

# GenAI Assistance for Deep Reinforcement Learning-based VNF Placement and SFC Provisioning in 5G Cores

Murat Arda Onsu<sup>1</sup>, Poonam Lohan<sup>1</sup>, Burak Kantarci<sup>1</sup>, Emil Janulewicz<sup>2</sup>,

<sup>1</sup>*University of Ottawa, Ottawa, ON, Canada*

<sup>2</sup>*Ciena, 383 Terry Fox Dr, Kanata, ON K2K 2P5, Canada*

<sup>1</sup>{monsu022, ppoonam, burak.kantarci}@uottawa.ca, <sup>2</sup>{ejanulew}@ciena.com

**Abstract**—Virtualization technology, Network Function Virtualization (NFV), gives flexibility to communication and 5G core network technologies for dynamic and efficient resource allocation while reducing the cost and dependability of the physical infrastructure. In the NFV context, Service Function Chain (SFC) refers to the ordered arrangement of various Virtual Network Functions (VNFs). To provide an automated SFC provisioning algorithm that satisfies high demands of SFC requests having ultra-reliable and low latency communication (URLLC) requirements, in the literature, Artificial Intelligence (AI) modules and Deep Reinforcement Learning (DRL) algorithms are investigated in detail. This research proposes a generative Variational Autoencoder (VAE) assisted advanced-DRL module for handling SFC requests in a dynamic environment where network configurations and request amounts can be changed. Using the hybrid approach, including generative VAE and DRL, the algorithm leverages several advantages, such as dimensionality reduction, better generalization on the VAE side, exploration, and trial-error learning from the DRL model. Results show that GenAI-assisted DRL surpasses the state-of-the-art model of DRL in SFC provisioning in terms of SFC acceptance ratio, E2E delay, and throughput maximization.

**Index Terms**—SFC Provisioning, Network Function Virtualization, DRL, Generative AI Approach, Variational Autoencoder

## I. INTRODUCTION

Traditional network services and functions cause long product cycles and low service agility due to their strong dependency on specific hardware. Network Function Virtualization (NFV) brings novelty to communication technologies by decoupling software from physical devices and deploying functions on virtual machines residing on general-purpose hardware, such as data centers (DCs), to reduce OPEX and CAPEX while increasing flexibility [1]. Service Function Chaining (SFC), in the NFV context, can be defined as a sequence of Virtual Network Functions (VNF) to deliver services such as Cloud Gaming (CG), Augmented Reality (AR), Voice over IP (VoIP), Video Streaming (VS), Massive IoT (MIoT), and Industrial 4.0 (Ind 4.0) [2] [3]. SFC provisioning requires satisfying all VNFs in its chain in proper order and establishing a packet transfer. However, SFC provisioning encounters several critical challenges, including efficient resource allocation, sequential execution of VNFs, high demands, and E2E delay limitations.

Machine Learning (ML) and Deep Learning (DL) methods have been highly investigated in the literature to overcome these challenges. Deep Reinforcement Learning (DRL) has made a significant breakthrough and impacted various areas due to its

advanced methodology. Combining both DL and Reinforcement Learning (RL), which provides exploration and trial and error methods by using DL model architecture; DRL has also been used in optimal VNF placement and SFC provisioning in literature [4] [5]. The goal of DRL is to maximize the expected reward by performing actions considering its current state. However, this methodology can struggle at the beginning of the training process or generalization of the environment, which makes it require additional assisted models such as a generative approach [6]. Generative models, such as Variational Autoencoder (VAE) or Generative Adversarial Network (GAN), can assist DRL in numerous ways, such as additional unseen data generation, understanding action-value or state-value distributions, state representations, and so on [7] [8].

In this research, we enhance the model from our prior study [9], which employed a DRL model with multiple input layers and priority points to unlock reconfigurability in SFC provisioning. The current work integrates a generative-assisted approach, deploying a VAE [10], to further refine the model's performance and adaptability. In the previous work, the data center (DC) is selected using the priority points, and selected DC information alongside overall system states is given to the DRL model. Then, the DRL model performs a VNF placement action on the selected DC. However, since the selection of the appropriate DC is solely based on a priority point, other critical aspects like available computational resources, network congestion, and task completion time are often overlooked. This may result in suboptimal DC selection, leading to inefficient resource allocation. Therefore, this study aims to use a VAE-based generative AI-assistance DRL model (GenAI-DRL) where the network and each DC information are given to the generative part, and it calculates the state-value function of each DC; then, the DC with maximum value will be selected for DRL to perform an action. Training of the VAE is performed by the dataset of DCs' initial state and next state after the DRL model's action inspired by [11]. The main contributions of this work are summarized as follows:

- Improving the earlier DRL model with the generative approach with VAE where DCs in the network are selected by their value function created by the generative model.
- Advanced training of VAE by using its current state and next state after the DRL model performs an action.

The dataset is collected during the simulation runtime from previous research's VNF provisioning model and randomly selecting the DC algorithm. After each DC is selected randomly, their next states and values are calculated and pushed forward alongside with current state to the dataset of VAE.

- Detailed analyses are done for E2E latency, SFC acceptance ratio, and overall network throughput.

The rest of the paper is organized as follows. Section II presents related works. Section III describes the system model and problem formulation. The proposed GenAI-assisted DRL model is explained in Section IV. Section V provides numerical results and discussions, and Section VI concludes the paper.

## II. RELATED WORKS

Several studies focus on SFC provisioning and VNF placement in the literature. In study [12], PSVShare is proposed for service placement problems in edge computing considering priority-based, least-cost, and resource-efficient direction. On the other hand, researchers in [13] adapt Q-learning to find the optimal SFC path considering the resource utilization of VNF placement in 5G. DRL algorithm based on Deep-Q-Network (DQN) is utilized in [14] for maximizing QoE while meeting QoS requirements in NFV-enabled networks. The DRL-Based method is also used in [15] for the VNF cooperative scheduling framework with priority-weighted delay to handle the issues of VNF queueing waiting and resource imbalances. Double Deep Q Network-based VNF Placement Algorithm is proposed in [5] for optimal VNF placement, which minimizes the rejection rate and E2E latency while maximizing the throughput.

Furthermore, generative models and robust RL methods have been investigated in the literature for effective resource management. For example, researchers in [7] introduce a GAN-powered Deep Distributional Q-Network (GAN-DDQN) to mitigate randomness in resource allocation by learning the action-value distribution for each service slice in 5G networks. Here, the GAN approximates the distribution using a generative approach to model the randomness and noise embedded in the resource allocation problem, enabling the model to adapt to dynamic environments. Study [8] combines curiosity-driven exploration with the VAE forming a DQN-CVAE model that generates more diverse and high-quality samples. This improves sample efficiency and enhances the training process of the Q-network. Another Approach, The Dreaming Variational Autoencoder (DVAE) presented in [11], generates action-value distributions by encoding state-action pairs into a latent space and decoding them into probable future states. This approach allows the model to predict future states based on past actions and store these predictions in a replay buffer for further training.

While DL and DRL methodologies are explored in the literature for SFC provisioning, generative approaches offer the potential for more robust training and optimization, particularly in complex systems where exploring and training AI agents effectively is challenging. However, the integration of DRL with generative approaches remains underexplored in existing

TABLE I: Service Function Chain (SFC) characteristics in 5G Core Network [2]

SFC Request	VNF Sequence	Bandwidth (Mbps)	E2E delay (msec)	Request Bundle
Cloud Gaming (CG)	NAT-FW-VOC-WO-IDPS	4	80	[40-55]
Augmented Reality (AR)	NAT-FW-TM-VOC-IDPS	100	10	[1-4]
VoIP	NAT-FW-TM-FW-NAT	0.064	100	[100-200]
Video Streaming (VS)	NAT-FW-TM-VOC-IDPS	4	100	[50-100]
MIoT	NAT-FW-IDPS	[1-50]	5	[10-15]
Ind 4.0	NAT-FW	70	8	[1-4]

research. Therefore, this study will investigate GenAI-assisted DRL to enhance the performance of the AI agent.

## III. SYSTEM MODEL AND PROBLEM FORMULATION

Fig.1 illustrates the network environment, including VNF instance (VNFI)-enabled DCs,  $\mathcal{D}$ , logical links between DCs,  $\mathcal{L}$ , and the GenAI-assisted DRL agent for SFC provisioning. Each DC,  $i \in \mathcal{D}$ , has limited computational and storage capacity denoted by  $\mathcal{C}_i$  and  $\mathcal{S}_i$ , respectively. The BW capacity of the logical link between DCs  $i$  and  $j \in \mathcal{D}$  is denoted by  $B_{ij}$  Mbps. The set of supported SFC requests is denoted by  $S = \{\text{CG, AR, VS, VoIP, MIoT, Ind 4.0}\}$ . Each SFC request  $s \in S$  is composed of ordered VNF sequences  $\mathcal{V}^s = (V_1^s \rightarrow V_2^s, \rightarrow \dots V_{N_s}^s)$  and have attributes of bandwidth requirement,  $B^s$  (Mbps), E2E delay,  $D^s$  (msec), requests bundle size,  $\Lambda^s$ , as represented in Table I. To satisfy the SFC request, the AI model should place and process all VNFs to proper DCs as per its chain order before exceeding the E2E delay. The set of all VNFs is denoted by  $\mathcal{V} = \{\text{NAT, FW, VOC, TM, WO, IDPS}\}$ . The placement of each VNF,  $v \in \mathcal{V}$ , requires some storage, computational capacity, and processing time of DC denoted by  $c^v$ ,  $s^v$ , and  $t_p^v$ , respectively. On each DC multiple same and different types of VNFs can be installed while following DC's storage and computational capacity constraints. Let  $x_i^v$  denote the number of VNF  $v \in \mathcal{V}$  installed on DC  $i \in \mathcal{D}$ , then the storage and computational capacity constraints are represented by  $C1 : \sum_{v \in \mathcal{V}} x_i^v s^v \leq \mathcal{S}_i, \forall i \in \mathcal{D}$  and  $C2 : \sum_{v \in \mathcal{V}} x_i^v c^v \leq \mathcal{C}_i, \forall i \in \mathcal{D}$ , respectively. It is assumed that each VNF of an SFC request can be processed by only one DC having that VNF type installed and available on it for allocation till its processing time duration. Let  $\Delta_i^{V_m^s}$  represent the binary variable which is set to 1 if the  $m^{th}$  VNF of SFC  $s \in S$  is allocated at DC  $i \in \mathcal{D}$ , otherwise it is set to 0. With this, the above assumption can be written as the following constraint  $C3 : \sum_{i \in \mathcal{D}} \Delta_i^{V_m^s} \mathbb{1}(x_i^{V_m^s} > 0) = 1, \forall s \in S, \forall m = \{1, 2, \dots, N_s\}$ . The constraint related to logical link BW capacity stating that the BW resources occupied by SFC requests should not exceed the capacity of logical links is represented as follows  $C4 : \sum_{\Lambda_s \forall s \in S} \sum_{m=1}^{N_s-1} \Delta_i^{V_m^s} \Delta_j^{V_{m+1}^s} B^s \leq B_{ij}, \forall i, j \in \mathcal{D}, i \neq j$ . At last, the sum of the processing delay of all VNFs in the chain,  $t_p^s$ , and propagation delay from source to destination,  $t_g^s$  should be less than the E2E delay tolerance limit of the SFC request i.e.  $C5 : t_p^s + t_g^s \leq D^s, \forall s \in S$ . Hence, while fulfilling the above-stated five constraints, the goal of this work is to maximize the acceptance ratio (AccRatio) of all types of SFC

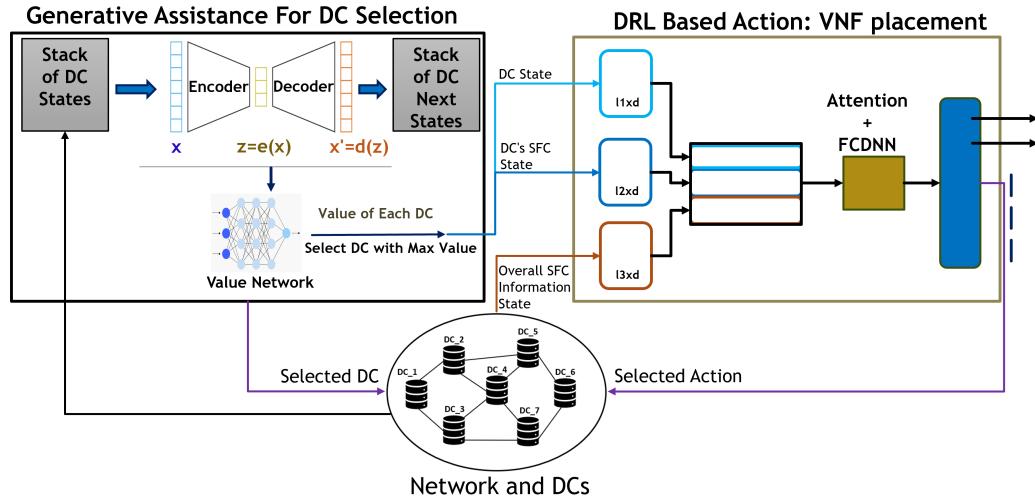


Fig. 1: (GenAI-DRL) System model Illustration with various DCs and GenAI-Assisted DRL Agent

requests received in bundles with optimally selecting the DCs for VNF placement and allocation to SFC requests which can be represented mathematically as follows:

$$(\mathcal{P}) : \underset{\mathbf{x}, \Delta}{\text{maximize}} R_a = \sum_{s \in S} R_s / \sum_{s \in S} \Lambda_s \\ \text{s.t.: } C1, C2, C3, C4, C5.$$

Where  $R_a$  denotes the AccRatio defined by the ratio of the total number of all types of accepted requests,  $R_s \forall s \in S$  to the total number of generated requests of all types of request bundles  $\Lambda_s \forall s \in S$ .

To solve this combinatorial and nonconvex problem, we have proposed a novel GenAI-assisted DRL-based SFC provisioning solution. In the proposed solution, a generative VAE model is used to select the most important DC to perform an action. The selected DC's information, alongside overall network information and SFC requests, is given to the DRL-based VNF placement model, and the DRL agent decides what kind of action to apply on that DC. GenAI model helps the state representation and generalization of each DC and its embedded representation is pushed forwarded to the value network where each DC is assigned a proper importance value. DRL model selects the most important DC considering these values and performs the action on the selected DC. Every time the AI agent takes action, the system model updates its parameters and changes state information.

#### IV. GENAI-ASSISTED DRL BASED SFC PROVISIONING

The proposed GenAI-assisted DRL agent for SFC provisioning is divided into two parts: generative VAE and Deep Q-network (DQN). In this algorithm, GenAI module selects the DC, and the selected DC's information is given to the DQN alongside environmental information for VNF placement. VAE is a GenAI model designed for unsupervised learning tasks, particularly in high-dimensional data contexts. It is a type of autoencoder, but with a probabilistic twist, and composed of two main components: the encoder and the decoder. The

encoder maps input data  $x$  into a latent space represented by a probability distribution, typically a Gaussian distribution. Instead of mapping directly to a fixed latent representation, it learns the parameters of this distribution, such as the mean  $\mu$  and the variance  $\sigma^2$ , capturing uncertainty about the latent variables. The encoded latent representation  $z$  is then sampled from this learned distribution. On the other hand, The decoder reconstructs the original input data from the sampled latent variable  $z$ . It aims to generate data points that are similar to the original input, ensuring that the latent space captures meaningful variations in the data. In this research, DCs' current state in the simulation is given to the VAE, and the model, instead of generating the original input sequence, tries to estimate the DC's next state after the DRL module performs the VNF placement action on each of them, so it helps generalization and reaching the optimum DC.

##### A. Generative AI (GenAI) Assistance Model

The generative model is trained using two key objectives: minimizing the reconstruction loss, which measures how well the generated data matches the next state after DRL module performs an action, and the Kullback–Leibler (KL) divergence between the learned latent distribution and a prior distribution, typically a standard normal distribution. The state of the DCs includes its resource availability, installed VNF functions, and the important SFC features such as how many SFCs of which type have a source on that DC, which SFC has minimum E2E remaining time, BW availability, and whether it is possible to allocate the SFC's VNF on that DC. This information is given to the VAE's encoder, and the decoder part tries to generate the next state if the DRL model performs VNF placement action. Model training is illustrated in Fig.2.

The dataset for generative VAE is collected during the simulation runtime. In that runtime, DC selection is performed on a random basis to improve the diversity and exploration. Then, its information is stored as DC's current state for VAE training. After DC is selected, already trained DRL from previous work

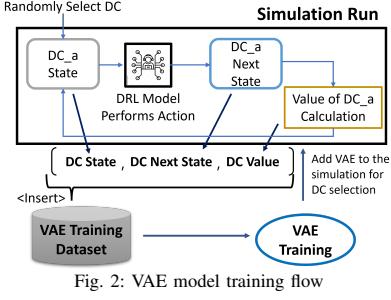


Fig. 2: VAE model training flow

[9] is utilized for selecting the proper actions (VNF allocation, VNF uninstallation, Idle wait), and the environment is updated. Based on these actions, the DC's state is updated, and its value is calculated by taking into account the SFCs that the DRL module aims to satisfy, along with factors such as their source, destination, remaining E2E latency, and the availability of resources and bandwidth. The DC's next state and value, alongside the DC's current state, are added to the dataset, and once the dataset is large and diverse enough, the simulation stops, and the generative VAE model starts training. Training is performed in two phases: VAE training and Value Network Training. In VAE training, the model is fed by the DCs' current state, and it tries to generate the next state while representing them in the embedded space. Upon completion of VAE training, the embedded representations of the DCs, along with their values, are used to train the Value Network.

Once the VAE model and the Value Network have been trained, they are used with DRL for SFC provisioning in the simulation runtime. Overall, DCs' current states are given to the VAE's encoder part, where their embedded representations are created. Then, these embedded representations are forwarded to the Value Network, and each of them is assigned a value that shows DC's importance. The DRL model selects the DC with the maximum value and performs an action on the selected DC.

### B. Deep Reinforcement Learning (DRL) Module

The trained DRL module that performs VNF placement action on the selected DC for SFC provisioning is used from our previous work [9]. A Markov Decision Process (MDP) is utilized for modeling decision-making in situations which has the components of state, action, next state and reward. The main aim of MDP is to find the policy that maximizes the cumulative reward value, and the DQN algorithm is applied to address the MDP. In the advanced DRL technique, DQN, deep learning is used to estimate the optimal action-value function, commonly referred to as the Q-value.

The reward is scalar feedback indicating the agent's performance for each action, where penalties are assigned for suboptimal decisions. An optimal action, like successfully fulfilling an SFC request by placing all its VNFs, results in a positive reward of +2. Conversely, dropping an SFC request incurs a penalty of -1.5, which is slightly less severe than the reward. Additional penalties are given for actions such as uninstalling essential VNFs (-0.5) and selecting invalid action

(-1) that leaves the state unchanged, keeping the model in the same action until the next step. The actions in the DQN module include placing/allocating a VNF, uninstalling a VNF from the selected DC, and idle waiting, which is also considered a valid action. The number of neurons in the output layer of the DRL corresponds to the total number of possible actions and is defined as  $[2 * |\mathcal{V}| + 1]$ , where  $|\mathcal{V}|$  denotes the number of VNF types.

Finally, States represent the information the DRL agent knows about the environment at a given time, and these are given as inputs of the module. DQN module is constructed via a fully connected deep neural network (FCDNN) layers architecture with 3 input layers, of which the first input layer is related to the selected DC's resources and the VNFs installation situation on it, the second layer corresponds to SFC information on that DC, and the third input layer takes the SFC information for the overall network. These layers include different numbers of input neurons, considering the states' features, but the same number of output neurons to generate the output with the same size. Then, the outputs of these layers are concatenated to form one instance forwarded to the attention layer to emphasize the important features. Then, it passes through several hidden layers until reaching the final layer. The number of neurons in the final layer is equal to the number of action types of the DQN, and the model performs an action considering the output value.

### C. SFC Provisioning Algorithm

---

#### Algorithm 1: SFC Provisioning via GenAI-DRL

---

```

Data: Overall_SFCs, Overall_Network
Result: DC_Status, DC_Next_Status, DC_Values
1 Gen_Model  $\leftarrow$  GenAI_Deploying();
2 DQN_SFC_Provisioning_Model  $\leftarrow$  DQN_Model_Deploying();
3 DC_Statuses  $\leftarrow$  [], DC_Next_Statuses  $\leftarrow$  [], DC_Values  $\leftarrow$  [];
4 while Overall_SFCs.SFCs_exists do
5   foreach DC  $\in$  Overall_Network.DCs do
6     | DC_Statuses.append(DC.get_state())
7   end
8   DC_Repr  $\leftarrow$  Gen_Model.VAE.Encode(DC_Statuses);
9   DC_Values  $\leftarrow$  Gen_Model.Value_Network(DC_Repr);
10  DC_Selected  $\leftarrow$  Max_DC(DC_Vals, Overall_Network.DCs);
11  DQN_State  $\leftarrow$  Generate_States(DC_Selected,
12    Overall_Network, Overall_SFCs)
13  DQN_SFC_Provisioning_Model.Action(States_for_DQN)
14  Overall_Network.update();
15  foreach (id, DC)  $\in$  Overall_Network..DCs_with_id do
16    | DC_Next_Statuses.append(DC.get_states());
17    | DC_Values.append(Calculate_Value(DC, DC_Statuses[id]))
18  end
19 end

```

---

In Algorithm 1, overall SFC information and network components such as DC's logical links and resources are given as a parameter, and the algorithm runs until there are no SFC requests in the system. At the beginning of the process, the GenAI model is deployed after being trained in the previous run, and the deployment of the DQN model follows it. In step 4, the simulation starts handling the SFC request, and in the first phase, all DCs' current states are stored in *DC\_Status* (Steps 5-7). These states' information is first sent to the encoder parts of the VAE in the GenAI model (step 8), and their embedded representations are collected. Then, these representations are forwarded to the value network (step 9), and each DC is

assigned a specific value. Afterward, DQN selects the DC with maximum value (step 10) and performs a VNF placement action on it by taking the SFC and source information of the selected DC and overall SFC information (steps 11-13). After the VNF placement is completed, system components such as DC start updating, and new DC state information alongside their calculated values are stored in *DC\_Next\_States* and *DC\_Values* are stored for another training and testing for generative modeling (Steps 14-18). The algorithm is concluded after handling all SFCs in the system.

## V. NUMERICAL RESULTS

### A. Simulation Setup and Training

The simulation model for the SFC provisioning algorithm is developed in Python. The network consists of DCs, logical links, and SFC requests that include specific VNF chains. Each DC is equipped with 2 TB of storage, CPUs ranging from 12 to 120 GHz, and 256 GB of RAM, while the logical links offer 1 Gbps bandwidth. SFC requests are generated by clients as bundled requests (see Table I) and are sent to the network for processing [16]. The simulation generates the six most common SFCs: CG, AR, VoIP, VS, MIoT, and Ind 4.0, as outlined in [2] characterized by specific bandwidth requirements, E2E delay, request bundle ranges, and VNF sequences, as detailed in Table I. The VNF sequences include components such as Network Address Translation (NAT), Firewall (FW), Video Optimization Controller (VOC), Traffic Monitor (TM), WAN Optimizer (WO), and Intrusion Detection Prevention System (IDPS). To satisfy an SFC request, its VNFs must be processed in the proper order. The vCPU, RAM, storage, and processing times for each VNF are sourced from [2]. Efficient VNF placement, installation, and allocation are crucial for meeting the E2E delay constraints of SFC requests and improving network performance, particularly in terms of AccRatio.

Pre-trained DQN is used for SFC provisioning via VNF placements from previous work [9] where the model is trained with 2-4 DCs environment with 350 updates every 20 episodes. For GenAI model training, the simulation runs with pre-trained DQN for SFC provisioning, and DCs are selected randomly during the runtime. After each action of DQN, DC's next states, alongside the current state and calculated value, are added to the dataset, and generative model training is performed by using this dataset. SFC requests are generated randomly within their bundle sizes (see Table I). A request is successfully fulfilled if all VNFs in its chain are processed within the required time; otherwise, it is dropped due to resource constraints or failure to meet E2E delay requirements.

### B. Testing and Results

To measure the proposed approach's performance and superiority over the previous methods and benchmark algorithm, the following metrics have been utilized: E2E delay, throughput, overall SFC AccRatio, and AccRatio of each SFC type under different network configurations with different amounts of demands. In Fig.3, the left diagram compares the performance

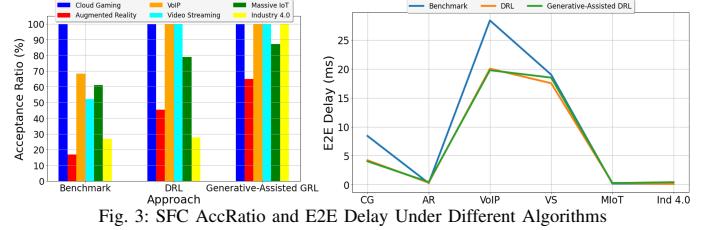


Fig. 3: SFC AccRatio and E2E Delay Under Different Algorithms

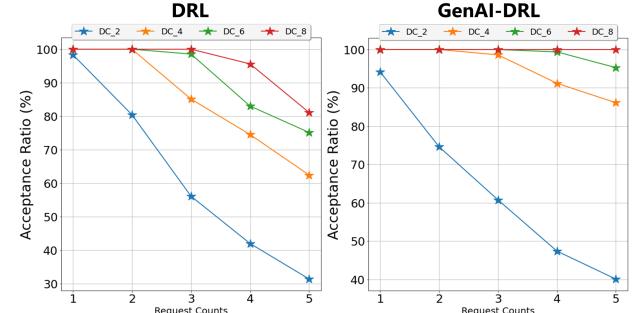


Fig. 4: GenAI-DRL and DRL Comparison of SFC AccRatio Under Different DCs Number and Request Counts

of the proposed Gen-AI DRL algorithm with the baseline rule-based heuristic approach and DRL in terms of AccRatio with 6 DC environment. Although the comparison is percentage-based, each SFC type has a different bundle size, as outlined in Table I. According to the results, the heuristic-based approach achieves minimum AccRatio, while it is followed by DRL and the proposed GenAI-DRL algorithm. VS and VoIP requests are not satisfied completely only in a benchmark, with 53% and 60%, while the Ind 4.0 satisfaction rate is below 30% for both DRL and heuristic approaches. AR reaches minimum AccRatio in heuristic approached with around 18% due to strict E2E delay limits, while it is increased to 45% and 64% in DRL and GenAI-DRL algorithm. The proposed algorithm 100% satisfies CG, VoIP, VS, and Ind 4.0 while MIoT AccRatio is 88%. Moreover, CG has the highest AccRatio for all algorithms due to its 80 ms delay tolerance, which is lower than that of VS and VoIP (100 ms), making low E2E delay tolerant requests prioritized. On the other hand, the right part of the figures compares the E2E latency of this runtime, and AR, VS, MIoT, and Ind 4.0 have the minimum E2E latency for all algorithms since having minimum E2E delay limits makes them be processed immediately. The benchmark model reaches maximum latency. Though the DRL model and GenAI-DRL model's E2E latency are almost similar and sometimes superior to each other, it must be noted that E2E delay results are calculated considering the accepted SFC requests, and dropped SFCs are not used in this calculation.

SFC AccRatio and throughput comparison for DRL and GenAI-DRL models are given in Figs. 4 and 5. Both figures show the performance comparison in various numbers of DCs, 2, 4, 6, and 8 and various request counts from 1 to 5. Note that each request count consists of all types of SFCs requests in bundles as per the size in Table I.

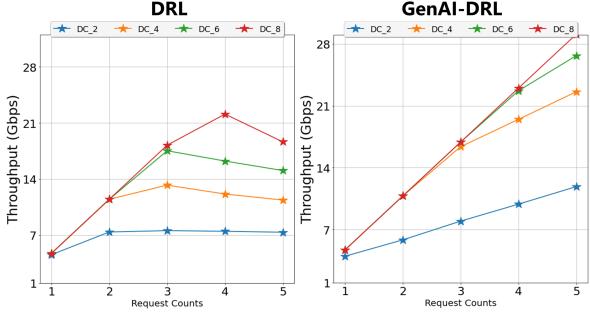


Fig. 5: GenAI-DRL and DRL Comparison of Throughput Under Different DCs Number and Request Counts

In Fig.4, the left part shows the SFC AccRatio of the DRL model, while the right part depicts the GenAI results. Though the DRL model can maintain its performance with maximum efficiency in 8 DCs till receiving 3 request counts, in other network configurations, there are some performance reductions once receiving more SFC requests. DRL model achieves 81% of AccRatio in 5 SFC request counts with 8 DC while this score reduces to 76%, 62%, and 31.5% in DCs 6, 4, and 2 environments, respectively. On the other hand, GenAI-DRL maintains its maximum performance for all request counts in 8 DC environments, and the other configurations, having 6, 4, and 2 DCs, achieve higher scores compared to the DRL model, which are 96%, 87%, and 40% for request count 5. Though the proposed method's initial performance for DC 2 with slightly lower for request number 1 compared to DRL, it maintains higher performance in greater requests due to its planning ability over more massive demands.

In Fig.5, the left part shows the network throughput (Gbps) of the DRL model, while the right part illustrates the GenAI results. For DRL model, throughput increases until request count 4 in 8 DCs case. In 6, 4, and 2 DCs cases, since the AccRatio of Ind 4.0, AR, and MIoT, which require more bandwidth, are low for request counts 4 and 5, maximum throughput is achieved at request count 3. On the other hand, the throughput of the proposed GenAI-DRL model keeps increasing in every configuration, and the maximum is achieved with more than 28 Gbps with 8 DC, while the minimum is around 10 Gbps for request number 5. Though SFC request rate reduction increases in higher request count as shown in Fig.4, the main reason it doesn't impact the throughput dramatically is that the proposed method effectively plans to satisfy the SFCs requests with high bandwidth requirements.

## VI. CONCLUSIONS

This research has explored the assistance of generative AI and its benefit to DRL in handling SFC requests under different network configurations with massive amounts of demands. A custom simulation with various DCs, connections, and SFC requests was created to collect data and train the generative model, and the trained model was used alongside the DRL model to satisfy SFC demands while the generative model handles DC selection for DRL model action. With the better

planning and action selection, the proposed method GenAI-DRL outperforms DRL and benchmark heuristic models in terms of SFC AccRatio, E2E latency and throughput. The ongoing work focuses on the scalability and SFC provisioning under a larger network.

## ACKNOWLEDGMENT

This work is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Alliance Program, in part by the MITACS Accelerate Program, and in part by the NSERC CREATE TRAVERSAL program.

## REFERENCES

- [1] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.
- [2] J. M. Zialet, B. Jaumard, H. Duong, P. Khoshabi, and E. Janulewicz, "A dynamic traffic generator for elastic 5g network slicing," in *2022 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 2022, pp. 1–6.
- [3] M. Setayesh and V. W. Wong, "Service function chain reconfiguration in 5g core networks using deep learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.
- [4] T. D. Tran, B. Jaumard, Q. H. Duong, and K.-K. Nguyen, "5g service function chain provisioning: A deep reinforcement learning-based framework," *IEEE Transactions on Network and Service Management*, 2024.
- [5] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal vnf placement via deep reinforcement learning in sdn/nfv-enabled networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 263–278, 2019.
- [6] M. Janner, *Deep generative models for decision-making and control*. University of California, Berkeley, 2023.
- [7] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.
- [8] G.-J. Han, X.-F. Zhang, H. Wang, and C.-G. Mao, "Curiosity-driven variational autoencoder for deep q network," in *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*. Springer, 2020, pp. 764–775.
- [9] M. A. Onsu, P. Lohan, B. Kantarci, E. Janulewicz, and S. Slobodrian, "Unlocking reconfigurability for deep reinforcement learning in sfc provisioning," *IEEE Networking Letters*, 2024.
- [10] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto, "Variational autoencoder," in *Variational Methods for Machine Learning with Applications to Deep Networks*. Springer, 2021, pp. 111–149.
- [11] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "The dreaming variational autoencoder for reinforcement learning environments," in *Artificial Intelligence XXXV: 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, December 11–13, 2018, Proceedings 38*. Springer, 2018, pp. 143–155.
- [12] A. Mohamad and H. S. Hassanein, "Psvshare: A priority-based sfc placement with vnf sharing," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 25–30.
- [13] D. Lee, J.-H. Yoo, and J. W.-K. Hong, "Q-learning based service function chaining using vnf resource-aware reward model," in *21st Asia-Pacific Network Operations and Management Symp.* IEEE, 2020, pp. 279–282.
- [14] W. Taktak, M. Escheikh, and K. Barkaoui, "Drl approach for online user-centric qos-aware sfc embedding with dynamic vnf placement," *Computer Networks*, vol. 251, p. 110637, 2024.
- [15] J. Yao, J. Wang, C. Wang, and C. Yan, "Drl-based vnf cooperative scheduling framework with priority-weighted delay," *IEEE Transactions on Mobile Computing*, 2024.
- [16] M. A. Onsu, P. Lohan, B. Kantarci, E. Janulewicz, and S. Slobodrian, "A new realistic platform for benchmarking and performance evaluation of drl-driven and reconfigurable sfc provisioning solutions," *arXiv preprint arXiv:2406.10356*, 2024.