Attention Networks for Time Series Regression and Application to Congestion Control

Victor Perrier, Emmanuel Lochin, Jean-Yves Tourneret, Patrick Gélard

Attention Networks for Time Series Regression and Application to Congestion Control

Victor Perrier
ISAE-SUPAERO, TéSA
Toulouse, FRANCE
victor.perrier@tesa.prd.fr

Emmanuel Lochin
ENAC
Toulouse, FRANCE
emmanuel.lochin@enac.fr

Jean-Yves Tourneret
Université de Toulouse, INP-ENSEEIHT/IRIT/TéSA
Toulouse, FRANCE
jean-yves.tourneret@inp-toulouse.fr

Patrick Gélard
CNES
Toulouse, FRANCE
patrick.gelard@cnes.fr

***Abstract*—This paper studies a new attention-based recurrent architecture, lighter and less computationally expensive than a global attention network. This type of architecture achieves better performance than commonly used recurrent networks for time series regression. An application to congestion control is considered, where the history of round trip times (RTT) evolution history is used to monitor congestion control. The performance of the proposed new congestion control strategy is evaluated with both synthetic and real traces, showing that it can be efficiently used to estimate the congestion state of a network.**

***Index Terms*—Attention Networks, Congestion Control, Time Series Regression, TCP, COPA**

I. Introduction

TCP congestion control (CC) is an essential mechanism for the transport of data and the fair sharing of Internet resources. The main goal of TCP is to prevent network congestion, as the latter might lead either to packet losses (due to router queue overflow) or to an increase of the end-to-end delay. As a matter of fact, the increase of the two metrics "latency" and "losses" results in a decrease of the data transfer time. Basically, the TCP CC algorithm is therefore an algorithm that schedules the transmission time of a packet, as a function of the congestion level of the network. The congestion level is determined thanks to the return path used by TCP acknowledgments, which allows monitoring using different metrics such as the evolution of the round trip time (RTT), the number of lost packets, the jitter (variance of the inter-packet spacing), etc. All these metrics, or a subset, are then crunched by the CC algorithm to compute the transmission time of a packet.

Network arrivals are often modeled as Poisson processes by simplicity. Although several proposals, modeling and predicting the Internet traffic remains a complex problem. This might explain why these models [1], [2] have never been practically used to improve the behavior of TCP CC. Nevertheless, to obtain better CC algorithms, the networking community has considered machine learning tools, where promising solutions have been proposed [3], [4]. REMY CC was a pioneer in this domain [3]. The main drawback is that a consistent CC algorithm can require more than 24 hours for offline learning, and is only valid for a given architecture [5]. Following this first attempt, several other proposals have emerged. Among others, COPA is a CC for improved video performance deployed by Facebook[1]. COPA [6] attempts to estimate the internal state of the network to send packets optimally. The goal is then to estimate the network congestion state from time series describing the evolution of key network parameters (queue size, bottleneck load evolution, ...). COPA is one of the most performant CC to carry video contents compared to BBR or CUBIC [7], explaining our specific interest for this CC mechanism. However, the contribution of this paper can be applied to any CC based on time series to perform (e.g., using RTT or loss histories).

The main question addressed in this paper is: how to deal with time-series available for CC to efficiently estimate the evolution of the network state and, particularly, the level of congestion? This problem motivates this work, which aims to study a new neural network (NN) architecture based on Attention for multivariate time series regression, with an application to network congestion prediction. We introduce a new model that is lighter and performs as well as global Attention. This work reduces the number of parameters to be estimated, resulting in a better computational cost when compared to Attention. More precisely, instead of having a complexity in $\mathcal{O}(L^2)$ at each new time step, the new architecture has a complexity in $\mathcal{O}(L)$, L being the size of the time series. Another contribution of the proposed method is to show the interest of the Attention networks for the regression of time series, as they allow estimating more precisely some functions such as the maximum function and to better consider the past of the time series that has to be predicted.

A previous contribution investigated whether Deep Learning (DL) algorithms could improve the CC estimator used in COPA [8]. In this study, we particularly focus on the analysis of the computational efficiency and the performance of Attention. Section II recalls the CC objective, and how COPA, among others, attempts to reach this objective by using a simple estimator. Section III introduces DL algorithms that could be used to reach the same objective with their pros and cons. The proposed DL architecture is presented in Section IV with a performance evaluation in Section V using synthetic and real traces. Section VI concludes this work.

[1] https://engineering.fb.com/2019/11/17/video-engineering/copa/

II. The congestion control objective

As previously discussed, the main objective of a CC algorithm is to reach the full capacity available in the network while minimizing the queue load. This optimal regime is schematized by the optimal point in Figure 1.

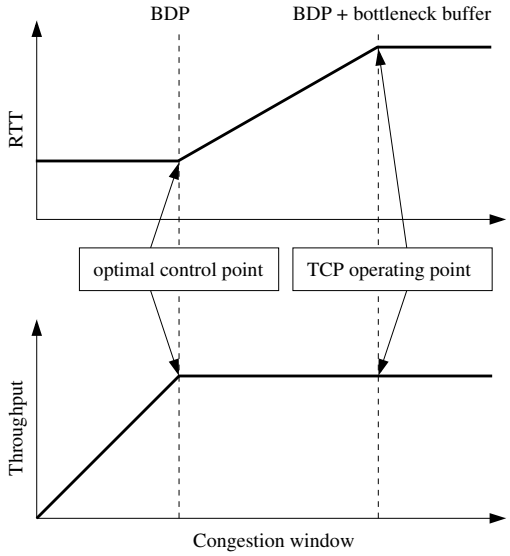


Fig. 1: Optimal congestion control point.

To better assess what is this optimal control point, consider a simple model where a fixed-sized bottleneck queue with a service μ bit/s, a max queue size q_m, and a current queue size or load q_l, is crossed by a single flow at λ bit/s. This flow might encounter three states:

1) if $\mu > \lambda$, the queue is always empty and packets are passed without delay;
2) if $\mu < \lambda$ and $q_l < q_m$, packets are stored and the end-to-end delay increases;
3) finally if $\mu < \lambda$ and $q_l = q_m$, arriving packets are dropped and severe congestion occurs.

The delay and delivery characteristics considering this bottleneck link are illustrated in Fig. 1. The problem is thus to operate as close as possible to this optimum operating point. This problem has been solved by COPA [6] using an estimator defined by:

$$\min_{i\in[t-L_1,t]} \boldsymbol{x}_i - \min_{i\in[t-L_2,t]} \boldsymbol{x}_i, \tag{1}$$

with $L_1 << L_2$ and where $\boldsymbol{x}_i$ is the univariate time series of RTT, i.e., the round trip times of packets in the network.

COPA uses delay signals to detect congestion and sense the network load with an oscillation of the capacity, as illustrated in Fig. 2. In the congestion avoidance phase, COPA increases and decreases the throughput periodically to probe the network congestion level. This estimator has shown good performance compared to currently used techniques such as TCP CUBIC. However, it relies on the hypothesis that COPA is the sole CC algorithm being used on that link. Indeed, in this case, the flows' throughput is synchronized, allowing the queue to evolve as shown in Fig. 2, and thus the use of the estimator

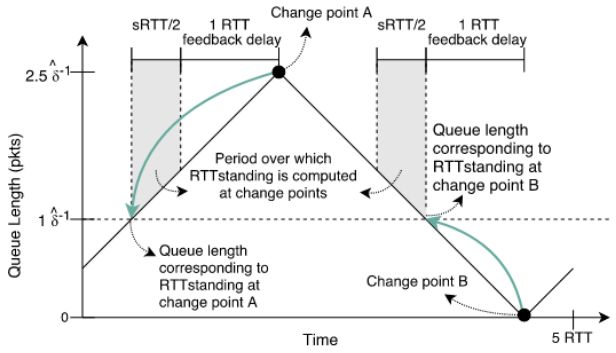


Fig. 2: COPA mechanism described in [6]. The aim is to stabilise the queue length around δ^{-1}.

(1) because the queues will be emptied periodically. However, if another CC is used, that estimator may not be as accurate.

The objective of this paper is to propose a supervised DL method that can improve (1). More precisely, we want to obtain an accurate estimation, even if there is a competing CC algorithm on the same path. COPA already handles competition by being more aggressive, but we think that having a correct estimation of the state of the network might improve performance.

III. Existing Deep Learning architectures for time-series regression

Standard time series regression models can be grouped into several families. The first family gathers prediction algorithms based on parametric methods using for example Kalman filters or ARIMA (AutoRegressive Integrated Moving Average) models and its improvements. Non-parametric methods also exist, such as those based on support vector regression (SVR) [9] or on k-Nearest Neighbors (kNN). A second family of regression algorithms is based on neural networks, in particular DL methods. To be able to process time series, Recurrent Neural Networks (RNN) were first developed [10]. To overcome the problem of gradient disappearance during training [11], other methods have been introduced to improve the principle of RNNs. These methods are based on more original architectures such as GRU (Gated Recurrent Units) [12], LSTM (Long Short-Term Memory) [13] and CNN (Convolution Neural Networks) [14]. More recently, a new architecture called Attention has shown its interest in several applications [15]. First used in natural language processing tasks, this architecture makes it possible to obtain more precise prediction models, considering that there is sufficient computing power. For example, we can cite the work of [16] which shows that Attention is sufficient on its own to solve translation or text interpretation tasks.

The problem studied in this paper is to build a neural network architecture to estimate useful metrics for CC such as (1). To perform the supervised learning, we propose to train different models (that will be explained later) with the

following cost function

$$\frac{1}{L}\sum_{i=1}^{L}\|f(\boldsymbol{X}_{1:i})-\boldsymbol{y}_i\|^2, \quad (2)$$

where $\boldsymbol{y}_i$ is a vector which contains the metrics we want to estimate at time i, $\boldsymbol{X}_{1:i} \in \mathbb{R}^{i\times d}$ is a matrix containing the observations up to time i (in (1) this matrix is composed of the RTT, but other complementary observations could be considered as well), L is length of the time series and f is the model defined by the NN to be trained.

A. LSTM networks

This section describes LSTM networks since they are commonly used for time series, and introduces the rationale of this study. We noticed that some regression tasks, such as the estimator presented in (1), cannot be accurately predicted with LSTM networks. Indeed, the predictions obtained with these networks can correctly estimate the maxima of a time series, but they fail to memorize this information for the next time steps. This observation will be further discussed in Section V-A (see Figure 5b)[2]. This absence of memory for LSTM networks has motivated the present study, which aims at building a model able to efficiently estimate functions such as the minimum and maximum of a sliding window from one or several time series. In the rest of this paper, we show that networks using Attention can estimate these maxima and minima with a better accuracy.

B. Attention in a nutshell

Before explaining how vanilla Attention networks perform, the following notations are introduced:

Notations	
Symbol	Signification
d	dimension of the Time Series
L	Length of the Time Series
M	Number of layers
$\mathbf{X} \in \mathbb{R}^{L\times d}$	Multivariate Time Series
$\mathbf{W}_k \in \mathbb{R}^{d\times d}$	
$\mathbf{W}_q \in \mathbb{R}^{d\times d}$	Matrices of Attention parameters
$\mathbf{W}_v \in \mathbb{R}^{d\times d}$	
$\mathbf{P} \in \mathbb{R}^{L\times d}$	Position encoding matrix concatenated with $\mathbf{X}$

Attention is a time series processing mechanism originally built to perform language translation tasks. However, it can easily be extended to other domains, such as time series regression. To illustrate how Attention works, we use the running example displayed in Fig. 3. This example shows a sentence where the relationships between each word are represented either by plain or dashed lines, depending whether there is a strong or weak relationship between two words. Actually, these relationships are defined thanks to a weight matrix. This weight matrix (where the sum of each row is 1) allows assessing the relationships between elements of the time series.

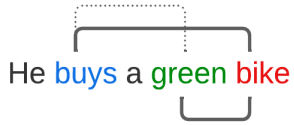


Fig. 3: Example of the attention mechanism for a sentence. A dashed line corresponds to a weak connection between words, whereas a plain line is used for a strong connection.

The Attention mechanism itself is not sensitive to the relative position of the elements of the time series $\boldsymbol{X}_i \in \mathbb{R}^d, i = 1, ..., L$. To solve this issue, it uses a position matrix $\boldsymbol{P} = (p_{i,j})$ with $i, j \in \{1, ..., L\} \times \{1, ..., d\}$ defined as [16]:

$$p_{i,j} = \begin{cases} \sin\left(\frac{i}{100^{2j/d}}\right) & \text{si } j = 2n, n \in \mathbb{N} \\ \cos\left(\frac{i}{100^{2j/d}}\right) & \text{si } j = 2n+1, n \in \mathbb{N} \end{cases}$$

The position matrix allows defining the position of a vector in a time series, like a clock can define a time instant with three hands for seconds, minutes, and hours. Note that the use of sine and cosine functions for $p_{i,j}$ ensures positions $p_{i,j}$ are in $]-1, 1[$. Attention layer can then be defined as follows:

$$\text{ATTENTION}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V}, \quad (3)$$

with

$$\text{softmax}(\boldsymbol{X})_{i,j} = \frac{e^{\boldsymbol{X}_{i,j}}}{\sum_{k=0}^{d} e^{\boldsymbol{X}_{i,k}}},$$

and

$$\begin{cases} \boldsymbol{K} = \mathbf{W}_k\boldsymbol{X}, \\ \boldsymbol{Q} = \mathbf{W}_q\boldsymbol{X}, \\ \boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{X}, \end{cases}$$

where $\mathbf{W}_k$, $\mathbf{W}_q$ and $\mathbf{W}_v$ are parameter matrices that are determined during the training phase. This example illustrates a particular case of Attention known as self-Attention (where $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ are functions of $\boldsymbol{X}$). Note that the result of (3) is a linear combination of the elements of the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_t)^T$.

The roles of the matrices $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$ are inspired by the SQL language. They represent queries (q), keys (k) and values (v). Note also that the duo of matrices $\mathbf{W}_q$ and $\mathbf{W}_k$ allows the model to determine which elements of the past are useful for the regression task. For example, if self-Attention is applied to a matrix $\boldsymbol{X}$ of univariate RTT (i.e., $d = 1$) with $\mathbf{W}_q = \mathbf{W}_k = 1$, the result of the softmax operation gives a matrix of $\mathbb{R}^{L\times L}$ with rows close to 0, except the row corresponding to the index of the maximum of $\boldsymbol{X}$ whose elements are close to 1. Thus, by choosing $\mathbf{W}_v = 1$, the result of the Attention is a matrix whose elements are approximations of the maximum of $\boldsymbol{X}$. If we are interested by the minimum of $\boldsymbol{X}$ (instead of the maximum), one can choose $\mathbf{W}_k = -1$.

[2]Note that due to their nature, the same observation holds for all RNN and CNN networks.

It is also possible to define more complicated queries, such as finding when the maximum of RTTs occurred (considering $\boldsymbol{X}$ also contains the time information with $d = 2$). In that case, we need to choose $\mathbf{W}_q = \mathbf{W}_k$ as the projection of $\boldsymbol{X}$ on the RTTs axis, and $\mathbf{W}_v$ as the projection of $\boldsymbol{X}$ on the time axis.

Despite their remarkable performance, Attention networks are difficult to use because of their massive size (the GPT-3 model created by OpenAI for text interpretation has 175 billion parameters), and long training times. Indeed, to apply an Attention model to a time series, it must be applied at each time step. Thus, at the tth step (with $t \in \{1, ..., L\}$), the computation complexity is linked to the matrix product. To treat a time series of length L, the complexity is therefore in the order of $\mathcal{O}(L^3)$ (the computation done at step $t-1$ cannot be used to ease the task because of the presence of the non-linear layers), whereas methods such as LSTMs have a complexity in the order of $\mathcal{O}(t)$. This computational time is a motivation to find a new NN architecture that is as efficient as Attention but faster, which is precisely the goal of this paper.

IV. Proposed architecture

To overcome both shortcomings of LSTM and Attention networks, we propose a new hybrid architecture defined as follows:

1) the observation matrix $\boldsymbol{X}$ is concatenated with the position matrix $\boldsymbol{P}$ yielding

$$\mathbf{X}_p = [\mathbf{X}, \mathbf{P}],$$

2) an LSTM layer is constructed:

$$\mathbf{H_i} = \mathbf{LSTM}(\mathbf{x}_1, ...\mathbf{x}_i) \in \mathbb{R}^{J \times d},$$

where J is the size of the vector produced by the LSTM network (with one layer and a unidirectional network), to be chosen by the user.

3) an Attention network is constructed as follows:

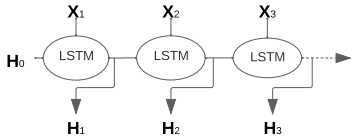
$$\mathbf{Y}_i = \mathbf{ATTENTION}(\mathbf{W}_q\mathbf{H}_i, \mathbf{W}_k\mathbf{X}_{1;i}, \mathbf{W}_v\mathbf{X}_{1;i}),$$

where the matrix $\boldsymbol{H}_i$ is used to determine which are the most important past elements. The idea of this architecture is not to use self-Attention directly (since if is too computationally intensive), but to generate, thanks to an LSTM network, a vector generating the requests (J is thus the number of requests).

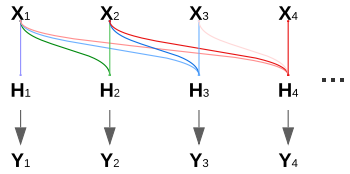
4) A non-linear layer (RELU activation function) is finally introduced as in many DL architectures:

$$\mathbf{Z} = \mathbf{FeedForward}(\mathbf{Y})$$

The previous steps 2), 3) and 4) can be repeated for each of the M layers of the network. The interest of this architecture when compared to LSTM and Attention will be shown the next section.



(a) Creation of the vectors $\mathbf{H}_i$ (first step).



(b) Attention used to know which $\mathbf{X}_i$ is used to get $\mathbf{Y}_i$.

Fig. 4: Two steps of the proposed NN architecture.

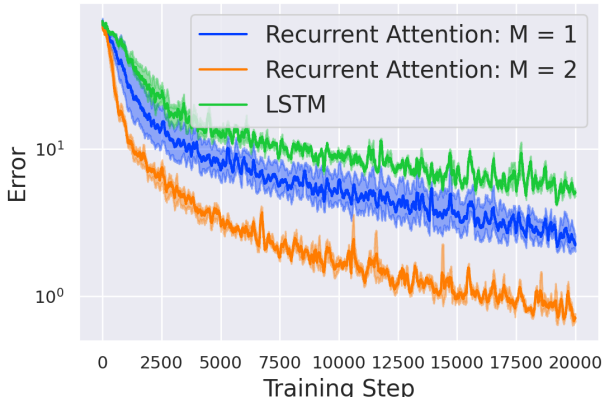
V. Results

This section evaluates the performance and ability of the proposed NN architecture to seek information from the past of time series within two use-cases: the first experiment considers synthetic data with an available ground truth whereas the second experiments are conducted using real data from the evolution of an IP router queue load.

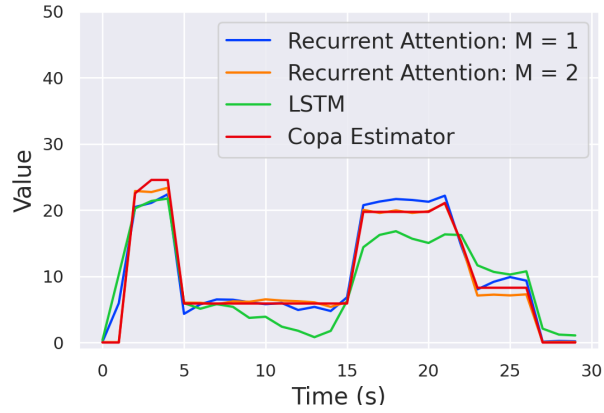
A. Finding a minimum

As explained in Section III-A, a simple estimator of the queue load is of the form (1). This section studies the capacities of the proposed NN architecture to approximate this estimator. The parameters of the NN were chosen by cross validation leading to $L_1 = 5$ and $L_2 = 30$. The NN was trained using a learning step of 0.001 and the optimizer ADAM. The RTT time series were randomly generated at each training step according to independent samples from a normal distribution ($\mathcal{N}(\mu, \sigma^2)$ with $\mu, \sigma \sim \mathcal{U}(0, 10)$ fixed for each time series) to prevent over-fitting.

Figure 5a shows that the new NN architecture, in addition to learn faster how to estimate the function f defined in (1), can learn to estimate the difference between two minima with more accuracy than an LSTM network, which reaches a learning plateau. Of course, this remains an artificial task and the proposed network was created to solve that kind of task. This first experiment also shows that the deeper the network (i.e., the larger M), the faster the elements from the past can be learnt. The poor performance of the LSTM network can be easily explained by the form of f, which is a simple relationship between elements from the past of the time series. Note that LSTM have problems to learn how to store elements in its hidden vector, and to memorize all values (for example, if the value of RTT is increasing, each value will be at some point a minimum of the sliding window). Conversely, the proposed Attention network directly refers to elements from the past, which can be accessed in one step.



(a) Training loss. The colored envelopes represent the maximal and minimal errors for 10 trained models.



(b) Estimations provided by LSTM and Attention networks for (1).

Fig. 5: Results for the synthetic task of estimating (1).

Figure 5b shows that LSTM networks cannot store the relevant information during a correct amount of time. Actually, LSTM regression seems to approximate the time series by a piece-wise linear function to minimize the mean prediction error. The LSTM network struggles to use specific past information contained in the time window of interest. Even if this information were available, the hidden vector used by the LSTM network would have difficulty storing all the relevant information contained in the time window of interest. Conversely, the Attention mechanism successfully uses the values from the past of the time series, yielding better estimates.

B. Application to network queuing estimation

This part considers a real application defined as the prediction of the queue level at a bottleneck for a given path of a network. For this purpose, we propose to use the time series of RTTs as well as the time of transmission of these packets. The training is done with a learning step of 0.0005 and the ADAM optimizer. The data was collected using the Mininet emulator and a parking lot scenario depicted in Fig. 6 as explained in [17].

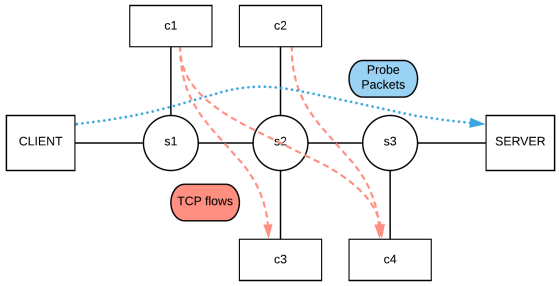


Fig. 6: Parking lot topology used for the tests. Probe packets are used to build the time series $\boldsymbol{X}$ and are sent with TCP on the path of interest.

Each link has a capacity of 50 Mbps and a nominal delay of 10 ms. Each queue enables FQ-CoDel by default with a size of 500 pkts. Packets are sent from the client to the server. TCP flows are sent between c1, c2, c3 and c4 to maintain a constant network load. Several studies have shown that short-lived flows, mainly generated by Web data transfers caused by user interactions, dominate the Internet traffic [18]. Thus, the TCP traffic is generated in such a way that the length of the TCP flows respects the Pareto principle (80% of short flows and 20% of long flows) over the long run. The objective of this first experimentation is to probe the network congestion level, i.e., the load of the queues at nodes? s1, s2 and s3. The probe flow follows the blue path outlined in Figure 6 and is a TCP flow containing real data. Each TCP flow following the red path has been generated as follows:

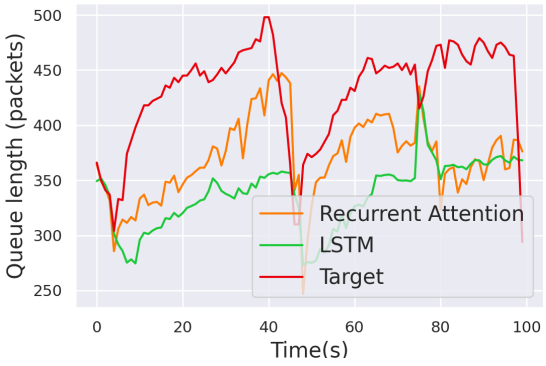
- the duration of each TCP connection has a Poisson distribution with parameter $\lambda_p = 1$;
- the time between two starting flows has an exponential distribution with parameter $\lambda_e = 10$;
- the server-client pair is randomly selected between the pairs (c1, c3), (c1, c4) and (c2, c4);
- each TCP flow is generated with the iPerf traffic generator.

This setup enables realistic and variable network conditions:

- both queues can act as the bottleneck depending on the network load and flows;
- the number of TCP flows changes to mimic a network load;
- TCP constantly switches between the slow-start and congestion avoidance phases. This allows a diverse distribution of the variables, representing various kinds of behavior (few flows, congestion, slow-start/cruise-control phase, unbalance between buffer load...).

Figure 7a shows that a plateau in the training phase is reached by the LSTM network, while Attention allows the relationship between the data and the network load to be learnt quickly, and with greater accuracy. As (1) is a good approximation of the load in the queues, we can expect the real approximate to have a strong connection to that equation, and so may need to use elements from far in the past. These

(a) Training loss. The colored envelopes represents the maximal and minimal errors for 10 trained models.



(b) Estimation of of the bottleneck load (with emulation) using LSTM and Attention architectures.

Fig. 7: Results for a concrete task of estimating the current load at the bottleneck in a network path.

remarks explain the fast learning of the proposed Attention network. Note that the good performance of the new NN algorithm is similar to that obtained with a global Attention network. However, the proposed architecture allows a faster training. Finally,it is interesting to note that for this example, it is not useful to increase the number M of network layers to obtain a better estimation accuracy.

Note that the parameters of the NN architecture implemented for this case were determined by cross validation leading to: $M = 1$, $J = 3$, $d = 6$ and 3 heads for the Attention network. The hidden dimension of the LSTM layers was set to 18, and the dimension of the hidden layer in the Feedforward network was 36. Finally, the learning rate used during training was 0.0005.

VI. Conclusion

This paper studied a new hybrid deep learning architecture using both an Attention network and a recurrent neural network for time series regression. This architecture offers an interesting compromise between accuracy and computation time. The application to queue congestion detection provided promising results as long as time past events can be useful to get a correct guess of the current state of the network. The accuracy of the regression is thus increased by long-term dependencies. This work might improve congestion control algorithms that consider RTT, losses, etc., time-series to compute packet sending time. In future work, it would be interesting to design new machine learning algorithms having a constant computational complexity at each time step, in order to facilitate the deployment of the algorithm.

VII. Acknowledgements

The authors would like to thank ISAE-SUPAERO and CNES for their funding support.

References

[1] V. Paxson and S. Floyd, "Wide area traffic: the failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226–244, 1995.

[2] M. Zukerman, T.D. Neame, and R.G. Addie, "Internet traffic modeling and future technology implications," in *IEEE INFOCOM 2003*, 2003, vol. 1, pp. 587–596 vol.1.

[3] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 123–134, 2013.

[4] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX NSDI Conf.*, 2015, pp. 395–408.

[5] Francis Y. Yan et al., "Pantheon: the training ground for internet congestion-control research," in *USENIX ATC*, Boston, MA, July 2018, pp. 731–743.

[6] Venkat Arun and Hari Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *USENIX NSDI*, Renton, WA, Apr. 2018, pp. 329–342.

[7] Nitin Garg, "Comparing copa with cubic, bbr for live video upload," Nov. 2019, IETF'106 ICCRG, Singapor, https://datatracker.ietf.org/meeting/106/materials/slides-106-iccrg-experiments-at-facebook-with-copa-00.

[8] Victor Perrier, Emmanuel Lochin, Jean-Yves Tourneret, Nicolas Kuhn, and Patrick Gelard, "How Attention Deep Learning Can Improve Copa Congestion Control Performance," in *The International Wireless Communications and Mobile Computing Conference (IWCMC)*, Dubrovnik, Croatia, June 2022, https://hal.archives-ouvertes.fr/hal-03630784/file/main.pdf.

[9] M. Awad and R. Khanna, "Support vector regression," in *Efficient learning machines*, pp. 67–80. Springer, 2015.

[10] Larry R Medsker and LC Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[11] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al., *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, IEEE Press, 2001.

[12] J. Chung et al., "Gated feedback recurrent neural networks," in *Proc. Int. Conf. on Machine Learning*, 2015, pp. 2067–2075.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[15] A. de Santana Correia and E. L. Colombini, "Attention, please! a survey of neural attention models in deep learning," *Artificial Intelligence Review*, pp. 1–88, 2022.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Proc. Conf. Advances in neural information processing systems*, 2017, vol. 30.

[17] Victor P and al., "How attention deep learning can improve copa congestion control performance," in *IWCMC IIIoT*, 2022.

[18] D. Ciullo, M. Mellia, and M. Meo, "Two schemes to reduce latency in short lived TCP flows," *IEEE Communications Letters*, vol. 13, no. 10, Oct. 2009.