AIN'T - An Artificial Intelligent Network Thermometer for Measurements of Link Saturation on TCP/IP Flows

Marcelo R. Silva[1] and Cesar Marcondes[2]
[1]Aeronautics Institute of Technology (ITA), São José dos Campos/SP - Brasil
[2]Aeronautics Institute of Technology (ITA), São José dos Campos/SP - Brasil

***Abstract*— The transmission capacity of data links is crucial for network administrators. This measure is particularly significant in operational environments where maintaining communication continuity is vital. However, the principal strategy of the most widely used tools or protocols for this purpose consists of inserting extra packets into the network and throttling its transmission capacity. Such an active strategy has the potential, even momentarily, to produce packet losses in combat support applications (SAD, for example) and crash communications on the network under analysis. Seeking to avoid network overload while measuring its saturation, this work proposes AIN'T (Artificial Intelligent Network Thermometer). AIN'T measures the level of congestion on the data link passively without inserting any data packets into the respective infrastructure. To this end, it applies MLP, LSTM, and CNN Deep Learning Networks. The results show that the models extracted from these neural network architectures can distinguish between high and low-level link saturation in an IP data network with over 99% precision.**

***Keywords*— Passive network monitoring, Deep Learning, Transmission Control Protocol (TCP).**

I. Introduction

The broad expansion of the Internet and the services available on it intensifies the good performance of IP communication networks as an imperative. This fact can be demonstrated, for example, by the expansion of network monitoring tools. In these scenarios, bandwidth monitoring tools (BMT) play a key role. BMT helps network administrators identify and eliminate bottlenecks that lead to low speeds, ensuring the right bandwidth to carry out an organization's business.

The available throughput, directly related to bandwidth, is one of the soundest metrics for managing a network. Unfortunately, most tools used for this purpose collect this information by introducing extra data into the network, seeking its transmission limit, which can cause connections to fail in overload scenarios. This type of approach, therefore, should not be adopted during critical transactions and prohibits infrastructure monitoring during operation when control of the status of the links is crucial.

A. Motivation

In armed conflicts, network performance has high importance and complexity. Communication failures can compromise the best-planned operations. The communication continuity obligation becomes even more challenging considering link heterogeneity and routes between the various echelons. The tactical level generally employs links that allow movement in large zones but with narrow bandwidth, extremely susceptible to outages. Thus, on the battlefield, the ideal would be network monitoring with no overloading.

The MTO [1] is an emblematic expression of the complexity of network monitoring on the battlefield. In this system, sudden variations in the link characteristics are expected regarding available bandwidth (thousands of bytes to hundreds of bits), RTT (hundreds to thousands of of microseconds), and error rate (high in HF). Such a scenario, composed of narrowband technologies, requires a passive network monitor at each link, promoting a better use of each transmission medium.

To enable link monitoring without compromising its operation, this work proposes the AIN'T (Artificial Intelligent Network Thermometer) tool. AIN'T assesses the saturation level of an end-to-end link passively without introducing new packets into the network. To achieve this, AIN'T applies artificial intelligence models that, once trained, can distinguish with high accuracy (above 99%) between moments of high and low link saturation.

B. Research Questions

This work, motivated by the issues presented so far, addresses the following research questions:

- **RQ1: Binary link saturation classification, using Deep Learning Networks:** Would it be possible to train a neural network so that it is capable of accurately mapping the saturation level of a data link as high or low from the establishment of a cut line?
- **RQ2: Tolerance to variation in the number of flows:** Would the models be capable of accurately classifying the saturation level of the network, even when there is variation in the number of flows established in it?
- **RQ3: Efficiency of low-dimensional models:** Would models receiving low-dimensional vectors (2 or 3) as inputs be capable of passively distinguishing degrees of link saturation above and below a given threshold?

II. Related Work

One of the most common tools for measuring the available bandwidth on a link is iPerf. Its version 3 [2] allows users to adjust several parameters according to the measurement interest. However, iPerf is an active measurement tool and uses packet insertion in the network to obtain its capacity. In addition, this depends on the active transport protocol in the TCP stack.

M. Silva, marcelo.silva.101270@ga.ita.br; C. Marcondes, cesar.marcondes@gp.ita.br.

Another traditional tool for throughput measurement is nuttcp [3]. Recently, this tool has produced data for training neural networks aimed at throughput prediction in cellular networks [4]. However, similar to iPerf, nuttcp also uses an active approach for its measurements, which restricts its use during critical network operation states.

Another rapidly expanding tool for network monitoring is the perfSONAR [5]. This tool is currently present on all continents, including partners such as RNP. Its general objective is to peruse end-to-end paths to improve the user experience. Nevertheless, perfSONAR uses active tests [6], which can potentially overload the network infrastructure, as mentioned above.

Current proposals, such as [7] and [8], employ artificial intelligence to boost the physical layer efficiency. These works raise models from machine learning to obtain a more efficient mechanism for link adaptation in terms of modulation and bandwidth. The advantages of this type of approach, however, would depend on adaptations at the electronic circuit level, which makes its scalability difficult.

Deep learning networks demonstrate a profound synergy with problems related to link congestion. Works such as [9], [10], and [11] attest that MLP, LSTM, and CNN deep networks can classify and control congestion in a TCP/IP network. Although closely related, congestion control alone does not provide end users with a view of the degree of saturation of a given link, as proposed by this work.

This paper presents AIN'T, a tool that indicates the saturation link degree without probe packets or IP infrastructure adaptation. As will be seen in the course of this work, AIN'T will only need access to the ACK packets that arrive at a given client, too simple in the considered context.

To provide the degree of overload of a link, AIN'T uses low-dimensional inputs (2, 3) Deep Learning Networks, including LSTM and CNN. A High-performance language implements these neural networks, responding within the RTT time scale, which, to the best of our knowledge, is unprecedented in the literature.

III. Background

Understanding where losses occur in a TCP/IP network is essential to know how it works. Considering a minimally designed network in terms of available bandwidth and link reliability, losses during packet transmission occur in routers when the number of packets exceeds the capacity of their input or output buffers [12]. Therefore, this work proposes to determine the degree of link saturation by observing the percentage of buffer occupancy.

The router drop premise is the construction foundation of the models extracted from Neural Networks to construct AIN'T. Section V-B designs a Supervised Learning where the Neural Network receives data sets associated with degrees of buffer occupancy above and below a certain level to adjust the respective synaptic weights.

A. Multilayer Perceptrons

In mathematical terms, a perceptron is a function that associates each element of R^n with a Real value given by

$$F(x,w) = \varphi(\sum_{j=1}^{n} w_{ij}x_j + b_i), \quad (1)$$

where $x = (x_1, x_2, ..., x_n) \in R^n$, $w = (w_{i1}, w_{i2}, ..., w_{in}) \in R^n$, φ is called activation function and $b_i \in R$. The circles in Figure 1 give a general view of the perceptron system.

A perceptron layer consists of a set of perceptrons that share the same sequence of inputs over time. MLPs are layers of perceptrons organized in such a way that the outputs of one layer serve as inputs to the next (Figure 1). The first stimulus layer is called the input layer, and the final one is called the output layer. The others are hidden layers.

Although the composition of the perceptron layers is simple, it is very powerful. According to the universal approximation theorem [13], an MLP with only one hidden layer is capable of arbitrarily approximating any continuous function from R^n to R on the hypercube $[0,1]^n$. That is why it serves as the basis for the neural architectures presented in the following.

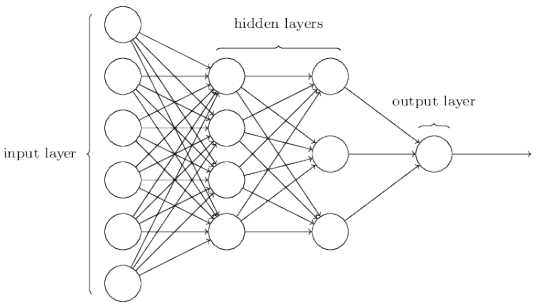

Fig. 1: The figure represents a 4-layer MLP, with two hidden layers and the respective input and output layers. The connection vectors arriving at the circles represent the inputs of the function $F(x, w)$ (Equation 1). Reprinted from [14]

B. Long Short Term Memory (LSTM) Networks

LSTM networks [15] are a special case of Recurrent Neural Networks (RNN). The inputs over time (x_1, x_2), the hidden state (h_1, h_2), and the cell state (c_1, c_2) (Figure 2) are mathematically processed to prevent gradient vanishing. The LSTM design efficiently doses the influence of more remote inputs on the final output of the network.

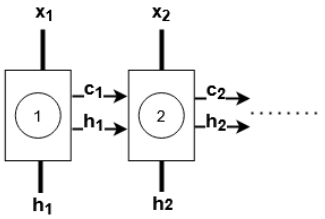

Fig. 2: Simplified LSTM example. This figure highlights the chaining components of this architecture: The hidden state (h_1 and h_2) and the cell state (c_1 and c_2). The treatment given to these outputs makes LSTMs suitable for predictions that depend on information with expressive gaps between them.

C. Convolutional Neural Networks (CNN)

A Convolutional Neural Network [16] is a Deep Neural Network that has a sequence of filters whose responses feds interconnected perceptron layers. The filters perform convolutional and pooling operations, intentionally highlighting

only remarkable features and reducing the dimension of inputs [17]. Figure 3 illustrates the operation of the CNN network adopted by this work, which decreases 3x3 dimension inputs to two-component vectors before activating a set of perceptrons.

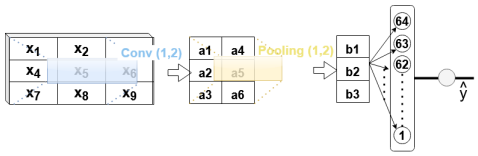

Fig. 3: CNN example. The figure highlights the convolution and pooling operations, which provide small-scale inputs to perceptron layers.

IV. Methodology

This paper adapts the five-stage methodology from [9], here composed of four stages: Data Mining, Data Processing, Model Training and Evaluation, and AIN'T Build. Each step feeds its following logical chain, culminating in a passive bandwidth monitor with higher precision integrated with the network. Figure 4 illustrates that.

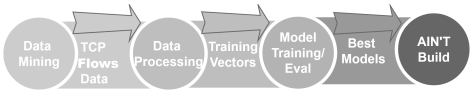

Fig. 4: The methodology applied by the research. Each phase supports the next until the chain culminates with AIN'T, classifying the degree of network saturation. Adapted from [9]

We used the ns3 [18] Traffic Analyzer (TA) [1] proposed by [9] to build training, validation, test, and generalization datasets. This work varies the TA parameters and performs several new simulations of a *dumbbell* topology (Figure 5) with a fixed number of stations (60 clients and 60 servers) and bottleneck rate (100 Mbps). All channels have a 2ms delay. Except for the bottleneck, the transmission rate is 1Gbps on all channels. The TA sets Router 01 queue to 300 packets of 1500 bytes and all MSS sockets to 1420 bytes.

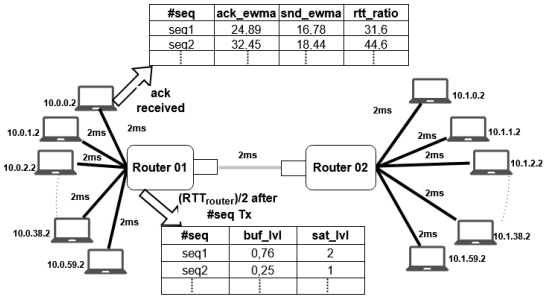

Fig. 5: The topology for getting training, validation, and testing data. This picture also illustrates TA work, raising the metrics to train neural network models. Adapted from [9]

The logic of the connections established in the dumbbell network is as follows:

- Applications are on terminals connected to Router 01, and TCP servers are on those connected to Router 02.
- IP address assignment varies by penultimate byte; Router 01 10.0.0.2 terminal connects to the 10.1.0.2 terminal of Router 02; likewise, the 10.0.1.2 terminal with 10.1.1.2; then 10.0.2.2 station with 10.1.2.2, until the last connection, between stations 10.0.59.2 and 10.1.59.2.

V. Data Mining and Processing

A. Data Mining

The data mining described below is inspired by [9], with some relevant adaptations. Through the dumbbell topology, TA establishes distributed TCP connections and generates two csv files per flow; the first file records the same parameters proposed by [9] for every ACK received by the sender:

- #seq: Sequence number, present in ACK packet.
- ack_ewma: Weighted exponential moving average of arrival time between ACK.
- snd_ewma: Weighted exponential moving average of the interval between timestamps present in ACK.
- rtt_ratio: It is obtained by dividing observed RTT by the minimum (RTT_{min}) during the current experiment.

Unlike [9], however, to update the second csv file, whenever a segment is sent in time t, TA associates sequence numbers with the occupation percentage of the edge router buffer (buf_lvl) to which the transmitters are connected (Router 01 in Figure 5) at t and with the saturation level (sat_lvl), which, now assumes 1 if $buf\_lvl <= 70\%$ or 2, case $buf\_lvl >= 80\%$. Therefore, this work reduces the originally proposed fuzzy area by 66%.

B. Data Processing

The data generated by the TA goes through the same stages introduced by [9] before forming the input vectors for the neural networks:

1) *Inner Join* (IJ): The first step consists of generating a table that associates the ack_ewma, snd_ewma, and rtt_ratio measurements, with buf_lvl and sat_lvl, through the #seq columns, present in both .cvs files collected by the TA.
2) Redundancy Elimination (RE): RE discards lines with identical ack_ewma, snd_ewma, and rtt_ratio, keeping the last record throughout the simulation.
3) Label Balancing (LB): LB assures that the quantity of records with $sat\_lvl = 1$ is the same as those with $sat\_lvl = 2$, discarding excess data.
4) Attenuators Cut (AC): Records with ack_ewma and snd_ewma above the ninetieth percentile (P_{90}) are ignored. Furthermore, records that presented the rtt_ratio above $4\times$ the average RTT value were also considered attenuators and, therefore, removed.
5) Input Normalization (IN): normalization divides each column by its maximum value.

The resulting table associates vectors with components ack_ewma, snd_ewma, and rtt_ratio, properly treated, with class 0 or 1, according to the value of sat_lvl. Note that the proposal is robust against leaks inherent to predictive models[19], since the tool extracts input data from the ACK packets.

[1]The TA code is available at `https://github.com/reisdout/ns3-MTO-CC/blob/main/NS3-MTO-CC.cc`

VI. Models Training and Evaluation

A. Overview

The TA obtains training data for 1.5 minutes by emulating connections among 60 Vegas TCP P2P whose flows provide the corresponding files for Data Mining and Data Processing steps. After Data Processing, we take a 20000-entries sample to train the models. The final transformation process introduced by [9] provides NAA inputs (Figure 9).

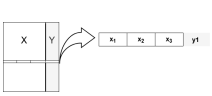

Fig. 6: MLP Vectors

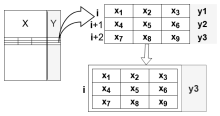

Fig. 7: LSTM Vectors

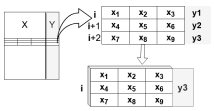

Fig. 8: CNN Vectors

Fig. 9: Table elements regrouped to satisfy the specific format of each NNA. Reprinted from [9]

As in [9], we evaluate twelve models, varying the NNA among MLP, LSTM, and CNN, associated with one combination of components (ack_ewma, snd_ewma, rtt_ratio), following the same model labeling logic. MLP_{123} refers to the MLP network, fed by the three components. $LSTM_{13}$ means LSTM neural network, receiving bidimensional (ack_ewma, rtt_ratio) vectors; same for CNN_{23} (CNN receives (snd_ewma, rtt_ratio)).

B. Model Training - Obtaining the Models

This paper maintains training parameters and AI libraries proposed by [9] (Table I). The process reserves 20% of 20000 table entries for testing and the remaining 80% for training and validation.

TABLE I: The Neural Network configurations with their training parameters. Adapted from [9]

Model	Spec
MLP	**Input:** 3 dimensional vectors. **Layers:** 3 (20 RELU). **Output:** Sigmoid.
LSTM	**Input:** Matrix 3x3. **Layers:** 2 (each one with 3 LSTM - 30% *dropout*). **Output:** Sigmoid.
CNN	**Input:** Image Vectors 3x3@1. **Convolution:**16 maps - 1x2. **Stride:** [1,1]. **Convolution Output:** RELU. **Pooling:** maxpooling, [1,2]. **Flattening:** 64 RELU inputs. **Output:** Sigmoid.
epoch: 3000; batch size 64; learning rate 0.0001; 20% for testing	

Again, Accuracy, Error, Precision, Recall, and F1, obtained from the *Receiver operating characteristics* (ROC) analysis [2], will be used to compare the models extracted from learning networks. The Table II and the ROC space present in figures 10 summarize the results of this methodology phase[3].

C. Analysis of Model Classification Results - Model Evaluation

Regarding the test data results, both ROC space (Figure 10) and Table II indices indicate the good performance of

[2]http://mlwiki.org/index.php/ROC_Analysis#ROC_Space

[3]The research data and notebooks, with the confusion matrices for each model, are available at https://drive.google.com/file/d/1OmzeiLwbyu120ifUBOCUQNlzJ9gzWjDn/view?usp=sharing.

TABLE II: Test metrics over test data. The indexes reveal that most models have accuracy above 95%.

Model	Accuracy	Error	Precision	Recall	F1
MLP_{123}	0.988	0.012	0.992	0.985	0.988
MLP_{13}	0.980	0.020	0.983	0.976	0.979
MLP_{23}	0.985	0.015	0.991	0.979	0.985
MLP_{12}	0.857	0.143	0.917	0.819	0.865
$LSTM_{123}$	0.995	0.005	0.993	0.979	0.986
$LSTM_{13}$	0.994	0.006	0.991	0.980	0.985
$LSTM_{23}$	0.995	0.005	0.991	0.982	0.986
$LSTM_{12}$	0.950	0.050	0.901	0.847	0.873
CNN_{123}	0.989	0.011	0.995	0.951	0.972
CNN_{13}	0.982	0.018	0.992	0.919	0.954
CNN_{23}	0.980	0.020	0.985	0.914	0.948
CNN_{12}	0.948	0.052	0.903	0.837	0.869

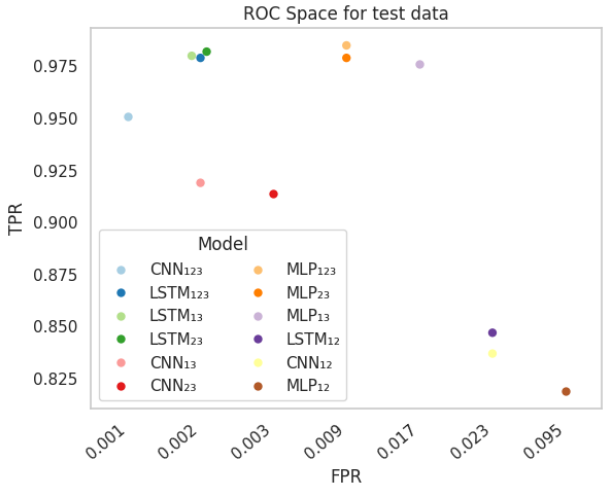

Fig. 10: ROC space for test data. The concentration of points around the point (0,1) (TPR = 1 and FPR = 0) reveals high rates of true positives and a low number of false positives for all models.

the $LSTM_{123}$, closely followed by $LSTM_{23}$. In the table, $LSTM_{123}$ and $LSTM_{23}$ architectures are the best in 4 of 5 presented metrics[4]. In ROC space, their points stand out, considering the distance to the lines TPR = 1 and FPR = 0. Also noteworthy are the MLP models, whose points are very close to line TPR = 1. With Accuracy close to 99.5%, $LSTM_{13}$; MLP_{123}, with the highest Recall and F1 and CNN_{123} (with the highest Precision) are very promising. Therefore, the best classifiers are those originating from the $LSTM_{123}$ and $LSTM_{23}$ models. These models will input the next phase of the methodology, the AIN'T construction, as described in the following section.

VII. AIN'T Construction

The AIN'T construction adapts the proposal presented by [9]. The Keras2c library [20] generates the best models ($LSTM_{123}$ and $LSTM_{23}$) C implementation from Keras. Following the observer design pattern [21], each Keras2c output code implements one C++ concrete class. AIN'T activates these classes according to configuration parameters.

During the simulations, TA measures the ACK arrival interval and the elapsed time between transmission and RTT.

[4]Model selection prioritizes Accuracy. Models that highlight this metric automatically have another one, the Error.

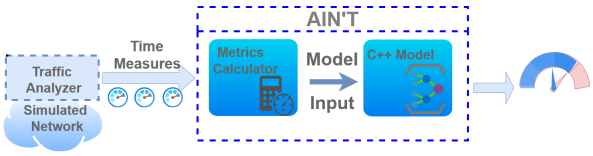

Fig. 11: AIN'T design. This figure shows how to integrate AIN'T into a TCP network to monitor a network, highlighting the main components and their respective integration.

TA passes these values to the AIN'T tool, which calculates the corresponding model's input. AIN'T tool indicates the bandwidth saturation with C++ model implementation outputs. Figure 11 illustrates that.

The proposed architecture provides a precise AIN'T update interval projection. The TA calculates the time measurements each time an ACK packet arrives at the transmitter. Thus, in an RTT time scale, TA starts AIN'T (figure 11) that updates the link saturation level. That means that the AIN'T monitoring is practically online. The following section shows AIN'T in action.

VIII. Network monitoring via AIN'T

The experiment setup implemented the above-mentioned design to verify the AIN'T classification capacity. To this end, the TA performed a 10-minute simulation on the aforementioned dumbbell network (Figure 5) as follows:

- During simulation, for each ACK received by the client with IP 10.0.0.2, the TA delivers the time measurements to AIN'T.
- If AIN'T provided three identical responses, the TA records the respective response and the real buffer occupancy percentage. That allows for comparing the efficiency of AIN'T in classifying bottleneck saturation.
- From 9 min onwards, except for the first client, the others closed their flows gradually, every 1 second. This approach aims to test the behavior of AIN'T facing different numbers of flows and transitions between low and high buffer occupancy.

AIN'T tool passed this test twice, activating the LSTM_{123} and LSTM_{23} models, respectively, as they stood out during the model evaluation phase. The Graphs of figures 12 and 13 show the results.

IX. AIN'T Performance Analysis

The presentation of the graphs in the figures 12 and 13 provides a quick assessment of the AIN'T tool's performance. The horizontal green dotted line highlights the training threshold (80%). If the queue occupancy percentage is below this line, the AIN'T should respond with level 1; otherwise, it should signal a value of 2. The overview of both graphs shows how efficient AIN'T was throughout the experiments.

Both versions of AIN'T, with LSTM_{123} and LSTM_{23}, show a sticky response profile within the training proposal:

- Practically, for the entire time that the router buffer is below 80%, the line corresponding to the AIN'T response remains in 1, as well as in 2 otherwise.
- Another interesting aspect is the rapid transition in the models' outputs at any relief in the router buffer. From ACK 1600 on, the AIN'T tool follows any pulse of the blue line below 80%, starting to respond with a level 1 occupancy a few ACKs later.
- From ACK 4050 onwards, a definitive reduction of the router buffer begins. The AIN'T quickly notices this change and immediately starts to follow it after a short period of instability.
- This efficiency in the transition of the AIN'T responses is also demonstrated in the opposite direction when the buffer breaks the 80% limit. The graph shows this between the points of ACK 1500 and 1600.

Although very similar, the LSTM_{123} and LSTM_{23} models present notable differences, mainly in the buffer transition areas. The AIN'T version implementing the LSTM_{123} class demonstrates greater efficiency in these points, going through shorter periods of instability with fewer transitions between levels 1 and 2. That can be verified by observing:

- The first transition (ACK 1450 to ACK 1500 in graph 1, and ACK 1550 to ACK 1600 in graph 2).
- At the buffer drawdown peaks throughout the entire period of high buffer predominance, between ACK 1500 to ACK 3900 in graph 1, and ACK1600 to 4050 in graph 2
- During the final transition to buffer below 80 in both graphs.

Thus, the experiments show a slight advantage when AIN'T operates the model extracted from the LSTM_{123} architecture over the one provided by LSTM_{23}.

X. Conclusion

The presented technique provides efficient models for detecting high or low link saturation. All AI models accurately classify the network bottleneck condition, with an accuracy of over 99% (LSTM_{123}, LSTM_{13}, LSTM_{23}) in highly complex scenarios (120 stations establishing 60 TCP flows). Therefore, the results presented by the models on the test data attest that Deep Learning models are capable of classifying the degree of link saturation (RQ1).

AIN'T, equipped with the models that presented the best performance, was tested in a new environment, different from the training setup, varying the number of flows in the network topology. During these long-term tests, the classification provided by AIN'T remained adherent to the different levels of buffer occupancy presented, promptly indicating their variations. So, the built models have a sufficient degree of generalization to support variations in the number of flows (RQ2).

The tests, in an unprecedented initiative, show the efficiency of Deep Learning Networks for activities related to passive link monitoring. It is worth noting that neural networks with low-dimensional inputs and few layers presented high values in all indices on the test data. During the AIN'T evaluation, the most accurate classifiers tracked link saturation, proving that Deep Learning models with low-dimensional input are capable of passively distinguishing the degree of saturation of a link from a certain threshold (RQ3).

However, some questions remain open for future work. Numerous other factors can affect the time measurements obtained as model input (jitter, burst losses, link unavailability, etc.). Another relevant issue is the investigation of training stopping criteria, different from the number of epochs. That

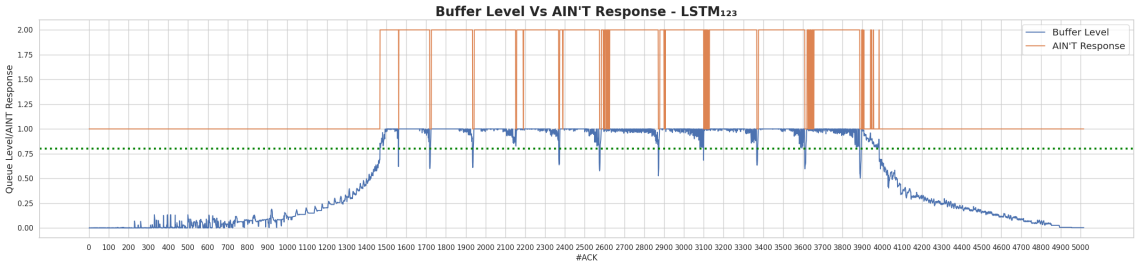

Fig. 12: The graph shows the AIN'T responses, operating the model extracted from the LSTM_{123}, and the buffer occupancy percentage, recorded simultaneously. The profile produced by the AIN'T responses, except for transition moments, remains low when the buffer is below 80% and high otherwise.

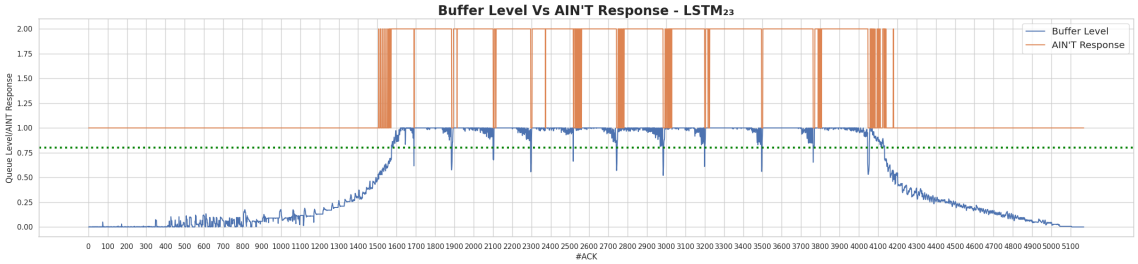

Fig. 13: This graph is analogous to the one in figure 12, but with AIN'T operating the LSTM_{23} mode. The model, during the experiment, indicates the buffer occupancy level except for transition moments.

opens space for new research to verify how the models behave under the influence of these factors.

This work represents the first phase of the research. The following steps will apply the approach outlined to real networks for explicit comparison with solutions proposed in the literature, thereby positioning the proposal among other available bandwidth measurement tools. Many of the resources available in the simulation environment are not present in concrete-component nets, which poses new challenges to AIN'T scalability.

REFERENCES

[1] M. Hinago and F. P. Piurcosky, "A capacitação no projeto SISFRON: as lições aprendidas do projeto piloto e as perspectivas para o prosseguimento das próximas fases." pp. 285–320, Dec. 2021. [Online]. Available: https://ojs.ufgd.edu.br/index.php/moncoes/article/view/14387

[2] "iperf3 - a tool for active measurements of the maximum achievable bandwidth on ip networks." [Online]. Available: https://iperf.fr/

[3] "nuttcp - network performance measurement tool." [Online]. Available: https://www.nuttcp.net

[4] O. Basit, P. Dinh, I. Khan, Z. J. Kong, Y. C. Hu, D. Koutsonikolas, M. Lee, and C. Liu, "On the predictability of fine-grained cellular network throughput using machine learning models," in *2024 IEEE 21st International Conference on Mobile Ad-Hoc and Smart Systems (MASS)*, 2024, pp. 47–56.

[5] "perfsonar - performance service-oriented network monitoring architecture." [Online]. Available: https://www.perfsonar.net/index.html

[6] A. Mazloum, A. AlSabeh, E. Kfoury, and J. Crichigno, "perfsonar: Enhancing data collection through adaptive sampling," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, 2024, pp. 1–6.

[7] S. Aggarwal, U. S. Sardesai, V. Sinha, D. D. Mohan, M. Ghoshal, and D. Koutsonikolas, "Libra: learning-based link adaptation leveraging phy layer information in 60 ghz wlans," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 245–260. [Online]. Available: https://doi.org/10.1145/3386367.3431319

[8] J. Hall, J. M. Jornet, N. Thawdar, T. Melodia, and F. Restuccia, "Deep learning at the physical layer for adaptive terahertz communications," *IEEE Transactions on Terahertz Science and Technology*, vol. 13, no. 2, pp. 102–112, 2023.

[9] C. A. C. Marcondes and M. R. d. Siva, "Redes de Aprendizado Profundo para Classificação e Controle de Congestionamento em Redes TCP/IP," in *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, 2024, pp. 57–70.

[10] R. Kazama, H. Abe, and C. Lee, "Evaluating TCP throughput predictability from packet traces using recurrent neural network," in *2022 IEEE Symposium on Computers and Communications (ISCC)*, 2022, pp. 1–6, ISSN: 2642-7389.

[11] L. Bai, H. Abe, and C. Lee, "RNN-based Approach to TCP throughput prediction," in *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*. Naha, Japan: IEEE, Nov. 2020, pp. 391–395. [Online]. Available: https://ieeexplore.ieee.org/document/9355940/

[12] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-down Approach*, 8th ed. Pearson.

[13] N. Cotter, "The Stone-Weierstrass theorem and its application to neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 290–295, Dec. 1990, conference Name: IEEE Transactions on Neural Networks.

[14] M. A. Nielsen, "Neural Networks and Deep Learning," 2015, publisher: Determination Press. [Online]. Available: http://neuralnetworksanddeeplearning.com

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

[16] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, Mar. 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10721-6

[17] L. E. Falqueto, R. L. Paes, and A. Passaro, "KNN e Rede Neural Convolucional para o Reconhecimento de Plataformas de Petróleo em Imagens SAR do Sentinel-1," *Spectrum - The Journal of Operational Applications in Defense Areas*, vol. 24, no. 1, pp. 29–33, Sep. 2023. [Online]. Available: https://spectrum.ita.br/index.php/spectrum/article/view/395

[18] "ns3 - a discrete-event network simulator for internet systems."

[19] A. Apicella, F. Isgrò, and R. Prevete, "Don't push the button! Exploring data leakage risks in machine learning and transfer learning," *Artificial Intelligence Review*, vol. 58, no. 11, p. 339, Aug. 2025, 0000118-Data leakage in machine learning. [Online]. Available: https://doi.org/10.1007/s10462-025-11326-3

[20] R. Conlin, K. Erickson, J. Abbate, and E. Kolemen, "Keras2c: A library for converting Keras neural networks to real-time compatible C," *Engineering Applications of Artificial Intelligence*, vol. 100, p. 104182, Apr. 2021.

[21] D. Nesteruk, *Design Patterns in Modern C++: Reusable Approaches for Object-Oriented Software Design*, 1st ed. New York: Apress, Jan. 2018.