work by putting forward a new approach to the problem of time series forecasting. The idea is to combine an LSTM network with a feature extraction module in order to facilitate the task of identifying patterns in the considered time series. In particular, we analyze the capacity of the Discrete Fourier Transform (DFT) [3] to extract the most relevant features from the time series, while canceling noise-like signals, which may affect learning. This combination not only decreases significantly the prediction error, but can also reduce the number of required neurons, thus accelerating the learning process. In addition, it maintains pretty good performance over longer prediction horizons, as can be seen in the sequel.

The rest of this paper is organized as follows. In Section II, we introduce a classification of the related work regarding time series forecasting. In Section III, we describe our proposed solution. Its performance evaluation and its comparison with existing approaches are the object of Section IV. The paper concludes in Section V.

II. Related Work

Several techniques have been proposed in the literature for function prediction, and more particularly for time series forecasting. They can be broadly classified into three main categories: statistical, machine learning-based and strategies mixing both types of methods. Although the list of techniques presented below is not exhaustive, this classification allows us to properly position the existing work and our approach.

A. Statistical-based approaches

Statistical approaches are traditionally used to make forecasts. One of the most widely recognized procedures for time series prediction is the Auto-Regressive Integrated Moving Average (ARIMA) one [20], which has been used in many studies. This technique has proven to be very effective when the data are stationary and show no upward

[1] University Oran 1 RIIR Laboratory, Oran, Algeria
[2] Univ. Rennes, Inria, CNRS, IRISA
[3] Inria, Univ. Rennes, CNRS, IRISA

or downward tendencies. However, the effectiveness of ARIMA remains relevant for linear systems only, making it inappropriate in many cases.

In [7], the Empirical Mode Decomposition (EMD)-based Multi-Model Prediction (EMD-MMP) algorithm has been proposed in order to predict network traffic in Software-Defined Networks. EMD-MMP combines the action of three methods: EMD [11], [19], Auto-Regressive Moving Average (ARMA), and Support Vector Machine (SVM) - Regression (SVR) methods. Unlike ARIMA, this technique can be used to deal with non-linear signals and appears to be suitable for a variety of use cases.

Although statistical techniques are still relevant, they have recently been surpassed by machine learning-based ones [15].

B. Machine learning-based approaches

Many recent studies are based on machine learning algorithms, which have the advantage of not requiring an explicit mathematical model. Moreover, once trained, the predictions are almost instantaneous, in contrast with statistical approaches. The LSTM neural network [10] is a top procedure in this category. It is based on a recurrent neural architecture that can process time series and is capable of learning and remembering input data over long sequences. This leads to its wide usage in time series prediction. In [15], the authors compare the accuracy of ARIMA and LSTM for predicting time series data, resulting in a very significant superiority of LSTM over ARIMA.

The success of the LSTM algorithm in predicting or recognizing sequences has led to the emergence of several extensions. Among them, the Bidirectional LSTM algorithm is one of the most efficient variants [16]. The idea behind this technique is to consider not only the past values of the sequence but also the future ones, thus improving its prediction performance compared to a classical LSTM.

C. Combined approaches

Some approaches combine different techniques whether statistical or machine learning-based or even others. The M4 competition has shown that combined prediction methods have a higher accuracy than non-combined ones [12].

In [2], the authors proposed using a Diffusion Convolutional Recurrent Neural Network (DCRNN) to forecast network traffic. They used two layers of Diffusion Convolutional Gated Recurrent Unit (DCGRU). The comparison of this approach with an LSTM-based network, a CNN-based network, a CNN-LSTM-based network and a Fully Connected Neural Network show the superiority of DCRNN in forecasting congestion events.

The approach we propose in this paper falls into this category of methods, since we combine the FFT and an LSTM neural network.

III. A combined FFT/LSTM strategy for time-series prediction

In this section, we will introduce our approach, which consists in combining FFT data pre-processing with an LSTM neural network. In what follows, we detail the proposed architectures and the input/output conditioning to make prediction efficient, even for large measurement horizons.

A. Input/Output

Our work involves developing a component that receives old data as input to predict new data as output.

The data is a sequence of chronologically ordered values, each value d_i representing the bandwidth read at time step i. From this data sequence, plus a sliding window $d_{i+1},\ d_{i+2}, \ldots,\ d_{i+n}$ of size n and the corresponding predicted value $\hat{d}_{i+n+m}$ (integer $m \geq 1$ is called the prediction horizon), we generate a set of pairs (vector of values in sequence of size n, predicted value) where the vectors will be used as input to our component and the predicted value will be its output (see Fig. 1). We write

$$\hat{d}_{i+n+m} = \Phi(d_{i+1},\ d_{i+2}, \ldots,\ d_{i+n}) \tag{1}$$

with Φ representing the prediction function.

We opted for a sufficiently large measurement horizon to avoid making multiple predictions before reaching the desired value, thus avoiding the propagation of errors. Indeed, given that we are at time step $i+n$, predicting the value at time $i+n+m$ implies the prediction of all the values before this last one. This implies a prediction from inputs with successive estimation errors. To overcome this problem, we proposed to estimate the value at time step $i+n+m$ directly, without resorting to the intermediate values.

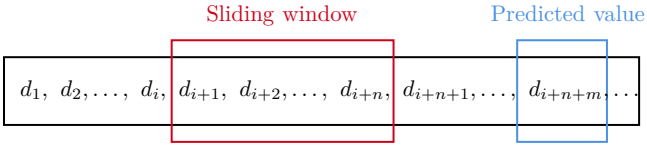

Fig. 1: Sliding window and predicted value.

B. Architectures

In our approach, we have combined LSTM and FFT (Fast Fourier Transform) in different ways. In what follows, we will present the DFT and the FFT and the different ways they have been used in our architectures.

1) DFT and FFT: The Discrete Fourier Transform (DFT) is a function allowing to perform a Fourier analysis on digitized signals. The DFT of a signal $[x_0,\ x_1,\ \ldots,\ x_{N-1}]$ of length N, is another signal $[y_0,\ y_1,\ \ldots,\ y_{N-1}]$ defined by

$$y_k = \sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi}{N}kn\right) - i\sin\left(\frac{2\pi}{N}kn\right)\right] \tag{2}$$

where $k \in \{0, 1, ..., N-1\}$ and i is the imaginary unit.

The DFT of a vector is calculated by means of the Fast Fourier Transform (FFT) [6], which is a fast algorithm with complexity $O(n \log n)$ for a vector of size n. The DFT is used in several domains, such as digital treatment of signals, image compression, voice recognition, etc. As for the classic Fourier transform, it converts signals from the time domain into the frequency domain. Note that the transformation to the frequency domain with DFT causes the loss of temporal information, but prediction needs this temporal dimension. We can solve this problem by using a sliding window. Indeed, the DFT provides a frequency dimension and the sliding window allows keeping the time dimension.

2) Architectures: We have produced four different architectures in our studies: LSTM (method 1), RealFFT-LSTM (method 2), CombinedFFT-LSTM (method 3) and HSFFT-LSTM (method 4), with two versions each, a simple LSTM layer and a double-layer LSTM, see Fig. 2. In the first one, the sliding windows are passed directly to an LSTM neural network to predict the future value. The sliding window, in the second architecture, undergoes a DFT preprocessing, which returns a vector of complex numbers, and then their real parts are injected into the LSTM input. In the third architecture, real and imaginary parts of the DFT vector are concatenated before being injected to the LSTM input. In the fourth proposed architecture, we have combined the HSFFT [13][1], which is explained below, and an LSTM.

The HSFFT module consists in duplicating the signal in order to make it symmetrical. Thus, we have to consider that the sliding window represents only half of the input signal and we duplicate it in order to obtain a symmetrical signal as shown in Fig. 3. This balanced signal is processed by the FFT algorithm, which provides a true symmetrical frequency output vector having a zero imaginary part. Then, only a part of this frequency vector is transmitted to the LSTM (duplicated information is removed).

The choice of HSFFT over FFT was made after conducting some comparative tests between the two methods. The results showed that HSFFT reduces the error rate, especially when combining two LSTMs.

IV. Performance evaluation

A. Simulations' settings

In the following, we compare the different versions of the LSTM method that we have previously introduced. We considered the following methods: the LSTM, the RealFFT-LSTM, the CombinedFFT-LSTM and the HSFFT-LSTM. They are composed of one or two LSTM layers with the same number of neurons. A fifth module has been devoted to the ARIMA method. All modules were developed in Python using the Keras library [5] with a Tensorflow backend [17].

[1]HSFFT stands for "Hermitian Symmetrical signal FFT".

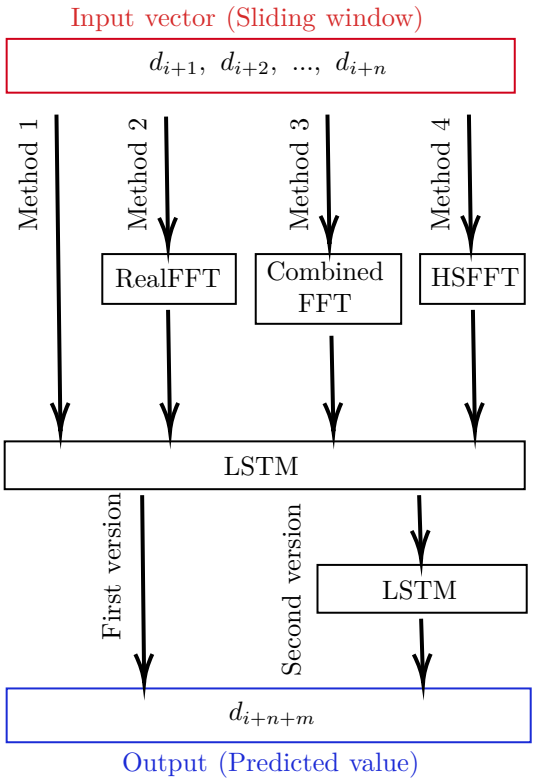

Fig. 2: Proposed architectures.

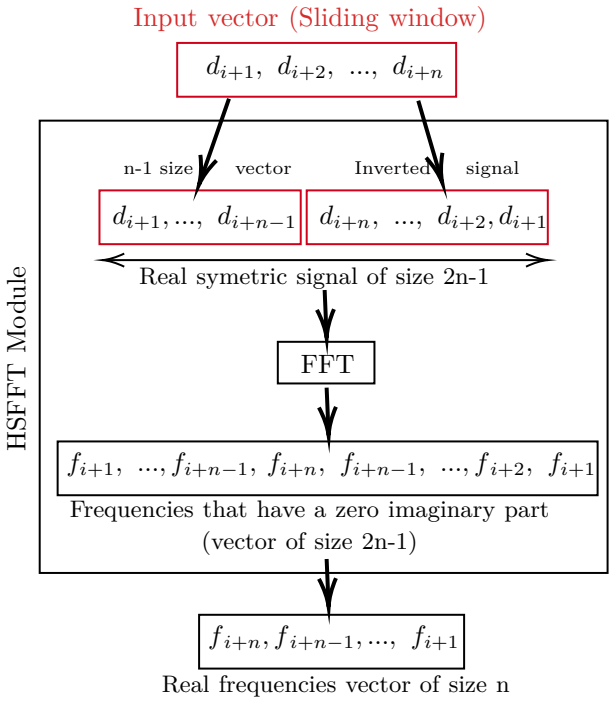

Fig. 3: Detail of the HSFFT module.

We performed the tests on real network traffic data coming from a mobile network operator. The benchmark contains 8928 values of bandwidth measurement at 10-

minute intervals, see Fig. 4. Data was first normalized to be compliant with the LSTM input. We have considered a sliding window with a size varying from 20 to 70 values (Fig. 1).

Data was divided into two parts: 70% was used for the learning phase and 30% for the test phase. For the training, we set the batch size to 1 and to learn, we use an Adam optimizer with a learning rate of 0.01. To evaluate the effectiveness of each module, we considered the calculation of the MAPE (Mean Absolute Percentage Error) between the real values and the predicted ones.

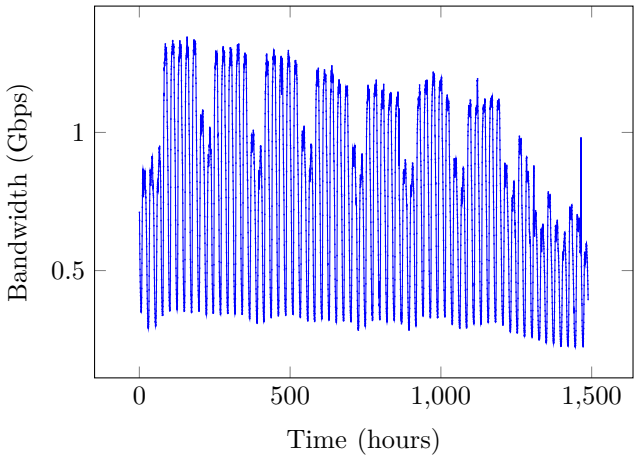

Fig. 4: Network data traffic.

The comparative study was conducted in two phases:

1) Phase 1: It consisted of comparing the HSFFT-LSTM method with similar ones, using a deep learning approach (LSTM, CombinedFFT-LSTM and RealFFT-LSTM).
2) Phase 2: Here, we compared our approach to the ARIMA method.

The obtained results of each module are treated, analyzed, and represented by box plots. We performed 72 series of tests in which every test was executed 20 times (for the calculation of confidence intervals). The tests have been carried out by changing the most influencing parameters one at a time in order to determine the error rate for different prediction horizons, and using different values for the size of the sliding window.

We have taken into consideration the effectiveness of each module based on the following criteria:

- accuracy over the prediction horizon (by varying the number of neurons within the LSTM and the size of the sliding window);
- accuracy as a function of the size of the sliding window;
- accuracy as a function of the number of LSTM neurons;
- prediction latency.

B. Results

1) Accuracy over the prediction horizon: We performed a first test comparing the LSTM, RealFFT-LSTM, CombinedFFT-LSTM and HSFFT-LSTM methods according to the prediction horizon, for an LSTM of 32 neurons, a sliding window of size 30 and a prediction horizon of 1 hour, 2 hours and 3 hours. We used the MAPE error, plotted in Fig. 5.

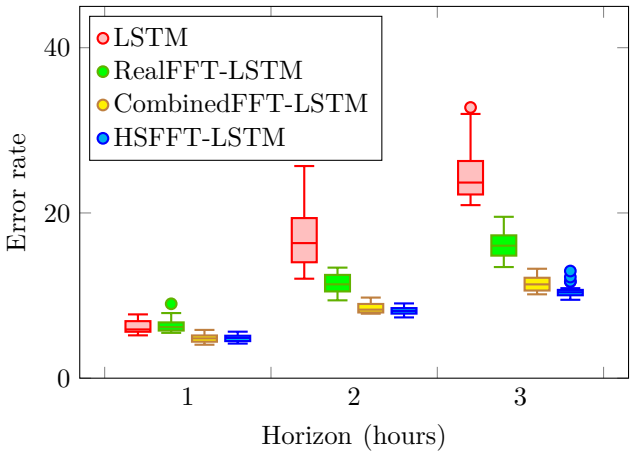

Fig. 5: Comparison between LSTM-based methods according to different prediction horizons (sliding window size = 30, neurons number = 32).

The obtained results show that increasing the prediction horizon has an almost linear impact on the prediction error. The increase of this error with the prediction horizon depends, however, on the strategy applied. Indeed, the larger the prediction horizon, the larger the difference between the considered strategies. Besides, one can notice that the methods using filters (i.e., RealFFT-LSTM, CombinedFFT-LSTM and HSFFT-LSTM) clearly outperform the LSTM alone, when increasing the distance with the forecasted value. For a prediction with a distance of 1h (6th value in the future), the different approaches show, nevertheless, results with the same order of magnitude. The two methods CombinedFFT-LSTM and HSFFT-LSTM exhibit the best performance with a slight advantage to the second one.

2) Accuracy as a function of the size of the sliding window: The first tests led us to conduct another series of experiments to analyze the behavior of the LSTM and HSFFT-LSTM methods when we changed the size of the sliding window. We conducted 6 series of 20 tests each, for different values of the sliding window, whose results are shown in Fig. 6. We also performed these experiments with two combined LSTM layers (see Fig. 7).

When using only one LSTM layer, we notice that for a rather small window size, HSFFT-LSTM shows a better accuracy compared to the LSTM alone. However, when the sliding window exceeds a certain size both methods give roughly the same performance with a slight advantage to

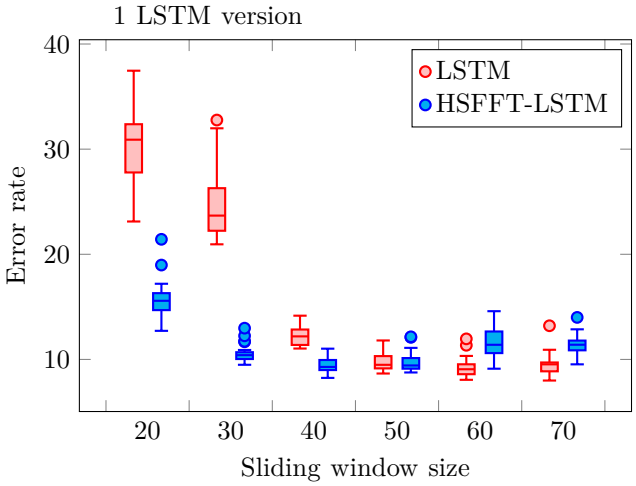

Fig. 6: Comparison according to different sliding window size (horizon = 3 hours) when using 1 LSTM layer.

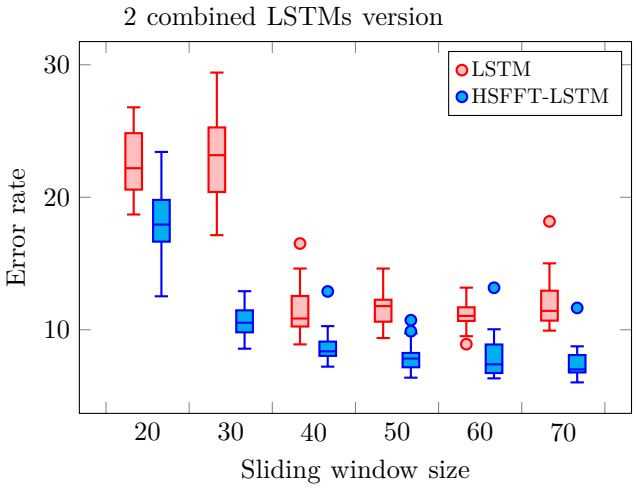

Fig. 7: Comparison according to different sliding window size (horizon = 3 hours) when using 2 LSTM layers.

the LSTM. For the two combined LSTM version, HSFFT-LSTM show better performance and improved results compared to a single LSTM.

3) Accuracy as a function of the LSTM neurons' number: We analyze, here, a series of tests designed to examine the impact of the number of neurons within the LSTM network on the error rate, over multiple measurement horizons. The comparison is made only between the two methods LSTM and HSFFT-LSTM. We chose an LSTM with 32, 64, 96 and finally 128 neurons, horizons of one hour, two hours and three hours, and each time we made 20 tests (see Fig. 8).

The results of this series led us to the conclusion that HSFFT-LSTM is better than an LSTM alone, regardless of

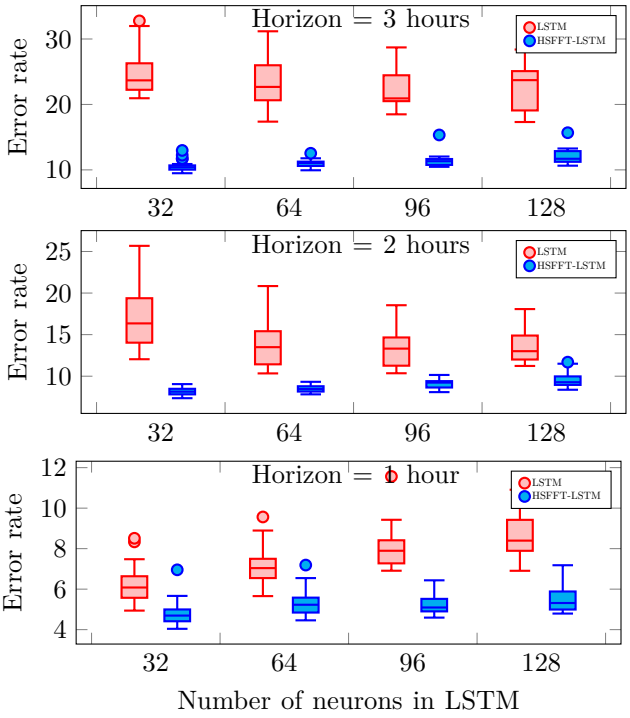

Fig. 8: Comparison between LSTM and HSFFT-LSTM methods according to the number of neurons and different prediction horizons (sliding window = 30).

the number of neurons. Moreover, with only a few neurons, HSFFT-LSTM still makes excellent predictions, unlike the LSTM, which deteriorates significantly for horizons far in the future. The HSFFT-LSTM procedure remains powerful for a fairly large prediction horizon, and even with a reduced number of neurons, which allows saving system resources.

4) HSFFT-LSTM compared to ARIMA with different sliding windows: We compare, here, our approach with a classical statistical method, which led us to perform another series of prediction tests using the ARIMA and the HSFFT-LSTM modules. Our sliding window size was 20, 30 and 40, using prediction horizons of one hour, two hours and three hours.

The results of these tests conducted us to deduce that our approach is especially good for large prediction horizons and large sliding windows (see Fig. 9).

5) Prediction latency: To compare the execution time of the ARIMA, LSTM and HSFFT-LSTM techniques, we conducted some experiments with a sliding window of size 10, and we executed ARIMA using data of size 288 to ensure its convergence. We divided the ARIMA execution time by 30 to make a fair comparison with 10-sized sliding window HSFFT-LSTM.

We found that LSTM and HSFFT-LSTM are significantly better in terms of execution time compared to

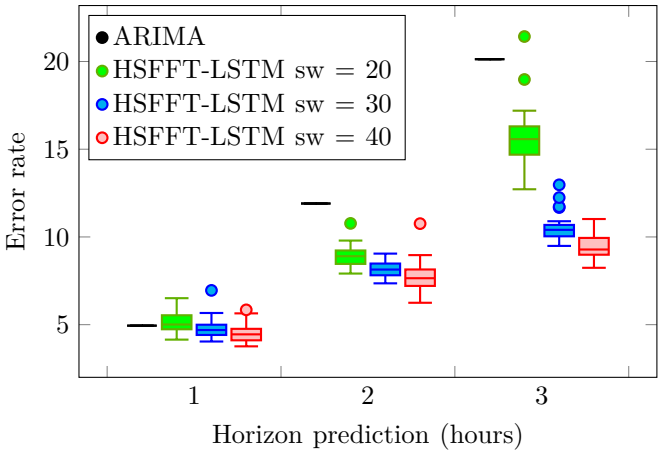

Fig. 9: Comparison between ARIMA and HSFFT-LSTM methods.

.

the ARIMA method. However, LSTM alone and HSFFT-LSTM are similar in terms of execution time, which means that the additional processing of the filter adds almost nothing in terms of resource consumption on this dataset (Fig. 10).

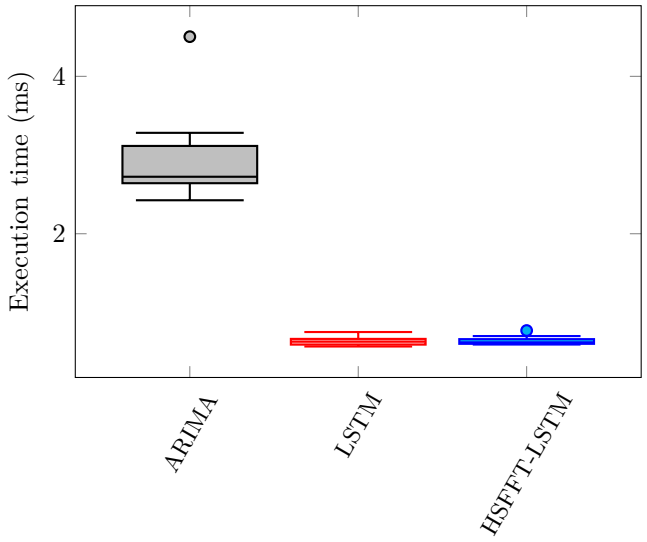

Fig. 10: Comparison between execution time of ARIMA, LSTM and HSFFT-LSTM methods.

In conclusion, we have seen that the inclusion of the FFT allows a significant improvement in the prediction of values over various measurement horizons. Even if it is difficult to rigorously justify the underlying reasons, we believe that the shift from the time domain to the frequency domain, with amplitudes reflecting the importance of the frequencies, allows the LSTM to better grasp the parameters impacting the prediction the most and to separate them more easily from potential noises that are not very visible on the time scale.

V. Conclusion

In this work we propose the use of data preprocessing before injecting them into an LSTM neural network for time series prediction. Our approach is based on the use of an FFT filter applied to a sliding window of data.

We combined LSTM with FFT in several ways and performed a series of comparative tests, exploring the procedures' performances by changing several parameters. Our study led us to conclude that FFT-LSTM is more effective in terms of prediction horizon and execution time than the classical ARIMA. Compared to similar methods such as LSTM alone, our analysis showed that for a large dataset and with a reduced number of neurons, our approach gives good results even for a large prediction horizon. In other words, the combination that we propose of methods not only reduces significantly the prediction error, but can also reduce the number of required neurons, thus accelerating the learning process. In addition, our approach maintains good performance over long prediction horizons.

In the future, we intend to use this technique in the management of an SDN/NFV network, notably through the dynamic scaling of services. We also plan to improve its rapid adaptation to potential variations in traffic patterns.

References

[1] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin. Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach. IEEE Network, 32(6):42–49, November 2018.

[2] Davide Andreoletti, Sebastian Troia, Francesco Musumeci, Silvia Giordano, Guido Maier, and Massimo Tornatore. Network traffic prediction based on diffusion convolutional recurrent neural networks. In IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 246–251. IEEE, 2019.

[3] William L. Briggs and Van Emden Henson. The DFT: An Owners' Manual for the Discrete Fourier Transform. Society for Industrial and Applied Mathematics, 1987.

[4] Fernando Camacho, César Cárdenas, and David Muñoz. Emerging technologies and research challenges for intelligent transportation systems: 5G, HetNets, and SDN. International Journal on Interactive Design and Manufacturing (IJIDeM), 12(1):327–335, Feb 2018.

[5] François Chollet et al. Keras. https://keras.io, 2015.

[6] James W. Cooley and John W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. Mathematics of Computation, 19(90):297–301, 1965.

[7] L. Dai, W. Yang, S. Gao, Y. Xia, M. Zhu, and Z. Ji. EMD-Based Multi-Model Prediction for Network Traffic in Software-Defined Networks. In 2014 IEEE 11th International Conference on Mobile Ad Hoc and Sensor Systems, pages 539–544, Oct 2014.

[8] Jan G. [De Gooijer] and Rob J. Hyndman. 25 years of time series forecasting. International Journal of Forecasting, 22(3):443 – 473, 2006.

[9] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network Function Virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine, 53(2):90–97, Feb 2015.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 1997.

[11] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time

series analysis. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 454(1971):903–995, 1998.

[12] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. International Journal of Forecasting, 36(1):54–74, 2020.

[13] Mark H. Richardson. Fundamentals of the Discrete Fourier Transform. Sound & Vibration magazine, 12(3):44–46, 1978.

[14] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling. In INTERSPEECH, pages 338–342, 2014.

[15] S. Siami-Namini, N. Tavakoli, and A. Siami Namin. A comparison of arima and lstm in forecasting time series. In 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pages 1394–1401, 2018.

[16] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparative analysis of forecasting financial time series using arima, lstm, and bilstm, 2019.

[17] Tensorflow. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org (Accessed on 2021.05.02).

[18] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang. Toward a software-based network: integrating Software Defined Networking and Network Function Virtualization. IEEE Network, 29(3):36–41, May 2015.

[19] Zhaohua Wu and Norden E. Huang. Ensemble empirical mode decomposition: A noise-assisted data analysis method. Advances in Adaptive Data Analysis, 01(01):1–41, 2009.

[20] Wucherl Yoo and Alex Sim. Time-series forecast modeling on high-bandwidth network measurements. Journal of Grid Computing, 14(3):463–476, Sep 2016.