# On Enhancing Network Throughput using Reinforcement Learning in Sliced Testbeds

**Daniel Pereira Monteiro**[1], **Lucas Nardelli de Freitas Botelho Saar**[1], **Larissa Ferreira Rodrigues Moreira**[1,2], **Rodrigo Moreira**[1]

[1]Institute of Exact and Technological Sciences – Federal University of Viçosa (UFV)
Rio Paranaíba – MG – Brazil.

[2]Faculty of Computing (FACOM) – Federal University of Uberlândia (UFU)
Uberlândia – MG – Brazil.

{daniel.p.monteiro, lucas.saar, larissa.f.rodrigues, rodrigo}@ufv.br

larissarodrigues@ufu.br

***Abstract.*** *Novel applications demand high throughput, low latency, and high reliability connectivity and still pose significant challenges to slicing orchestration architectures. The literature explores network slicing techniques that employ canonical methods, artificial intelligence, and combinatorial optimization to address errors and ensure throughput for network slice data plane. This paper introduces the Enhanced Mobile Broadband (eMBB)-Agent as a new approach that uses Reinforcement Learning (RL) in a vertical application to enhance network slicing throughput to fit Service-Level Agreements (SLAs). The eMBB-Agent analyzes application transmission variables and proposes actions within a discrete space to adjust the reception window using a Deep Q-Network (DQN). This paper also presents experimental results that examine the impact of factors such as the channel error rate, DQN model layers, and learning rate on model convergence and achieved throughput, providing insights on embedding intelligence in network slicing.*

## 1. Introduction

Disruptive applications, such as 8K video streaming, Virtual Reality (VR), and Augmented Reality (AR), had led to an increased demand for high network throughput [Khan et al. 2022]. Additionally, other application families, including remote surgery, smart factories, and autonomous vehicles, require low-latency and high-reliability connectivity [Aripin et al. 2023]. Ensuring the compatibility between these conflicting requirements within a physical network is a significant challenge for both management and resource orchestration [Khan et al. 2022]. To address this, various advances have been made in network slicing, virtualization, programmability, security, and Artificial Intelligence (AI), especially in mainstream mobile networks, such as beamforming and energy-aware solutions [Khan et al. 2022, Moreira et al. 2023, Brilhante et al. 2023].

The literature has explored approaches for performing network slicing that can handle errors in the underlying channel while ensuring throughput using canonical techniques such as artificial intelligence, and combinatorial optimization [Ojijo and Falowo 2020, Liu et al. 2023]. Some of these techniques involve intervention in the link [Moreira et al. 2021], while others involve intervention in the communicating entity, such as those based on Transmission Control Protocol (TCP) [Li et al. 2019b, Siddiqi et al. 2022] Congestion Control. The proposed control in the communicating entities aims to reduce the reception window in the event of packet loss, thereby reducing the amount of traffic in the network. However, this approach is not sufficiently flexible to incorporate intelligent mechanisms seamlessly.

In this paper, we propose and evaluate the `eMBB Agent`, which is based on Reinforcement Learning (RL) coupled with a vertical application to improve the throughput of network slicing to guarantee Service-Level Agreement (SLA). Functionally, `eMBB-Agent` analyzes the vertical application variables and proposes actions within a discrete space to increase or decrease the reception window. Subsequently, the `eMBB-Agent` verifies the effectiveness of its actions through an Deep Q-Learning (DQN). Experimentally, we verified how factors such as the channel error rate, number of layers in the DQN model, and learning rate impact the model convergence and achieved throughput.

The remainder of this paper is organized as follows. In Section 2, we contextualize the related work on intelligent throughput enhancement. The proposed experimental method is presented in detail in Section 3, followed by a description of the experimental setup and results in Section 4. Section 5 discusses concluding remarks and future research directions.

## 2. Related Work

Recent efforts, such as [Zhang et al. 2019], have sought to improve the *Multipath Transmission Control Protocol (MPTCP)* protocol through reinforcement learning techniques. Using asynchronous training, this study allows parallel execution of packet scheduling, data collection, and neural network training. The goal was to optimize scheduling in real time by employing an asynchronous algorithm for neural training.

The work proposed by [Li et al. 2019a] aimed to improve network efficiency using the *SmartCC* algorithm. This algorithm employs reinforcement learning techniques to improve the congestion window management. *SmartCC* uses an asynchronous reinforcement learning mechanism to acquire a set of congestion rules. While [Tang et al. 2018] presented a traffic prediction algorithm based on deep learning. This algorithm aims to anticipate the workload and network congestion. After the prediction, partial channel allocation based on deep learning is performed to prevent possible congestion by assigning appropriate channels.

The study carried out by [Beig et al. 2018] examined mobile users using the MPTCP protocol, with the aim of optimizing congestion control in heterogeneous networks. This study proposes an algorithm based on *Q-learning (QL)* to improve *throughput*, with the aim of maximizing throughput. [Vieira and Garcez 2011] developed a mathematical expression to calculate the probability of data loss on the servers. This expression is used to condition the estimation of the probability of data loss in analog servers that have a finite *buffer* and receive time-dependent multifactor flows.

Table 1 aims to clarify the standards adopted by related studies in relation to the metrics and technologies used. The *throughput* metric is frequently used as an evaluation criterion in several studies. Another observed constant is its use as a search variable for designing experiments.

**Table 1. Prior works aimed to enhance network conditions using AI.**

| Approach | Evaluation Metric | Search Variable | AI | Evaluation Testbed |
|---|---|---|---|---|
| [Tang et al. 2018] | Throughput | Software-Defined Networking (SDN) Controller | Reinforcement Learning | API C++/ WILL |
| [Beig et al. 2018] | Throughput | Throughput | Q-Learning | NS3 |
| [Vieira and Garcez 2011] | Probability, buffer | Buffer | Does Not Use | Own |
| [Zhang et al. 2019] | Goodput, Delay, Download | MinRTT, Round-Robin | Deep Reinforcement Learning (DRL) | Own |
| [Li et al. 2019a] | ACK, Round Trip Time (RTT) | Throughput, RTT, Jitter | RL | Own |
| **Our Approach** | Throughput, and RTT | Congestion Window | DQN | NS3 on Fabric |

## 3. Evaluation Method

This study examines the influence of various factors, including the number of layers in the DQN model, percentage of channel error, and learning rate, on the convergence time and data transmission rate (throughput) between two applications. The study employs metrics such as congestion window size, packet size, total number of bytes sent, average, total number of recognized segments, and network throughput. The aim is to optimize the current throughput on a link generated by the NS3 simulator, focusing on the impact of the `eMBB-Agent` on the search space, as shown in Fig. 1.

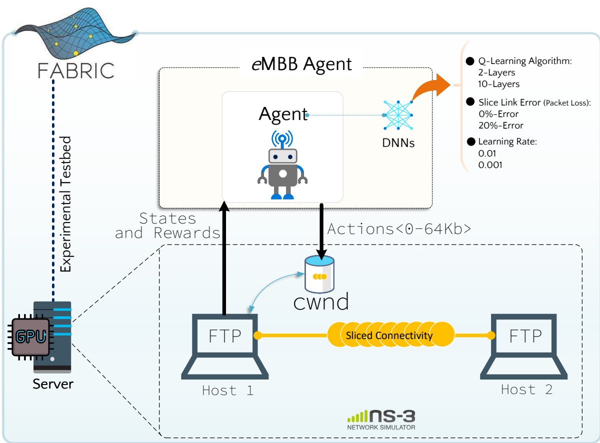



**Figure 1. Proposed Evaluation Method.**

We used Network Simulator 3 (NS3) to create a network topology and simulate the transmission of packets between two File Transfer Protocol (FTP) applications, as shown in Fig. 1. We set the bandwidth between the hots to 10 Mbps and 2 Mbps between the routers to induce congestion, as illustrated in Fig. 2. We used configurations *NN-2* containing two hidden layers, *NN-4* with four hidden layers, and *NN-8* with eight hidden layers.

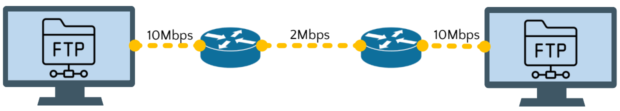



**Figure 2. Experiment Topology.**

Through partial factorial combination, that is, we take combinations two-by-two and carry out experiments on the levels of variations to verify the influence of these variations. Each combination was run 10 times to generate a statistical sample. We measured the error rate in packets in scenarios of $0\%$ and $20\%$, and varied the learning rate hyper-parameter to 0.01 and 0.001. The parameters of the combinations performed are listed in Table 2.

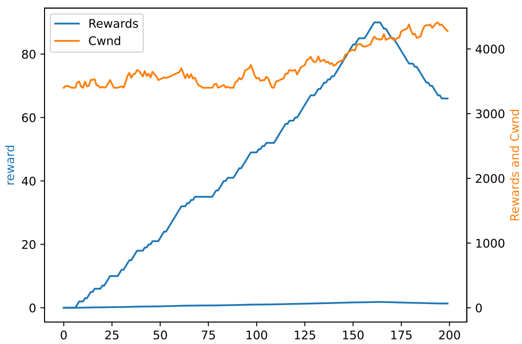
**Table 2. Factorial-partial Experimentation Combinations.**

| Factor | Levels | | |
|---|---|---|---|
| **# Layers DQN Algorithm** | 2 | 4 | 8 |
| **Learning Rate** | 0.01 | | 0.001 |
| **Network Error Rate** | 0% | | 20% |

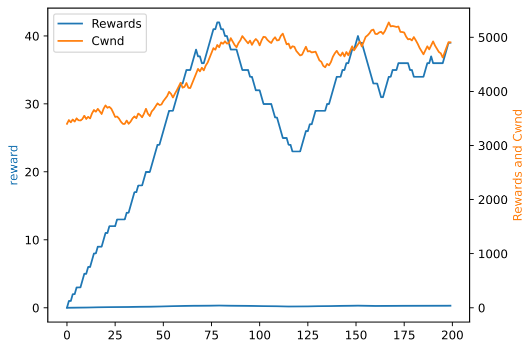
To train the DQN models and their variations, we used RTX 4060*ti* 16 Gb GPU hardware with an Intel(R) Core (TM) i5-4430 CPU @ 3.00GHz with 32 GB of acRAM.

## 4. Results and Discussion

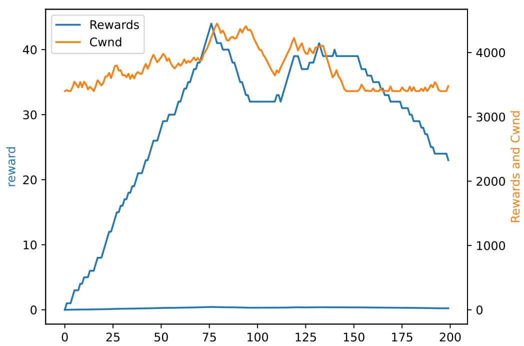
The central objective of the experiments was to analyze the behavior of the network in response to different configurations of Neural Networks (NNs) by using the DQN algorithm with the NS3-GYM [Gawlowicz and Zubow 2018] tool. With this technology, it has become possible to combine RL algorithms and interventions in the control mechanisms of communication networks by using NS3. In this study we considered NS3-GYM online.




**(a) NN-2**




**(b) NN-4**

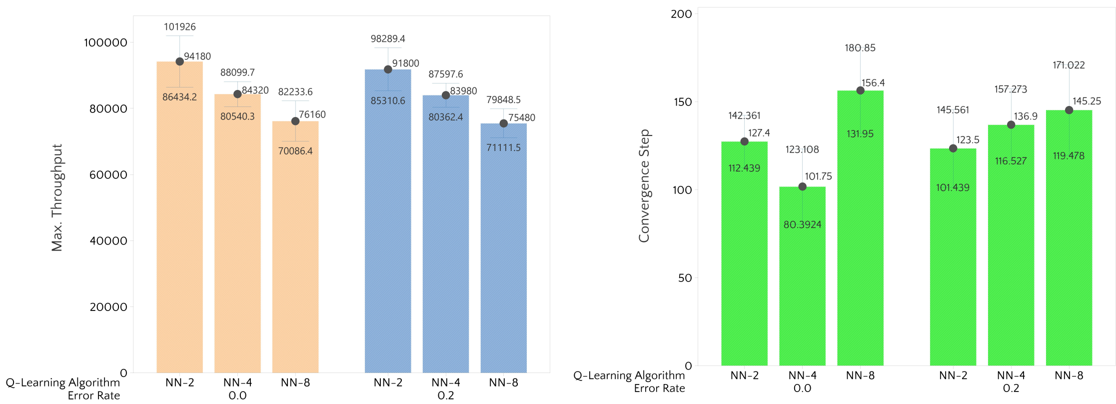



**(c) NN-8**

**Figure 3. Increasing *cwnd* using different DQN structures.**

During the simulation, a progressive increase in the congestion window was observed, which was directly related to a significant increase in the rewards obtained by the `eMBB-Agent`. According to Fig. 3, advantageous network flow rates were identified with the implementation of the *Neural Network (NN)-2* model containing two layers. In detail, Fig. 3a, 3b and 3c provide a visual representation of the algorithm over 200 *steps*.

Thus, it is possible to verify that the larger the size of the *cwnd* variable in a smaller number of epochs, the faster `eMBB-Agent` increases the communication throughput.




**(a) Maximum throughput achieved by different *NN* configurations.**

**(b) Convergence step required by different *NN* configurations.**

**Figure 4. Analyses of a physical link with ($20\%$) and without ($0\%$) errors.**

The *NN* configuration, shown in Fig. 3a, 3b and 3c exhibit a progressive increase in throughput with an increasing congestion window *cwnd* based on decisions and receiving rewards for correct choices. Three models (*NN-2*, *NN-4*, and *NN-8*) were analyzed, and *NN-4* presented a lower average flow rate considering *NN-2* and *NN-8*. Subsequently, *NN-8* manifested the second-worst throughput. This behavior is attributed to the time required to train a DQN with more layers, given the time sensitivity in the experimental scenario.

Fig. 4a highlights the performance of the *Q-Learning* algorithm in three configurations: *NN-2*, *NN-4* and *NN-8*, representing the average throughput of the network slice in an error-free and with error in slicing. *NN-2*, with an error-free demonstrates the best performance, followed by *NN-4*, while *NN-8* displays lower performance, with a lower Average Network Throughput. We associate this with its complexity, which requires more computational resources owing to its deep neural network structure.

Fig. 4b shows the convergence times of the DQN algorithm for three different architectures: *NN-2*, *NN-4* , and *NN-8*. As can be seen, in an error-free channel, algorithm *NN-4* exhibits the best performance, with a convergence time $20.09\%$ lower than *NN- 2* and $32.99\%$ lower than *NN-8*, respectively.

Alternatively, in a channel with $20\%$ error induction, *NN-2* reaches convergence, that is, fullness in the second variable faster. Thus, the convergence time for *NN-2* was $10.34\%$ less than that of *NN-4* and $15.00\%$ less than that of *NN- 8*. We associate this better performance of the simpler *NN* with better training time.

Finally, we investigate the effect of the error rate, learning rate, and RL algorithm on network throughput through regression. Tables 3 and 4 present the variables considered, their estimated impacts, associated coefficients, standard errors, and T and P-*values*, providing information about the relationships between Algorithm *DQN* , error rate, learning rate, and network throughput.

**Table 3. Influence of *network error rate* and DQN structure on Network Throughput.**

| Term | Influence | Coefficient | Standard Error | T-*Value* | P-*Value* |
|---|---|---|---|---|---|
| Constant | | 84320 | 1075 | 78.41 | 0.000 |
| Network Error Rate | -1133 | -567 | 1075 | -0.53 | 0.599 |
| DQN Algorithm | -17170 | -8585 | 1317 | -6.52 | 0.000 |
| Network Error Rate × DQN Algorithm | 850 | 425 | 1317 | 0.32 | 0.748 |

**Table 4. Influence of *network error rate* and *learning rate* on Network Throughput.**

| Term | Influence | Coefficient | Standard Error | T-*Value* | P-*Value* |
|---|---|---|---|---|---|
| Constant | | 84320 | 1075 | 78.41 | 0.000 |
| Network Error Rate | -1133 | -567 | 1075 | -0.53 | 0.599 |
| Learning Rate | 907 | 453 | 1075 | 0.42 | 0.678 |
| Network Error Rate × Learning Rate | 680 | 340 | 1075 | 0.32 | 0.748 |

## 5. Concluding Remark

The analysis of congestion algorithms with various combinations of artificial *NN* demonstrated an inverse correlation between the number of layers and the efficiency of optimizing the flow in communication networks, as indicated by the data from linear regression analysis. This suggests that an increase in network complexity leads to a decrease in network throughput. The study found that neither the network error rate nor the learning rate had a statistically significant effect on the network throughput.

This study struggled with some constraints, one of which was that the analysis was carried out in a simulated environment using Network Simulation Library 3 (NS3). To overcome this limitation, we suggest that future research explore the functionality of `eMBB-Agent` in a real-world setting, in which variables can be manipulated to assess its impact in more complex and dynamic situations. This provided a more accurate and comprehensive understanding of the practical implications of the findings across various operational scenarios. In addition, we suggest the optimization of additional parameters and evaluation of latency and reliability to further improve network performance.


## Acknowledgments

We acknowledge the financial support of the FAPESP MCTIC/CGI Research project 2018/23097-3 - SFI2 - Slicing Future Internet Infrastructures. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.


## References


Aripin, N. M., Zulkifli, T., and Radzi, N. A. M. (2023). Performance Analysis of 5G Network Slicing for Hospital of the Future. In *2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 18–21.

Beig, E. F. G. M., Daneshjoo, P., Rezaei, S., Movassagh, A. A., Karimi, R., and Qin, Y. (2018). Mptcp throughput enhancement by q-learning for mobile devices. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1171–1176, Conference. IEEE.

Brilhante, D. d. S., Manjarres, J. C., Moreira, R., de Oliveira Veiga, L., de Rezende, J. F., Müller, F., Klautau, A., Leonel Mendes, L., and P. de Figueiredo, F. A. (2023). A Literature Survey on AI-Aided Beamforming and Beam Management for 5G and 6G Systems. *Sensors*, 23(9).

Gawlowicz, P. and Zubow, A. (2018). ns3-gym: Extending openai gym for networking research. *CoRR*, abs/1810.03943.

Khan, B. S., Jangsher, S., Ahmed, A., and Al-Dweik, A. (2022). URLLC and eMBB in 5G Industrial IoT: A Survey. *IEEE Open Journal of the Communications Society*, 3:1134–1163.

Li, W., Zhang, H., Gao, S., Xue, C., Wang, X., and Lu, S. (2019a). SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks. *IEEE Journal on Selected Areas in Communications*, 37(11):2621–2633.

Li, W., Zhou, F., Chowdhury, K. R., and Meleis, W. (2019b). QTCP: Adaptive Congestion Control with Reinforcement Learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458.

Liu, Y., Clerckx, B., and Popovski, P. (2023). Network Slicing for eMBB, URLLC, and mMTC: An Uplink Rate-Splitting Multiple Access Approach. *IEEE Transactions on Wireless Communications*, pages 1–1.

Moreira, R., Rodrigues Moreira, L. F., and de Oliveira Silva, F. (2023). An intelligent network monitoring approach for online classification of Darknet traffic. *Computers and Electrical Engineering*, 110:108852.

Moreira, R., Rosa, P. F., Aguiar, R. L. A., and de Oliveira Silva, F. (2021). Deploying Scalable and Stable XDP-Based Network Slices Through NASOR Framework for Low-Latency Applications. In Barolli, L., Woungang, I., and Enokido, T., editors, *Advanced Information Networking and Applications*, pages 715–726, Cham. Springer International Publishing.

Ojijo, M. O. and Falowo, O. E. (2020). A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network. *IEEE Access*, 8:14977–14990.

Siddiqi, S. J., Naeem, F., Khan, S., Khan, K. S., and Tariq, M. (2022). Towards AI-enabled traffic management in multipath TCP: A survey. *Computer Communications*, 181:412–427.

Tang, F., Fadlullah, Z. M., Mao, B., and Kato, N. (2018). An Intelligent Traffic Load Prediction-Based Adaptive Channel Assignment Algorithm in SDN-IoT: A Deep Learning Approach. *IEEE Internet of Things Journal*, 5(6):5141–5154.

Vieira, F. H. T. and Garcez, S. G. (2011). Estimação de probabilidade de perda de dados em redes através de modelagem multifractal de tráfego e teoria de muitas fontes. *Revista de Informática Teórica e Aplicada*, 18(1):13–30.

Zhang, H., Li, W., Gao, S., Wang, X., and Ye, B. (2019). ReLeS: A Neural Adaptive Multipath Scheduler based on Deep Reinforcement Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1648–1656, INFOCOM. IEEE.