Proactive and Reactive Decision Based Agent Placement: Reliability and Latency Perspective

Sisay Tadesse Arzo * Mona Esmaeili†, Yonatan Melese Worku‡, Zeinab Akhavan§, Michael Devetsikiotis¶, and Payman Zarkesh-Ha,‖
*†‡¶‖Department of Electrical and Computer Engineering, University of New Mexico, USA
§Department of Computer Science, University of New Mexico, USA
{*sarzo, †mesmaeili, ‡yonatanmelese61 §zakhavan ¶mdevets ‖pzarkesh}@unm.edu

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001516

***Abstract*—6G is aiming at fully incorporating in-network intelligence towards automated network management. In this regard, a multi-agent-based network automation architecture as a service design is proposed. The architecture introduces in-network intelligence, designing intelligent agents as the fundamental unit which is used as a building block in autonomous network system design. This work focuses on the dynamic agent placement problems in edge/cloud data centers. Agents are softwarized and intelligent versions of network functions that are traditionally implemented in hardware such as firewalls, packet gateways, etc. This paper, based on a combination of proactive and reactive solutions, considered decision accuracy in developing an intelligent decision algorithm that can be used in the prediction agent design. The proactive decision is based on a deep learning prediction algorithm using time-series workload forecasting. However, in case of unforeseen events that are missing from the historical dataset, the proactive decisions could be less reliable. Therefore, network-state feedback should be considered to determine the current network conditions using the change in instant arrival rate as a reactive decision. Then combining it with the proactive decision should be able to capture the unpredictable traffic spikes. The result is used to determine the number and type of agents to instantiate at a given time in the edge/cloud data centers. Using a public dataset in our algorithms, we predicted the workload request for a few days. The result shows improved decision accuracy over the existing solutions using the appropriate amount of dataset, machine learning models, and rate estimation.**

I. Introduction

In-network intelligence is actively being considered as the main characteristic for 6G network automation. Therefore, full network automation is expected to be the feature of future networks such as 6G [1]. The current trend is to develop these properties through network softwariziation being supported by software define networking (SDN) and network function virtualization (NFV). To enable network automation, a multi-agent-based architecture for service design is proposed [2], [3]. The architecture introduces in-network intelligence designing intelligent agents as the fundamental unit which can be used as a building in autonomous network system design. Multi-agent architecture is competing with microservice-based service design approaches having an advantage over microservice-based service design approaches. This is mainly due to the multi-agent service design can provide both proactive and reactive responses while microservice can only respond reactively (responds to http request).

978-1-6654-3540-6/22 © 2022 IEEE

Agents have to be carefully designed to perform a particular function. An example of agent design is provided in [4] which shows a classifying agent design. In the paper, it has been emphasized that the accuracy of the incorporated deep learning(DL)/machine learning(ML) model determines the reliability of the agent as a proactive decision-maker. However, since the designed agent uses only the proactive DL/ML model, the accuracy may severely be affected for reactive events. Such events are never seen in the historical dataset that is used to train the ML/DL model used as the brain of the agent in the decision-making process.

One aspect of multi-agent based network automation is dynamic agent placement, chaining, and orchestration. Agents are virtualized intelligent units that can be used in multi-agent system design. Automated agent placement in edge/cloud data centers is very useful and necessary for Quality of Service (QoS) improvement in an automated network system. However, agent placement, chaining, and orchestrations, depending on various user QoS requirements and service arrival distributions while considering edge computational resources, are challenging network automation processes.

In this work, we focus on traffic-aware reactive and proactive agent placement in an edge/cloud data center. First, We formulate an algorithm combining proactive and reactive decisions. Then, we design the agent using the combination of DL models as a proactive prediction and rate estimation algorithms as a reactive adaptation. Using this combined algorithm, the agent predicts the number and type of agents to instantiate and create an agent chain. We used long short-term memory (LSTM), Gated Recurrent Unit (GRU), convolutional neural network (CNN), and a multilayer perceptron (MLP) as a comparison of proactive agent prediction. Furthermore, we used rate estimation based on a given threshold.

In general, our contribution is summarized as:

- Formulate an algorithm combining reactive and proactive decisions.
- Design a prediction agent using the formulated algorithm as a brain of the agent.
- Use the design prediction agent, predicting number and type of service processing agents, creating and allocating agents in edge/cloud data center as a function of incoming traffic.
- Evaluate the performance of the prediction agent.

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.

The rest of the paper is organized as follows. Section II presents related works. We discuss our method which combine proactive and reactive decision for agent placement using a time serious DL and rate estimation technique in Section III. Section IV is dedicated for performance evaluation of the proposed method. Finally, we present our concluding remark in Section V.

II. Related Works

1) Deterministic VNF resource allocation and management: Agent placement has some similarities with VNF placement with the added in-network intelligence incorporated in the agent, which could impact the general placement requirements. Since VNF placement is relatively a well-studied subject in the literature now, let us review some of the most recent approaches. VNF placement with deterministic objective has been studied in various literature which uses dynamic programming [5], [6]. Some other works, using methods such as genetic algorithm-based dynamic VNF placement and service function chaining [7]–[9], have also been presented. However, they lack the network and service arrival adaptability that is required by autonomic network management. Few research works have also discussed resource allocation and VNF placement considering specific network environments such as edge and geo-distributed data centers [10], [11].

Autonomic networking requires the self-managing adaptive capability for unpredictable network and user demands. Machine learning (ML) exploits information from the network and users' data to learn and perform the required change to suit the demands. In [12], an autonomic VNF placement is discussed considering a three data center hierarchy for workload offloading between the data centers depending on the traffic load. In [13], the authors presented a method called z-TORCH orchestration mechanism that uses unsupervised learning to monitor VNF KPI and reinforcement learning to a find a trade-off solution for reliability and complexity of the monitoring system. Time series prediction for VNF placement is presented in [14], [15]. The authors in [14] first analyzed the traffic characteristics of the data center and devised a traffic forecasting technique. Based on traffic forecasting, they develop a deterministic algorithm to determine the VNF resource scaling.

Our proactive prediction decision is similar to the work in [16] that uses an LSTM-based prediction for VNF. However, this the has limitation of capturing the sudden unexpected traffic spike. In [17], the authors used a reinforcement learning-based method for reactive VNF placement. However, in addition to the delay to reach a decision, the cost of online training in terms of computing to reach a decision is very high which makes this method sub-optimal.

An interesting deterministic algorithm is presented in [15] that combining online and offline elastic resource prediction as a function of incoming traffic for dynamic VNF placement. This work is the closest to our approach in that attempted to characterize the normal traffic through an offline processing method that captures the regular traffic pattern. Moreover, the work also discussed an online method to capture the sudden expected traffic spike while evaluating its correlation with the offline method. Furthermore, they developed an algorithm that combined the two methods. In addition to the deterministic nature of the developed algorithm, this work also didn't consider other factors such as reliability and latency for a DL model training and prediction for VNF placement in an edge data center scenario. A proactive -reactive auto scaling algorithms using MLP as the proactive decision is proposed in [18]. This paper has also limitations in-terms as it only consider data center resource instead of agents. Moreover, they didn't evaluate the combine and contradictory effect of considering service requirements such as reliability, latency, resource efficiency and energy consumption. To the best of the authors knowledge, there is no work that combined proactive decision and reactive decision while considered decision accuracy for automated agent placement in edge/cloud environment

III. Overview of Traffic Aware Agent Placement

A multi agent architecture for network automation provided in [2] requires designing agents and orchestrating them to create agent chain to compose the autonomous system. In a distributed cloud/edge environment, where agents could be instantiated and chained to provide the workload processing arriving at the data center, agent instantiating and placement are directly related to the incoming traffic with the available resource constraint. The prevalent approach is using classical forecasting algorithms or ML/DL methods to predict the number of agent N_{Ag} to instantiate at a give time, which is predicting N_{Ag} as a function of users/service workload request. This requires learning the patter from historic data D. In mathematical terms, determining the posterior probability N_{Ag} given prior information D. Using Bayes' Theorem we will have

$$P(N_{Ag}/D) = \frac{P(N_{Ag})P(D/N_{Ag})}{P(D)} \tag{1}$$

The important factor here is the final decision need to be achieved with certain level of confidence for the decision to be reliable. Our aim here is to improve accuracy to be formulated as a resource forecast problem. Since workload traffic have hourly, daily, and seasonal pattern, time series analysis is reasonable to use for proactive agent placements. However, if an unforeseen event is happens which has never been seen in the historical or seasonal based dataset that used to train the DL model, the prediction accuracy of the agent will significantly affected for the events duration.

A. Time Series Workload and Throughput Forecasting

The two main areas for time series analysis are classical time series forecasting, such as seasonal auto-regressive integrated moving average (SARIMA) and ML/DL such as LSTM. SARIMA forecasting methods is commonly used for uni-variate time series data forecasting which captures the seasonal and trend patterns in the data. However, recently, DL models such as LSTM, GRU, CNN and MLP are being

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.

used for time series forecasting. Such DL models improve the accuracy of prediction. LSTM is best suited for series data as it accurately predicts memorizing the historic data, sequentially capturing it through its structure. LSTM is a very specialized version of RNNs that are able to learn long-term dependencies. Compared to other sequence learning algorithms such as the Markov model, SVR and RNNs, LSTM is relatively insensitive to gap length. Structurally, different versions of LSTM exist. The common type of LSTM has three information regulating gates and cells to store information as depicted. We will briefly describe 4 deep learning model swe used in the next paragraph.

GRU is an improved form of LSTM and was introduced to solve the vanishing gradient problem faced by RNNs. It consists of two gates, update gate and reset gate. The reset gate is like a forget gate in LSTM where it filters out the unnecessary information from the past. Then the update gate decides what information needs to be passed to the output layer. MLP is a classic version of artificial neural networks consisting of at least three layers, an input layer, a hidden layer, and an output layer. Each layer is composed of many perceptrons utilizing a non-linear activation function to learn the non-linear data. CNN is another type of neural network extensively used in image classification applications. In order to process massive images, it leverages a layer called convolution layer reducing the dimension of the input data without losing critical features. This allows the network to be deeper and learn more easily.

B. C-RAN Traffic Generation

In this paper, we focus on C-RAN functions placement that are design as agent. In this case, we have all the services arriving to/from all the three type of cells. The data or workload from mobile end users are generated using equation 2 [19].

$$P_{ue} = (3A + A^2 + \frac{1}{3}MCL)(\frac{R}{10}) \tag{2}$$

where A is the number of antennas, M the modulation bits, C the code rate, L the number of spatial MIMO-layers and R the number of physical resource blocks (PRBs). The processing load Pue is measured in GOPS. The CPU usage requirement to and edge data center for all the mobile users considered in this case is linearly related to the throughput and is given by [20]

In addition to the prediction aggregators' throughput, in this work, we also would like to show how we can predict the workload demand on a data center from the incoming aggregate throughput using equation 3.

$$P_{CPU} = C \times Th(t) + b \tag{3}$$

Where C and b are constants.

IV. A Combined Proactive and Reactive Based Agent Placement

This section presents our proposed approach of combining reactive and proactive decisions for agent placement

A. Proactive Decision

We consider the DL (LSTM, GRU, CNN, and MLP) based agent's prediction result as the proactive decision. We first train the DL models using historical dataset. Using the train DL model as the cognitive component, we design the prediction agent. The prediction agent then predicates the number and type of processing agent that should be instantiated at a given time. The orchestration agent creates the required agent chain. This is only for classic implementation and comparison with the combination algorithms that combines the reactive decision with the predicted proactive values.

B. Proactive and Reactive Decisions with Network State Feedback

Proactive decision based on information extracted form the historically recorded data provides a reasonable estimation accuracy form most data center network and computing resource management decisions. However, due to the random nature of users or service distribution which may cause network traffic spikes, proactive decision could could be sub-optimal decision. Network traffic spikes could be caused by various reasons. These could be because of malware outbreak, unexpected user service demand due to some events in the areas for example a football match or concert. Therefor, a network state feedback is needed to determine the current network conditions to provided the resource demands. The second part of equation 4 is aimed at capturing the current network state through traffic arrival measurement. For autonomic decision accuracy is important factor. Therefore, combining different techniques could be one solution. Moreover proactive decision may not exactly show the current states since there could be sudden change. Reactive decision could be incorporated.

$$N_{Ag} = N_{Ag-DL} + \frac{(\lambda_{ave} - \lambda)T}{Ag_{CPU}} + C \tag{4}$$

or

$$N_{Ag} = \alpha N_{Ag-DL} + (1-\alpha)\frac{\lambda T}{Ag_{CPU}} + C \tag{5}$$

where N_{Ag} is the estimated number of agents to be made available at a given time, N_{Ag-DL} is the estimated number of agents based on historic data for a particular day and hour using a DL model, λ_{ave} is the average workload arrival rate of the edge/cloud data center at a given time, λ is the average workload arrival rate in the previous hours of a particular day of interest, Ag_{CPU} is the amount of a given agnet CPU requirement, T is the duration which could be 60sec/60min depending on selection, α is the decision factor ratio which has a value between 0 and 1 and C is constant for over-provisioning factor for agents.

Here two alternative but similar formulations are provided. The two formulations are similar but they differ from decision combining perspective. Equation 5 determines the proactive decision as the main decision for the number of agent required to be active at a given instance until a the prediction error surpasses some threshold. Whenever, the prediction error surpasses a given threshold, the reactive value will be added to the

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.

DL based agent's prediction result. The second formulation is using the a both the prediction and rate estimation at the same time with some percent of (which is determined by the value of α) proactive decision value and some percent of reactive decision value.

1) Discussion on performance of the method in terms of decision latency, decision reliability, and decision computational efficiency: Since DL training incur latency and computational resources, it needs to be done more efficiently. For example, it could be possible to periodically retrain the DL models of the agents using the updated dataset. For proactive decision to be made periodic updating of the historic data and periodically evaluating it every night or other day or per week depending on the operator cost and traffic dynamics. To gain optimal performance in terms of latency, accuracy, and retraining resource consumption [4], the training period has to be carefully selected. The optimization has to consider the latency as it impacts the end to end delay. The accuracy is dependent on the dataset used to (re-)train the DL model. Since system evolve in time, using the most recent dataset providence better accuracy. However re-training more frequently consumption of computation resource and in cures decision delay by the agent. Rate measurement/estimation feedback for reactive decision is may not take huge computation demand as well as latency to get and finally evaluate the number and amount of resources required.

V. Performance evaluation

In this section we will discuss our final results and simulation scenarios.

A. Dataset Preparation

We generated the dataset from simulation using ns3 developing 5G network based on mm-wave module. In the simulation, we considered the three tier C-RAN architecture with micro, femto, and picco cell. The scenarios we considered is for Manchester city where we considered the number of eNodeBs as 37 and the average fraction of users per tier as 37. The user and eNodeB distributions are considered as poisson distributed and is plotted in figure 4a for 24 hours. Moreover, a Gaussian noise is added to emulate the randomness in the daily distributions [21]. This is plotted in figure 4b for 7 days. A single traffic aggregating getaway is considered to collect the incoming traffic and measure the throughput. Based on this scenario we generated the dataset which is plotted in figure 4b.The generated data which is the throughput for the aggregator have hourly pattern during the day which is repeated in each day. This pattern is directly related to the load demand in the data center as described by equation 2??. Hence, we can consider this demand as a sequential data prediction or regression problem.

B. Model Training

A specialized version of RNN called LSTM is chosen to be appropriate for such type of time series dataset. We treated the problem of predicting the throughput at the traffic aggregating gateway as a linear regression problem where we predict the pattern of the incoming traffic for several weeks. Based on the estimated value, we calculated the CPU workload requirements at the edge data center using equation 2. The final predicted result is linearly related with the total load required for the next few days (per day per hour). In the training procedure, we first prepare the data for the training through several steps using python code with SciPy and keras deep learning library. That means we

We then selected the structure of the DL models to have a single visible input layer, a 2 LSTM blocks or neurons hidden layer, and an output layer for single value prediction.

We choose LSTM, GRU, CNN, and MLP for comparison. LSTM works better on a non-stationary data that shows growing trend and seasonal pattern [22]. Furthermore, even if it is computationally complexity of LSTM could predict for long time more accurately in a changing trend without frequent updating of the data. This is an advantage in the decision process a it reduces the frequency of retraining the LSTM prediction model. This is reduces the overall computational demand in a longer period.

C. Performance Result and Discussion

This subsection presents the performance evaluation result. We have measured and compared DL model accuracy for LSTM,GRU,CNN and MLP based prediction agent as a proactive decision. We then compared them with the combination algorithm based agent prediction result which uses the proactive and reactive decision together.

1) Accuracy of Prediction With and Without Reactive Decision: Figure1 shows a) real traffic versus predicted traffic employing proactive method and b) real traffic versus predicted traffic from our proposed solution(Proactive + Reactive). Figure 2 depicts the RMSE value for different cases. As it can be seen from the figure, the blue bar evaluates the training process and how accurate it is. The proactive decision based result (orange bar) performs better for proactive decision if there is no event that could cause traffic spike. However, when there are events that create unexpected traffic arrival, its performance decreases (green bar). The red bar shows the improvement in case of spike traffic arrivals by incorporating reactive decision to the proactive decision. As shown in the figure for different DL models, LSTM performs better followed by GRU, MLP and CNN, respectively.

The result also varies with the amount of training dataset used.

Figure 3 shows the performance comparison of different proactive decisions while comparing the result against the combined approaches.

As can be seen from the figure the performance result improves as we increase the size of the training dataset. Models trained with large and recent dataset performs better meaning that adding more recent data as it comes from the system improves the accuracy. However, this increases the training frequency impacting the overall latency and resource consumption.

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.

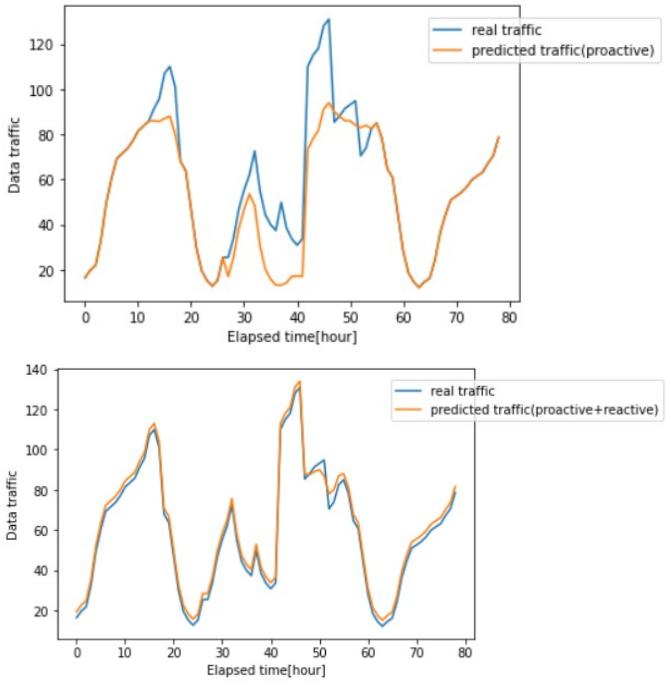


Fig. 1: Real traffic Vs Predicted traffic

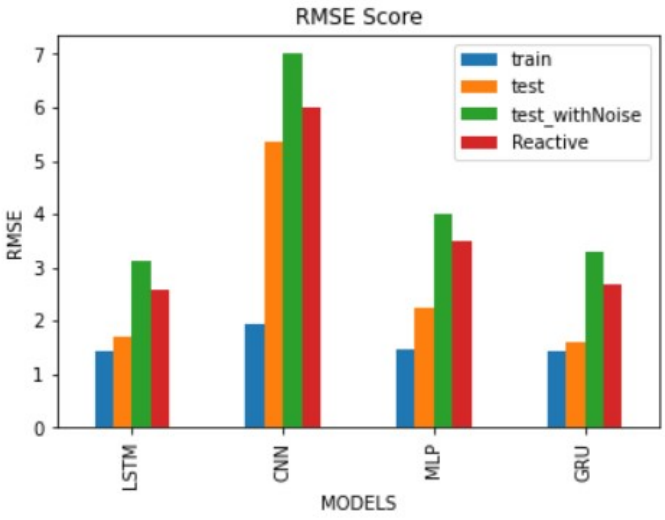
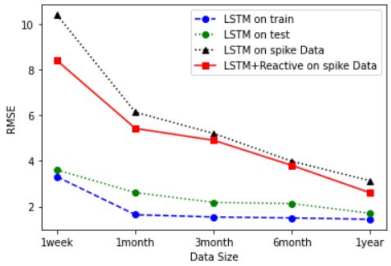
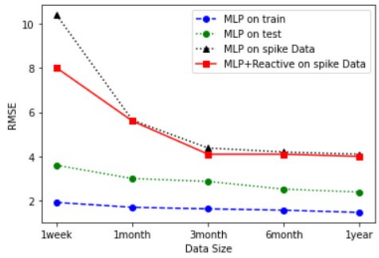


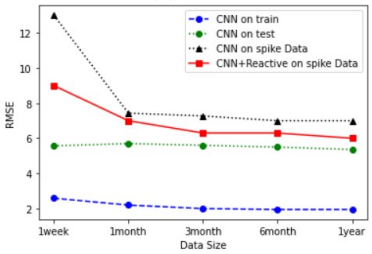
Fig. 2: Shows a) Accuracy of DL Models on training data set (consider as baseline for comparison) and on two different kinds of test data sets; a test set without a traffic spike and a test set with bursty/spike traffic. b) Accuracy of DL models when combined with reactive method .



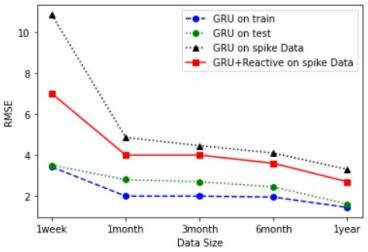
(a) LSTM



(b) MLP



(c) CNN



(d) GRU

Fig. 3: Proactive vs. Reactive on Datasets with/without Event

2) Training Latency Comparison For Proactive Decision: Since training latency impacts the end-to-end delay for workload processing and re-training latency, we evaluate the training latency of the DL models and compare them. Figure 4 illustrates the training latency of LSTM, GRU, CNN and MLP. As shown in the figure, training latency increases as the dataset size grows for different DL models.

We provide the hardware configuration information shown in Table I as it affects the prediction and CPU usage results. Therefore, the same hardware was used to run the LSTM, GRU, CNN, and MLP models.

VI. Conclusion

This paper presented an algorithm that combines proactive and reactive decision to improve the accuracy for workload

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.

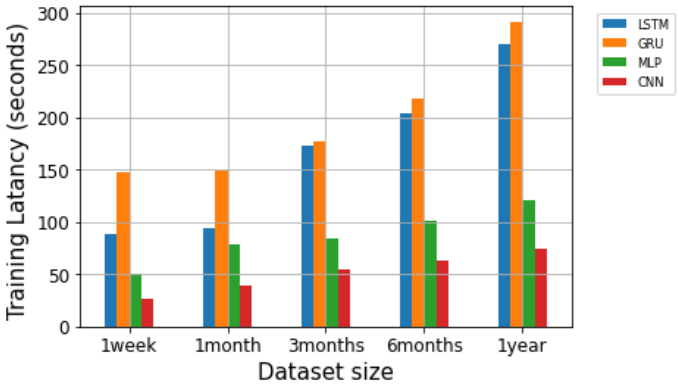


Fig. 4: Comparison of Training Latency for Different Dl Models.

TABLE I: Hardware Configuration

Hardware	Capacity
Memory	16 GB 2133 MHz LPDDR3
Processor	2.7 GHz Intel Core i7
Storage	2 Terabytes

prediction agent. Based on the prediction a agent chain is created to execute the incoming workload as per the requirements. We used LSTM, GRU, CNN, and MLP as a proactive decision model and compared them. We then develop an algorithm that combines the decision of the DL models with reactive decision to improve the performance of the prediction agent. Since accuracy is related to training frequency that impacts the end to end latency of the user and resource consumption edge/cloud data center resources, we have evaluated the training latency and processing workload for a given round of training the DL models. We can see that the proactive decision is well suited for normal traffic. However, the performance degrades if there is unexpected sudden traffic spike. The combined proposed approach improves by providing the agent a shorter window of looking at the most recent event to add reactive decision.

VII. Acknowledgments

This work has been partially funded by the US National Science Foundation under the New Mexico SMART Grid Center - EPSCoR cooperative agreement Grant OIA-1757207.

References

[1] Hexa-X, "D1.2 – Expanded 6G vision, use cases and societal values — including aspects of sustainability, security and spectrum," Apr. 2021. [Online]. Available: https://hexa-x.eu/wp-content/uploads/2021/05/Hexa-X$_D$1.2.*pdf*

[2] S. T. Arzo, R. Bassoli, F. Granelli, and F. H. P. Fitzek, "Multi-agent based autonomic network management architecture," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3595–3618, 2021.

[3] S. T. Arzo, C. Naiga, F. Granelli, R. Bassoli, M. Devetsikiotis, and F. H. P. Fitzek, "A theoretical discussion and survey of network automation for iot: Challenges and opportunity," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 021–12 045, 2021.

[4] C. N. Serugunda, S. T. Arzo, F. Granelli, R. Bassoli, M. Devetsikiotis, and F. H. Fitzek, "Autonomous network traffic classifier agent for autonomic network management system," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[5] D. Cho, J. Taheri, A. Y. Zomaya, and P. Bouvry, "Real-time virtual network function (VNF) migration toward low network latency in cloud environments," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, June 2017, pp. 798–801.

[6] Y. Nakagawa, C. Lee, K. Hyoudou, S. Kobayashi, O. Shiraki, J. Tanaka, and T. Ishihara, "Dynamic virtual network configuration between containers using physical switch functions for nfv infrastructure," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, Nov 2015, pp. 156–162.

[7] W. Zhou, Y. Yang, M. Xu, and H. Chen, "Accommodating dynamic traffic immediately: A vnf placement approach," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[8] S. Schneider, S. Dräxler, and H. Karl, "Trade-offs in dynamic resource allocation in network function virtualization," in *2018 IEEE Globecom Workshops (GC Wkshps)*, Dec 2018, pp. 1–3.

[9] H. Cao, H. Zhu, and L. Yang, "Dynamic embedding and scheduling of service function chains for future sdn/nfv-enabled networks," *IEEE Access*, vol. 7, pp. 39 721–39 730, 2019.

[10] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, "Virtual network function placement in iot network," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, June 2019, pp. 1166–1171.

[11] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2179–2192, Oct 2019.

[12] F. Slim, F. Guillemin, A. Gravey, and Y. Hadjadj-Aoul, "Towards a dynamic adaptive placement of virtual network functions under onap," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2017, pp. 210–215.

[13] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-torch: An automated nfv orchestration and monitoring solution," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1292–1306, Dec 2018.

[14] H. Tang, D. Zhou, and D. Chen, "Dynamic network function instance scaling based on traffic forecasting and vnf placement in operator data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 3, pp. 530–543, March 2019.

[15] A. Bilal, T. Tarik, A. Vajda, and B. Miloud, "Dynamic cloud resource scheduling in virtualized 5g mobile systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[16] H. Kim, D. Lee, S. Jeong, H. Choi, J. Yoo, and J. W. Hong, "Machine learning-based method for prediction of virtual network function resource demands," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, June 2019, pp. 405–413.

[17] S. I. Kim and H. S. Kim, "Method for vnf placement for service function chaining optimized in the nfv environment," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2019, pp. 721–724.

[18] "Aws scaling (reactive vs proactive vs predictive)," [Online]. Available: https://www.gritfeat.com/aws-scaling-reactive-vs-proactive-vs-predictive/.

[19] T. Werthmann, H. Grob-Lipski, S. Scholz, and B. Haberland, "Task assignment strategies for pools of baseband computation units in 4G cellular networks," in *2015 IEEE International Conference on Communication Workshop (ICCW)*, June 2015, pp. 2714–2720.

[20] Y. Luo, S. Huang, J. Chou, and B. Chen, "A computation workload characteristic study of c-ran," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, July 2018, pp. 1599–1603.

[21] M. Laner, P. Svoboda, and M. Rupp, "Modeling randomness in network traffic," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 393–394, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2318857.2254809

[22] Z. Zhao and Y. Zhang, "A traffic flow prediction approach: Lstm with detrending," in *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, Dec 2018, pp. 101–105.

Authorized licensed use limited to: UNIVERSITY OF NEW MEXICO. Downloaded on June 04,2023 at 15:54:58 UTC from IEEE Xplore. Restrictions apply.