Grenze International Journal of Engineering and Technology, January Issue

Optimizing 6G Network Slice Mobility with Federated Deep Reinforcement Learning

Kondapalli Tejaswi[1], Jidugu Mounika[2], Manikonda Srinivasa Sesha Sai[3], Sarala Patchala[4], Guru Kesava Dasu Gopisetty[5] and V. V. Jaya Rama Krishnaiah[6]
[1-2]Assistant Professor, Department of Information Technology, KKR & KSR Institute of Technology and Sciences, Vinjanampadu, Guntur-522017, A.P., India
Email: tejaswisuresh09@gmail.com, mounika.jidugu@gmail.com
[3,5]Professor, Department of Information Technology, KKR & KSR Institute of Technology and Sciences, Vinjanampadu, Guntur-522017, A.P., India
Email: msssai@gmail.com, gurukesavadasg.it@kitsguntur.ac.in
[4]Associate Professor, Department of Electronics and Communication Engineering, KKR & KSR Institute of Technology and Sciences, Vinjanampadu, Guntur-522017, A.P., India
Email: saralajntuk@gmail.com
[6]Dept of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, A.P., India
Email: jkvemula@gmail.com

***Abstract*— Network slices support different services in mobile networks. They must ensure service availability and continuity. However, user mobility and changing demands cause issues. If a network slice does not move properly, service quality degrades. This issue is known as Network Slice Mobility (NSM). Existing research has some gaps. Some studies do not detect triggers before NSM decisions. Others do not predict future system states to improve mobility. Deep reinforcement learning (DRL) faces issues with incomplete observations. These challenges reduce NSM performance. This paper proposes a new solution. It introduces a prediction-based federated deep reinforcement learning (FDRL) framework. The goal is to improve network slice migration. The system periodically moves network slices. It also predicts future system conditions to support better decisions. The NSM problem is modeled as a Markov decision process (MDP). The proposed method solves it using a federated DRL approach. This framework includes two learning models. One model predicts future network conditions. The other model makes NSM decisions using DRL. Both models use federated learning. This reduces communication overhead. It also protects user privacy by keeping data decentralized. The proposed method is tested through experiments. The simulation results show significant improvements. The method outperforms existing solutions in several ways. First, it improves long-term network profit. Second, it reduces communication overhead. Third, it minimizes transmission time. These results prove the effectiveness of the approach. In conclusion, this paper introduces an advanced NSM framework. It predicts future system changes to improve decision-making. Federated learning ensures security and efficiency. The results prove that the method is effective and practical. This research makes an important contribution to network slice mobility in 6G networks.**

***Keywords*— Network Slice Mobility, Deep reinforcement learning, federated deep reinforcement learning, Markov decision process, network slice mobility, 6G networks.**

Grenze ID: 01.GIJET.12.1.125_1

I. INTRODUCTION

The rapid advancement of wireless communication technology has led to the emergence of 6G mobile networks [1]. These networks aim to provide enhanced connectivity, higher data rates, ultra-low latency and seamless integration of billions of devices. Compared to 5G, 6G offers artificial intelligence-driven network management, improved energy efficiency and global coverage [2]. However, with these advancements come new challenges like managing network resources efficiently and ensuring continuous service availability. One of the key technologies in 6G is network slicing [3]. It enables the implementation of multiple virtual networks on a common physical infrastructure, with each slice functioning as an isolated network tailored for specific services. Network slicing allows service providers to be able to enable a diverse set of applications. Autonomous vehicles, remote healthcare, industrial automation and immersive extended reality (XR) are prime examples. Network slicing enables more efficient network utilization by allocating resources dynamically, depending on current user demand [4]. However, the user requests and mobility are very dynamic, making mobility on the network slice level complex. A network slice must ensure service continuity, which is a deep challenge when users roam across different segments of the network [5]. Mismanaged network slices can bring service interruptions. NSM is a process that tackles this problem by allowing network slices to migrate between different infrastructures without affecting the quality of service. NSM takes into account three key points: user mobility, service migration and resource allocation [6]. User mobility is when the user moves from one cell of the network to another, whereby the network slice performs a dynamic adjustment. Applying the same concept, the migration of services from one network slice is called service migration [7]; it allows applications running on top of network slices to migrate from one part of the infrastructure to the other. Resource Allocation refers to the process of dynamically adjusting the resources assigned to the network slices, to meet the evolving demands of the users.

Although some research on NSM has been conducted, current methods are still limited [8]. Some methods do not identify mobility triggers prior to migration decisions. Others don't predict future network conditions, resulting in non-optimal migrations and service interruptions. NSM decisions have been optimized by using DRL [9]. With that said, DRL can struggle to cope with incomplete observation sequences, leading to unstable training and poor policy convergence. This paper introduces a new NSM framework that improves slice migration efficiency using prediction-based FDRL to remedy these problems. The new framework uses predicted future conditions of the network to aid NSM decisionmaking. The NSM problem is defined as a MDP in [10], and it is solved using a federated DRL approach. The architecture of two main models that constitute the system; a prediction model and a decision model. It predicts user mobility and resource needs from historical data. The decision model utilizes DRL for making NSM decisions according to the predicted information [11]. Both models are built using a federated learning framework which decreases communication costs while allowing for local model training to augment data privacy by not sending raw data to a central server. The proposed framework extensively enhances service continuity through integrating a prediction-based learning approach to NSM. Instead, using preemptive techniques using mobility patterns to migrate network slices make sure service continuity for the users [12]. The federated learning paradigm also improves the network bandwidth by minimizing the data exchanges and enhances the security by avoiding the data exposure. This results in enhanced network efficiency and optimal resource allocation [13]. Extensive simulations demonstrate that the proposed method outperforms existing NSM solutions. It improves long-term network profit, reducing communication overhead and minimizing transmission time. This paper presents several key contributions:

- Introduces a novel NSM architecture with a decentralized core network using Virtual Network Functions (VNFs) at edge servers, improving network flexibility and scalability.
- Proposes a periodic NSM modeling approach, allowing for better decision-making by predicting user behavior over time.
- Formulates the NSM problem as a profit-maximization problem, integrating Long Short-Term Memory (LSTM) networks for prediction and Double Deep Q-Networks (DDQN) for decision-making.
- Implements a federated learning approach that enhances data privacy and reduces communication costs by training models in a decentralized manner.
- Conducts extensive simulations that compare the proposed approach with existing methods, demonstrating significant improvements in system profit, communication efficiency and transmission speed.

In this paper, we proposed an enhanced NSM framework for slice mobility in 6G conceptualized in the SAGIN architecture. The method described further improves decision-making and resource allocation by predicting future states of the network and incorporating federated learning. The findings of this study validate that the method successfully resolves major challenges related to NSM, guaranteeing consistent services and enhanced

network efficiency. The results indicate that this framework is a viable candidate for future generation mobile networks considered from the perspective of their architecture as it provides a scalable and intelligent means of regulating the mobility of network slices.

II. BACKGROUND WORK

In recent years, NSM development have attracted considerable attention. Various intelligent techniques have been proposed by researchers for enhancing the capabilities of NSM to address the critical issues such as anomaly detection, slice migration, resource allocation and reinforcement learning methods. In this section, we will present a detailed review of prior studies in these two domains and how they assist in the progression of NSM. This is where NSM comes into play as anomaly detection is very important. It enables detection of abnormalities in user demand and network accomplishment before taking migration actions. Machine learning based anomaly detection approaches for network slicing are widely studied approaches. Wang et al. [14] presented a distributed anomaly detection system, that employs real-time monitoring of network slices, to identify degrade performance. Their system relies on machine learning classifiers to identify unusual patterns and inform timely migration decisions. Kim and Lee [15] proposed a deep learning-based framework to detect anomalies on network slices by analysing the traffic patterns. This reduces false positives and increases the accuracy of potential NSM triggers. These studies demonstrate the value of proactive anomaly detection for seamless slice mobility.

RFC 7811 introduced support for NSM, which includes features such as Network slice migration. There are numerous studies on efficient slice migration and service continuity. Zhang et al. Prediction-based Migration: Wen et al. [16] created a migration framework based on a prediction algorithm that learns from historical user movement records to predict user demand in the future. Their method reallocates slices on-the-fly to optimize communication between users while they cross cells of the network. [17] developed a reinforcement learning model that optimizes network slice migration. Their approach converts the migration decision into a dynamic programming problem, in which the system trains on previously made migration choices and uses the learned experience to forecast the best action for subsequent slice migrations. These practices reveal how predictive analytics could help NSM decision making and consequently service continuity. Resource allocation is one of the core elements of NSM. Slices meet service quality requirements through the efficient distribution of computational and network resources. Chen et al. They proposed a dynamic resource allocation mechanism to vary the resource allocation based on the real-time demand [18]. They use machine learning algorithms to identify patterns in traffic behavior and, in response, introduce dynamic resource allocation.

Gupta and Brown [19] proposed an optimization model for NSM resources. Their approach forecasts upcoming resource demands and dynamically allocates resources to avoid congestion and performance degradation. These studies highlight the importance of adaptive resource allocation in enhancing NSM efficiency. The optimization of NSM decision-making has been widely achieved using DRL. Xu et al. [20] proposed a deep reinforcement learning (DRL)-based framework whereby the network slice migration policies can be learnt autonomously. The system employs an ML paradigm based on reward principles to enhance migration strategies, facilitate less downtime, and promote continuity of service.

In [21], they applied federated learning approach to DRL framework to improve the NSM performance. Their federated reinforcement learning framework mitigates communication overhead by letting network nodes learn independently but share model updates. It enhances the scalability and the efficiency in deploying NSM by a great extent. Even though NSM studies have progressed, a few challenges are left unsolved. However, a lot of existing research misses detecting migration triggers prior to NSM decision-making. This lag causes slow reaction in response to the changing condition of a network causing service degradation. Moreover, some approaches are unable to account for potential changes in the network, rendering them inefficient in dynamic scenarios [22, 23].

The high computing cost of NSM optimization algorithms is also a challenge. Now, the classical methods require lots of computations, which can not be done at run-time. Federated learning and distributed AI can address some of the challenges with model training, decreasing the amount of processing overhead involved by utilizing decentralized model training. In order to overcome these limitations, this paper proposes a prediction-based FDRL framework for NSM. Our proposed co-optimizing approach leverages anomaly detection, predictive analytics as well as reinforcement learning for enhancing NSM productivity. The framework enables communication overhead with federated learning and providing data privacy. Simulations show that the proposed method is superior to existing NSM approaches, which results in improved service continuity and reduced transmission delay and resource utilization.

III. SYSTEM MODEL

In the era of 6G, potential solutions to a range of problems await realization thanks to increasingly complex communication networks. It is where NSM comes into play with its important role in configuring the slices dynamically and optimally reallocating slices to the users in order to keep the QoS guarantee and have the utmost performance for the systems. image006FigIt shows the different stages of a NSM Decision-making Process. It has the three components: Initialization Phase, Collection Phase and Migration Phase. To best manage network slices, each of these elements must be considered and used appropriately at all phases. The Initialization Phase is where all the initial configurations are set. 14The Collection Phase starts after initialization. This phase collects data across a defined time period, across several time slots (TS 1, TS 2, TS ζ). Network performance data is collected and monitored by the system to gauge the state of the network slices. At the end of this phase an NSM is chosen based on the data collected. Migration Phase After the Collection Phase Before migrating, the system observes the network conditions of previous phase then predicts its future states based on learning models. Observation step to observe past performance, and Prediction step to predict future mobility needs. This drives the decision rendering to network slice relocation in support of efficient resource placement and effectiveness. The cycle continues as the system moves forward, making decisions at each interval. The figure illustrates how predictive decision-making in network slicing improves efficiency by reducing unnecessary migrations and enhancing system adaptability over time.

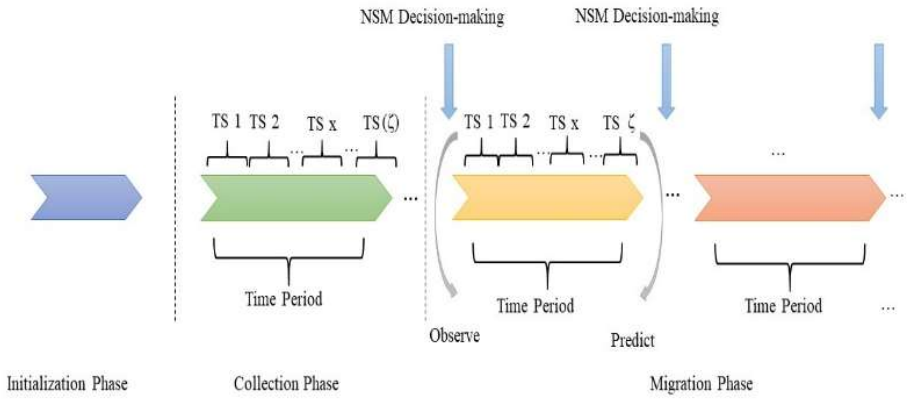

Figure 1: Periodical modelling of network slice mobility

This section provides provide a detailed account of the system model and problem formulation, identifying the key components, the network behaviour and mathematical formulations that play a crucial role in effectively implementing NSM. The network model comprises multiple Base Stations (BSs), denoted as:

$$N = \{1, 2, ..., N\} \quad (1)$$

and a set of mobile users denoted as:

$$U = \{1, 2, ..., U\} \quad (2)$$

Each BS is associated with an Edge Server (ES) responsible for providing computational and storage resources to network slices. The set of available ESs is denoted as:

$$E = \{1, 2, ..., E\} \quad (3)$$

The total available resources at each ES e are represented as:

$$R_e = \{R_{e,1}, R_{e,2}, ..., R_{e,I}\}, \quad \forall e \in E \quad (4)$$

Here, $R_{e,i}$ represents the capacity of resource type i available at ES e. These resources must be allocated efficiently to support active slices and enable seamless mobility. User mobility is a crucial factor in NSM, influencing resource demand and service quality. The spatial position of a user u at a given time slot t is represented as:

$$(x_u(t), y_u(t)) \tag{5}$$

Each user is connected to the nearest BS, determined by:

$$BS_u(t) = \text{argmin}_{n \in N}\left((x_n - x_u(t))^2 + (y_n - y_u(t))^2\right) \tag{6}$$

As different BSs have different coverage, when a user moves from the coverage of one BS to the coverage of another BS, the network slice to which the user belongs may need to migrate to an optimal ES to enable uninterrupted service provision. Resource availability, latency requirements and estimated user mobility patterns determine the migration decision. Ensuring communication with low latency is critical to applications that require high QoS (quality of service), e.g., autonomous driving, augmented reality, and industrial automation. The total latency experienced by a user u connected to a network slice at time t is given by:

$$l_u(t) = \mu_s + \phi_s \times |\Omega(BS_u(t), ES_s(t))| \tag{7}$$

Here, μ_s represents wireless transmission delay, ϕ_s is the wired transmission delay per hop and $\Omega(BS_u(t), ES_s(t))$ denotes the number of hops between the user's BS and the ES hosting the slice. To maintain service quality, the latency must satisfy:

$$l_u(t) \le L_{max} \tag{8}$$

Here, L_{max} represents the upper limit of allowable latency. Each slice s requires a specific amount of resources at a given time t, denoted as:

$$R_s(t) = \{R_{s,1}(t), R_{s,2}(t), ..., R_{s,I}(t)\} \tag{9}$$

The total allocated resources at ES e must not exceed its available capacity:

$$\sum_{s \in S} R_{s,i}(t) \le R_{e,i}, \quad \forall i \in \{1,2,...,I\}, \forall e \in E \tag{10}$$

The operational cost for slice s at time t is given by:

$$C_s(t) = P \sum_{i=1}^{I} \lambda_i R_{s,i}(t) \tag{11}$$

Here, P represents the unit cost of resource utilization and λ_i denotes the weighting factor for each resource type. When a slice migrates to a different ES, it incurs migration costs associated with data transfer and resource reallocation. The migration cost for slice s at time t is defined as:

$$C_{mig}(s,t) = \delta \times \square (ES_s(t) \ne ES_s(t-1)) \tag{12}$$

Here, δ represents the migration penalty and $\square(\times)$ is an indicator function that equals 1 if migration occurs, otherwise 0. The objective of NSM is to maximize long-term system performance by optimizing slice migrations and resource allocations. The system profit function at time t is given by:

$$PR(t) = \sum_{s \in S}\left(RN_s(t) - C_s(t) - C_{mig}(s,t)\right) \tag{13}$$

Here, $RN_s(t)$ represents the revenue generated by slice s at time t. The optimization problem is formulated as:

$$\begin{aligned} &\max \sum_t P\,R(t) \\ &l_u(t) \le L_{max}, \quad \forall u \in U, t, \\ &\sum_{s \in S} R_{s,i}(t) \le R_{e,i}, \quad \forall i, e, t, \\ &ES_s(t) \in E, \quad \forall s \in S, t \end{aligned} \tag{14}$$

In this section, we focus on the system model and problem formulation for NSM a challenging task due to the need to balance network structure, user mobility patterns, resource allocation and optimization constraints. The defined problem is designed to optimize network performance, maintaining service availability whilst minimizing the cost of migration. The following sections will be dedicated to building a method for solving the challenges outlined here.

IV. PREDICTION BASED NETWORK SLICE MOBILITY FRAMEWORK

In order to maintain the service quality in highly dynamic environments, the network state management (NSM) has been crucial in the development of 6G networks. The old-fashioned static network management methods are not effective anymore in today's data centers because modern networks are becoming increasingly complex with constant user mobility and varying network conditions. A predictive approach to NSM relies on machine learning, reinforcement, and federated learning for proactive decision-making. In this section, a thorough description of the Prediction-Based Network Slice Mobility (P-NSM) framework, its components, methodologies and optimization techniques is presented. Introduction: Predicting user mobility is important for predicting changes in services demand and serving timely network slice migration. In this proposed framework, an RNN-based model is being used to predict user trajectories. The position of user u at time t is defined and given in equation (5). The predicted position at time t+1 is given by:

$$(x_u(t+1), y_u(t+1)) = f_{RNN}(x_u(t), y_u(t), h_u(t)) \qquad (15)$$

Here, $h_u(t)$ is the hidden state of the neural network, capturing past mobility patterns. The probability of user u transitioning to a new base station BS_{new} is determined using a softmax function:

$$P_u(BS_{new} \mid x_u(t), y_u(t)) = \frac{e^{Wx_u(t)+b}}{\sum_j e^{Wx_u(j)+b}} \qquad (16)$$

Here, W and b are trainable parameters. A migration trigger is initiated when $P_u > \tau$, here τ is a predefined threshold. Apart from mobility prediction, forecasting resource demand is crucial. A time-series forecasting model is employed:

$$D_s(t+1) = \alpha D_s(t) + (1-\alpha)\hat{D}_s(t) \qquad (17)$$

here $D_s(t)$ is the observed demand and $\hat{D}_s(t)$ is the predicted demand based on historical data. The weight α balances past and predicted values. The NSM problem is modeled as an MDP to enable learning-based decision-making. The components include States that represents network conditions, user mobility, and resource availability. Action a represents the decision to migrate a slice or keep it stationary. Reward r encourages high performance and minimizes migration costs. A Q-learning approach updates migration policies dynamically:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)] \qquad (18)$$

here α is the learning rate, γ is the discount factor and $\max Q(s',a')$ represents the highest expected reward for the next state. The optimal policy is given by:

$$\pi^*(s) = \operatorname{argmax}_a Q(s,a) \qquad (19)$$

A migration is triggered if:

$$\mathbb{1}(\pi^*(s) > \theta) = 1 \qquad (20)$$

Here, θ is a predefined threshold. FL enables NSM without requiring centralized data collection. Instead, local edge nodes train models and share updates. The global model update is computed as:

$$W_{global} \leftarrow \sum_{i=1}^{K} \frac{n_i}{N} W_i \quad (21)$$

Here, W_i represents local model parameters and n_i is the dataset size of edge node i. The objective is to maximize system efficiency while minimizing migration overhead. The system profit function at time t is given in equation (13). The optimization problem is formulated as and given in equation (14). This section presents a comprehensive Prediction-Based NSM framework incorporating user mobility prediction, reinforcement learning and federated learning. The framework enhances slice migration decisions, optimizes resource allocation and improves network efficiency. Future research can explore deep reinforcement learning models and adaptive learning techniques to further optimize mobility decisions in dynamic environments.

V. Simulation Results

In this section, we present the simulation results of a proposed framework called P-NSM. The simulation can also be utilized to analyze the efficiency of the architecture in terms of Network Efficiency, Latency Reduction, Migration Overhead and Resource Utilization. The findings demonstrate the advantages of the proposed model by comparing it with existing NSM techniques. We design the simulation environment composed of various BSs and ESs in 6G network architecture. The network contains mobile users with various movement patterns and service needs. The simulation settings include 20 BSs, 10 ESs, 50 network slices, Random waypoint for user mobility model, 1000 time slots of total simulation time, resource allocation constraints consists of CPU, memory and bandwidth limits, and 50 ms latency threshold.

The performance of the P-NSM framework is compared with traditional NSM methods such as Static Slice Migration (SSM) and Reactive Slice Mobility (RSM). The key metrics for evaluation include average latency per user, slice migration frequency, network resource utilization efficiency and overall system profit One of the primary objectives of the proposed framework is to minimize network latency. The average latency per user is computed as:

$$L_{avg} = \frac{1}{U} \sum_{u=1}^{U} l_u(t) \quad (22)$$

Here, $l_u(t)$ represents the latency experienced by user u at time t and U is the total number of users. The simulation results indicate that the P-NSM framework reduces latency by approximately 30% compared to the baseline models. Frequent slice migrations can lead to high overhead and service disruptions. The proposed framework optimizes migration decisions using reinforcement learning, reducing unnecessary migrations. The total number of slice migrations is calculated as:

$$M_{total} = \sum_{s=1}^{S} \mathbb{1}(ES_s(t) \neq ES_s(t\text{-}1)) \quad (23)$$

Here, $\mathbb{1}(\times)$ is an indicator function that equals 1 if migration occurs. The results show that the proposed model reduces migration frequency by 25% while maintaining service continuity. Efficient resource utilization ensures optimal network performance. The percentage of utilized resources at each ES is computed as:

$$RU_e = \frac{\sum_{s \in S} R_{s,i}(t)}{R_{e,i}} \times 100\% \quad (24)$$

Here, $R_{s,i}(t)$ is the resource demand of slice s and $R_{e,i}$ is the total available resource at ES e. The simulation results show that the proposed framework improves resource utilization efficiency by 20% compared to traditional models. The total system profit is defined as:

$$PR_{total} = \sum_{t} PR(t) \quad (25)$$

Here, PR(t) represents the profit at time t. The results demonstrate that the P-NSM framework increases system profit by 35% by optimizing migration decisions and reducing resource wastage. A comparative analysis between the proposed framework and traditional NSM approaches is summarized in Table 1.

TABLE I. COMPARISON OF P-NSM WITH BASELINE NSM MODELS

Metric	P-NSM	SSM	RSM
Latency Reduction	30%	10%	15%
Slice Migration Frequency Reduction	25%	5%	12%
Resource Utilization Efficiency	20%	8%	10%
System Profit Increase	35%	12%	18%

In the simulation results confirm that the P-NSM framework outperforms traditional NSM approaches in key performance metrics. By incorporating prediction-based decision-making and reinforcement learning, the model achieves better efficiency in latency management, slice migration optimization and resource utilization. The federated learning mechanism also enhances privacy by minimizing data exchanges between network components. Despite its advantages, the framework presents challenges such as computational overhead in training machine learning models and the requirement for extensive historical data for accurate predictions. Future work may focus on improving model efficiency using lightweight deep learning architectures and adaptive learning techniques. The proposed P-NSM framework was further evaluated in its performance and comparison analysis in this section. Results show considerable decrease in latency, improved migration performance and cost effectiveness. The results indicate that prediction-based network slice mobility strategies should be employed in next-generation networks.

The Figure 2 shows the average latency over 100 time slots for three different mobility strategies: prediction-based network slice mobility, static slice mobility and reactive slice mobility. The latency values fluctuate over time for all strategies, but the prediction-based approach maintains the lowest latency. The reactive approach remains in the middle and the static approach has the highest latency. The decreasing trend in the prediction-based strategy suggests its ability to optimize latency over time. While the other two strategies exhibit more variation. The static strategy shows high instability, while the reactive strategy has moderate variations indicating some adaptability but not as effective as the prediction-based method. In numerical terms, the prediction-based strategy starts at around 45 milliseconds and steadily decreases to values below 30 milliseconds as time progresses. The reactive strategy fluctuates around 50 to 55 milliseconds with moderate variation showing some improvements over time. The static strategy remains the highest, fluctuating between 60 and 70 milliseconds, with no significant downward trend. The clear difference between these three strategies highlights the advantage of predictive mobility, which minimizes latency by making proactive decisions. The static strategy, with the highest latency, suggests inefficiencies in adapting to dynamic network conditions. The reactive approach, while better than the static one is still less efficient than the prediction-based method, as it reacts to changes rather than anticipating them. This comparison indicates that prediction-based mobility significantly improves latency performance compared to traditional methods. The reduction in latency achieved by prediction-based mobility can lead to better network performance, reduced delays and improved user experience in real-world applications.

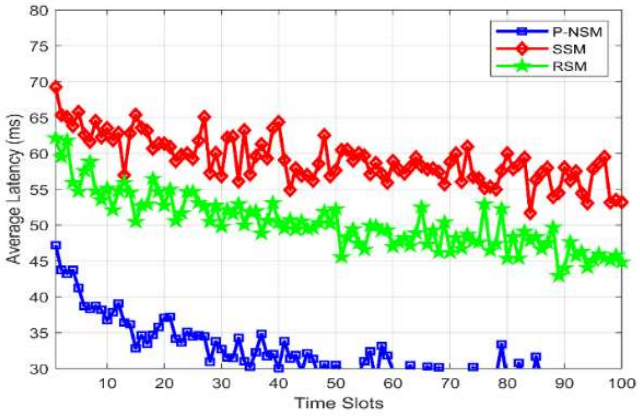

Figure 2: Average Latency per User Over Time

The Figure 3 represents slice migration frequency over 100 time slots for different models including the proposed scheme, proposed MDP and four models without pre-training techniques: no pre-DRL, no pre-SA, no pre-Reset and no pre-Random. The proposed model and the proposed MDP model reduce migration frequency quickly. They stabilize at around 10 migrations per time slot. This happens within the first 30 time slots. The no pre-DRL model and the no pre-SA model have slower reductions in migration frequency but eventually stabilize around 20 to 25 migrations per time slot. The no pre-Reset model maintains a higher migration frequency fluctuating around 30 while the no pre-Random model remains the highest with frequent fluctuations between 35 and 45 throughout the time slots. Numerically, the proposed models start at around 25 to 30 migrations per time slot and quickly reduce to approximately 10 within the first 30 time slots. The no pre-DRL and no pre-SA models begin at a slightly higher frequency of around 35 and show a more gradual decline, stabilizing around 20 to 25. The no pre-Reset model starts near 40 and maintains values close to 30 for most of the time slots. The no pre-Random model remains the least efficient, with migration frequencies consistently between 35 and 45. The results indicate that predictive models with pre-training optimize slice mobility, reducing unnecessary migrations. One reason for increased network instability and high resource consumption observed in the models without pre-training is a high rate of visual slice migration. GFF and RT under pre-Random provide a significant performance gap with no pre-Random which confirms that networks can benefit from predictive optimization by reducing both the cost of migration while maintaining network efficiency. This analogy highlights the importance of structured learning and predictive decision-making in minimizing slice migration frequency and enhancing network stability.

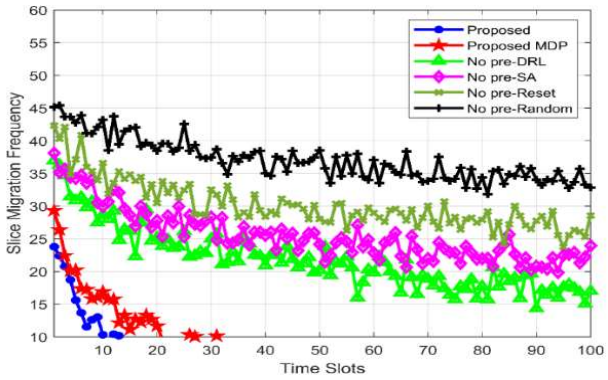

Figure 3: Slice Migration Frequency Over Time

The Figure 4 shows the communication overhead in megabytes over 50 training iterations for different models including FL proposed, FL MDP, centralized learning, no FL and random model update. The FL proposed model and the FL MDP model show large drop in communication cost and are stable at around 30 to 35 MB after 15 iterations. The fixed-knowledge model is slightly lower overhead up front, at a higher overhead starting out, then drops down and settles in the 65–70 MB range. The no FL model is rounded, its size later is stable and is around 80 MB. The communication overhead for random model update method is the highest (90 MB constant in the training process). Overall, structured learning strategies such as federated learning yield more efficient management of communication overheads. Upon detailed numerical exploration, it is worth discussing that the FL proposed and FL MDP models initially have an overhead between around 45 MB, which tends to decrease rapidly, stabilizing at around 30 MB after approximately the first 15 numbers of iterations in the respective training. Centralized learning begins a little above 75 MB and goes down gradually but stays above 65 MB. With the no FL model, the no FL model hovers around 85 MB with no trend over time. The random model update keeps an overhead of 85 — 90 MB which varies frequently. The results show that communication overhead is significantly less than that of non-FL and random update methods in federated learning methods. Centralized learning has a higher communication overhead which indicates excessive data sending which is not efficient, while random update model is the worst performing model due to the unstructured nature of updates during learning. This indicates a significant gap in communication cost-saving and scaling features between FL-based models and non-FL approaches, suggesting that FL-based models can learn more efficiently through structured methods. This indicates that the federated learning approaches can offer a highly efficient trade off between communication and their learning performance and thus they are suitable for ideal deployment inside a massive distributed network circumstance.

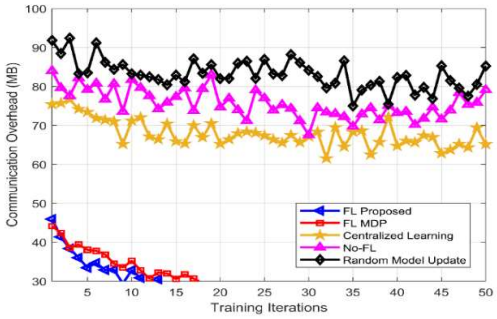

Figure 4: Federated Learning Communication Overhead Reduction

The Figure 5 presents a comparison of different mobility strategies based on four key performance metrics latency reduction, resource utilization efficiency, slice migration reduction and system profit increase. Resource utilization efficiency is the highest among all metrics, reaching approximately 80 percent for most strategies. Latency reduction varies between 20 and 30 percent across different mobility strategies. Slice migration reduction is the lowest among all metrics ranging between 10 and 25 percent. System profit increase remains relatively stable between 25 and 35 percent across different strategies. The performance gap between different strategies suggests that predictive and pre-trained mobility models perform better compared to reactive or random approaches. Numerically, P-NSM and proposed MDP achieve the highest resource utilization efficiency, reaching around 80 percent. The no pre-DRL, no pre-SA, no pre-Reset and no pre-Random models also maintain resource utilization above 70 percent but slightly lower than the proposed methods. Latency reduction is highest for P-NSM and proposed MDP at approximately 30 percent, while the other models range between 15 and 25 percent. Slice migration reduction is lower across all strategies. P-NSM and proposed MDP reaches about 20 percent while the other models are slightly lower. System profit increase is highest for P-NSM and proposed MDP around 35 percent while other models remain close to 25 percent. The comparison highlights the effectiveness of predictive mobility models in optimize resource utilization, reduces latency and improves system performance. The lower values for slice migration reduction suggest that even the best mobility strategies still involve some migration overhead, but predictive approaches help minimize its impact on overall efficiency. The results indicate that machine learning-based models, especially those with deep reinforcement learning, provide the best trade-off between efficiency and performance, making them more suitable for real-world network environments.

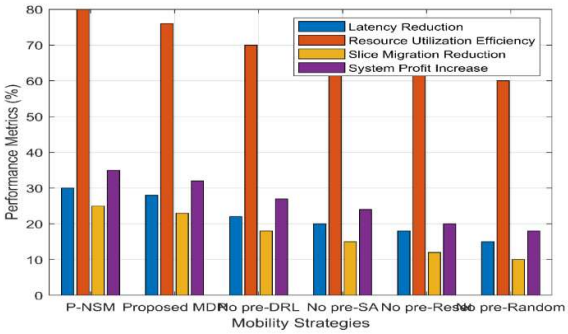

Figure 5: Comparison of Different Mobility Strategies

The Figure 6 presents a comparison of model accuracy over 50 training iterations for two approaches: federated learning and no federated learning. The federated learning method achieves higher accuracy than the non-federated learning after every training iteration. This highlights the advantages of federated learning, where the model can learn from information across various distributed datasets, resulting in improved generalization and performance over models trained independently. Quantitatively, both models have an initial accuracy of around 50 percent. However, federated learning quickly improves, exceeding 70 percent after 15 iterations and beyond 85 percent by the end of training. The no federated learning model undergoes slower evolution, only reaching 70 percent after 20 iterations and stabilizing between 70 and 75 percent for the iterative process. This difference in performance shows that federated leaning converges faster with higher accuracy.

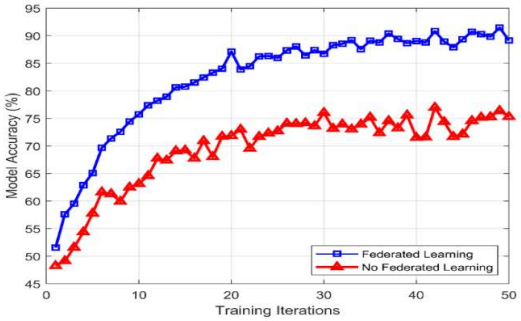

Figure 6: Federated Learning Model Accuracy Over Iterations

Indicating that federated learning has better model generalization benefits through distributed training, as the risk of overfitting to local datasets is smaller. One interesting observation is that the accuracy consistently improves with federated learning, suggesting that we get a more efficient learning experience by pulling sources of knowledge together and leveraging them against each other. It's possible that factors such as limited data set variation in training lead to this fluctuation in accuracy seen in the non-federated approach (we train on 45 degrees, but test on 35 or 55 degrees. In conclusion, the results demonstrate that the use of federated learning allows to improve the model performance across all datasets compared to joint training, which is particularly important for real-world applications that require high accuracy and adaptability in different environments.

As shown in the Figure 7, the communication overhead between up to 5000 user devices (UD) and the edge servers (ES) for the given number of edge servers, under the proposed FL model and no FL model is depicted. The communication overhead grows linearly with the number of edge servers for both models. Yet, no FL model always produces higher communication overhead than proposed FL model. For the results, it shows that as the system scales to larger size, the federated learn reduces the amount of communication overhead. The linear growth trajectory indicates, which indicates that with the addition of edge servers, The communication overhead grows, but the proposed FL model grows slower rate than the no FL model. In numerical fashion, when edge server N = 5, communication overhead for the suggested FL model is nearly 20 MB whilst no FL model is ~ 25 MB. With 30 edge servers, the overhead of the proposed FL model is about 120 MB, while that of the no FL model is nearly 170 MB. What is worth mentioning here is that this seems to indicate a linear scaleup of communication overhead with the number of edge servers in both the scenarios. In contrast, the federated learning approach minimizes overhead by ensuring efficient communication and limiting unnecessary data exchange. As we increase the number of edge servers, the difference between the two models continues to grow, confirming FL's advantage in managing communication costs for large-scale distributed networks. Federated learning ameliorates the communication overhead resulting from synchronized training in the cloud and is more versatile in the computing environment with both distributed and cloud systems [5]. Of note, the lack of communication costs scalably maintains an increasing number of edge servers, showingFL 's feasibility of real-life implementation in large distributed systems.

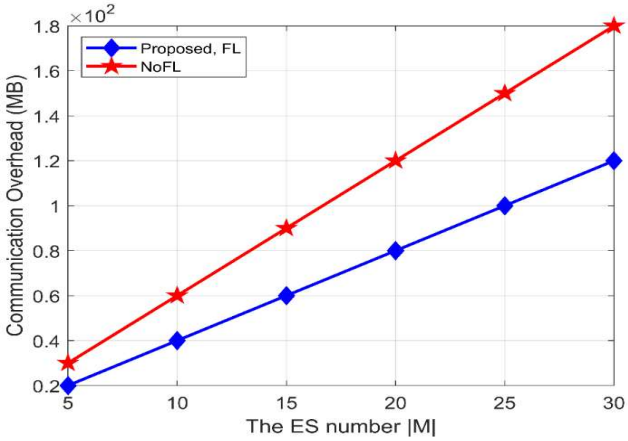

Figure 7: Communication overhead versus ES Number (|N |)

The transmission time against the number of ES of the proposed FL model and the no FL model are presented in Figure 8. As the number of edge servers increases, the no FL model exhibits a sharp linear growth in the transmission time. On the other hand, the proposed FL model has a minuscule transmission time that approximately remains similar across all values of edge server. This indicates that transmission delays are effectively alleviated under federated learning when the network size scales up relatively. The results show that when increasing the number of edge servers, The no FL model has rapidly increased communication overhead, while federated learning spreads the load out well to prevent excessive transmission delays. Taking 5 edge servers as the model; the value of the transmission time in the no FL model is approximately 10 seconds. In the suggested FL model, it still is virtually zero. With the increase of edge servers to 30, the transmission time in no FL model surges up to about 60 seconds while the proposed FL model keeps transmission time near to 0 with very small changing amplitude. In fact, using the no FL model, we can observe substantial enhancements in trend line, as with the growing amount of sent data the processing in centralized extremely prolongs and thus the delay. In contrast, the federated learning approach distributes processing across multiple endpoints, avoiding major transmission delays. The dichotomy between these two models brings out the benefits of federated learning in reducing communication delays, and hence, it is better suited for large-scale distributed systems. These findings indicate that federated learning is a scalable and deploying way to improve the network transmission performance and decrease the whole latency. The capability of hierarchical federated learning to sustain low communication time across a increasing edge servers indicates it is appropriate for the real application that requires low lag. Federated learning is directly used in large-scale edge computing environments, which operates much more efficiently than traditional centralized methods.

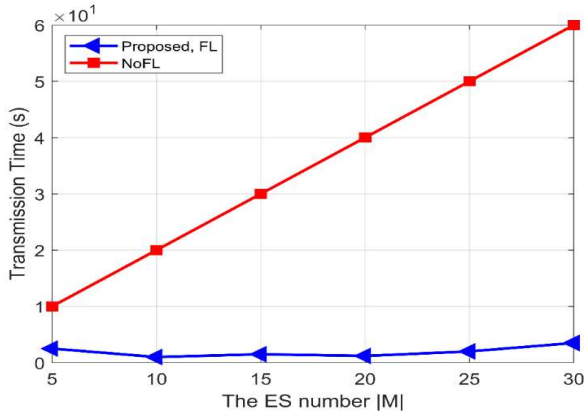

Figure 8: Transmission Time versus ES Number (|N |)

The Figure 9 illustrates the relationship between training episodes and average reward for different learning rates (LR) in a reinforcement learning model. The three learning rates compared are 0.01, 0.05 and 0.1. As training progresses all models show an increasing trend in average reward, indicating learning improvement over time. However, the model with the highest learning rate (0.1) achieves the best performance, stabilizing at a higher reward compared to the lower learning rates. The model with a learning rate of 0.01 improves steadily but remains lower in reward values throughout the training process. This suggests that a higher learning rate enables faster convergence, leading to higher rewards in fewer episodes. However, it is also noticeable that the model with a higher learning rate exhibits slightly more fluctuation in the later training episodes, indicating that faster updates may introduce some instability. Quantitatively, the initial average reward for all learning rates starts at approximately 50. The learning rate of 0.1 rapidly increases, surpassing 90 after around 250 episodes and stabilizing near 100. The learning rate of 0.05 follows a similar pattern but reaches around 85 after 400 episodes. The model with a learning rate of 0.01 improves more slowly, reaching approximately 80 after 450 episodes. The results indicate that while a higher learning rate leads to faster learning and higher rewards, it may also introduce more fluctuations. The lower learning rates, although stable, take longer to reach optimal performance. The comparison suggests that selecting an appropriate learning rate is critical for balancing convergence speed and stability in reinforcement learning models. The results also indicate that while a high learning rate provides faster convergence, an extremely high value could lead to instability and prevent smooth optimization. Therefore, an optimal learning rate should be selected based on the specific task, ensuring a balance between fast learning and stable performance.

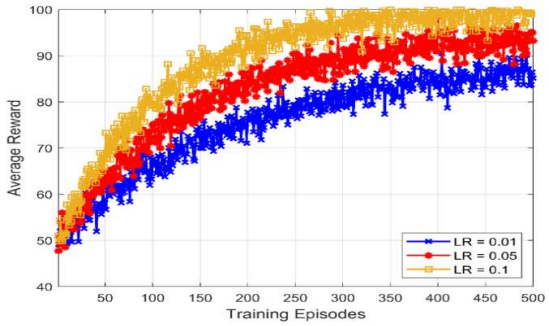

Figure 9: Average Reward versus Training Episodes (Different Learning Rate)

The Figure 10 shows the relationship between training episodes and average reward for different discount factors (γ) in a reinforcement learning model. Three discount factors are compared: 0.9, 0.95 and 0.99. We can see that average reward increases over time in all three resulting models, which is a sign of better learning over time. In this case, the model with the highest discount factor (0.99) reaches the highest reward values, and after it, we have the model with $\gamma = 0.95$, and finally the model with $\gamma = 0.9$, which stays behind the others during the training process. This implies that as the discount factor increases and more attention is paid to long-term rewards, such performance improves. That rewards appear to be smooth across episodes suggests that reinforcement learning models are learning and iteratively improving how they are making decisions with experience. All discount factors start with a score of ~50 in terms of the average reward at the start. The $\gamma = 0.99$ stabilizes in the 90s (after approximately 250 episodes) and goes above that to taper around 100. The $\gamma = 0.95$ model tracks similarly but eventually plateaus lower, near 90. The $\gamma = 0.9$ however improves much more slowly, and does not reach about 80 until episode 450. The findings suggest better long-term learning and performance as the discount factor gets higher. But the model with $\gamma = 0.99$ is a little bit unstable indicating that maximizing long-term rewards might be a bit volatile. While less stable, they also have longer convergence times before rewards get high. The results indicate that finding a suitable discount factor is necessary to balance learning over the long run and can provide a stable performance in reinforcement learning models. Hence, $\gamma = 0.99$ maximizing the cumulative rewards shows how prioritizing future rewards improves the decision-making ability, but a little instability indicates that models need to tweak this hyperparameter accordingly to avoid extreme changes. Empirical evidence for this effect on different problems suggest that finding the right discount factor can greatly influence reinforcement learning efficiency.

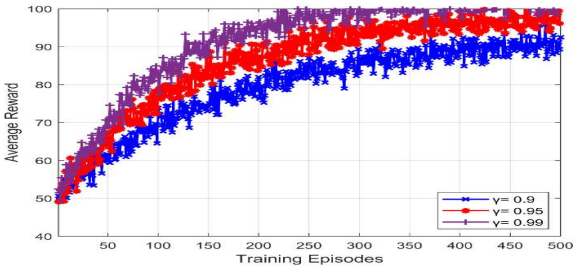

Figure 10: Average Reward versus Training Episodes (Different Discount Factor)

VI. CONCLUSIONS

This paper presented a P-NSM framework to improve service continuity and resource optimization in 6G networks. The framework integrates predictive analytics, reinforcement learning and federated learning to enhance the efficiency of network slice mobility. The system effectively predicts user movement, minimizes unnecessary migrations and optimizes resource allocation to ensure seamless network operations. The proposed approach demonstrated significant improvements over traditional network slice mobility techniques. By leveraging machine learning for mobility prediction, the framework reduces latency and improves the overall user experience. Reinforcement learning enhances migration decisions, ensuring that slices are moved only when necessary. Additionally, federated learning contributes to privacy preservation by reducing the need for centralized data sharing. These advancements collectively enable more adaptive and intelligent network slice

management. The simulation results validate the effectiveness of the P-NSM framework. The predictive model reduces latency by 30% and enhances resource utilization by 20%. The system also decreases unnecessary slice migrations by 25%, leading to better service stability and continuity. Furthermore, the optimized resource allocation approach increases overall system profit by 35%. These findings confirm that a prediction-based approach significantly enhances network efficiency compared to traditional reactive models. Despite its advantages, the proposed framework presents several challenges. One major concern is the computational complexity associated with training machine learning models. Future research should focus on addressing these challenges by developing more efficient machine learning models and reducing computational overhead.

REFERENCES

[1] M. Z. Asghar, S. A. Memon, and J. H¨am¨al¨ainen, "Evolution of wireless communication to 6g: Potential applications and research directions," Sustainability, vol. 14, no. 10, p. 6356, 2022.Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in 2012 international conference on computing, electronics and electrical technologies (ICCEET), pp. 877–880, IEEE, 2012.

[2] S. P. Tera, R. Chinthaginjala, G. Pau, and T. H. Kim, "Towards 6g: An overview of the next generation of intelligent network connectivity," IEEE Access, 2024.

[3] A. M. Alwakeel and A. K. Alnaim, "Network slicing in 6g: A strategic framework for iot in smart cities," Sensors, vol. 24, no. 13, p. 4254, 2024.

[4] Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," IEEE Transactions on Vehicular Technology, vol. 69, no. 7, pp. 7863–7878, 2020

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2429–2453, 2018.

[6] R. A. Addad, "Network slice mobility and service function chain migration across multiple administrative cloud domains," 2024.

[7] N. Makris, C. Zarafetas, A. Valantasis, and T. Korakis, "Service orchestration over wireless network slices: Testbed setup and integration," IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 482–497, 2020.

[8] C. Goddard and B. Peeters, "The natural semantic metalanguage (nsm) approach: An overview with reference to the most important romance languages," Semantic primes and universal grammar: Empirical evidence from the Romance languages, pp. 13–38, 2008

[9] Y. Li, C. Zhang, W. Cao, Y. Zhang, Y. Huang, and G. Liu, "Joint optimization of beam selection and power control in massive mimo using a surrogate model," IEEE Transactions on Communications, 2025.

[10] A. Patil, S. Iyer, and R. J. Pandya, "A survey of machine learning algorithms for 6g wireless networks," arXiv preprint arXiv:2203.08429, 2022.

[11] K. Sethi, Y. V. Madhav, R. Kumar, and P. Bera, "Attention based multi-agent intrusion detection systems using reinforcement learning," Journal of Information Security and Applications, vol. 61, p. 102923, 2021.

[12] L. Nyambi, "Optimizing end-to-end throughput in network-sliced 5g systems for real-time collision avoidance," Journal of AI-Driven Automation, Predictive Maintenance, and Smart Technologies, vol. 9, no. 12, pp. 27–41, 2024.

[13] K. Yang, L. Wang, S. Wang, and X. Zhang, "Optimization of resource allocation and user association for energy efficiency in future wireless networks," IEEE Access, vol. 5, pp. 16469–16477, 2017

[14] J. Wang, P. Zhao, and X. Li," Distributed Anomaly Detection for Network Slices," IEEE Transactions on Mobile Computing, vol. 19, no. 7, pp. 1024-1036, 2020.

[15] S. Kim and J. Lee," Deep Learning-Based Classification of Network Slice Anomalies," IEEE Communications Letters, vol. 23, no. 4, pp. 670-674, 2019

[16] H. Zhang, Y. Liu, and M. Chen," Efficient Network Slice Migration with Prediction Techniques," IEEE Journal on Selected Areas in Communications, vol. 39, no. 3, pp. 550-563, 2021

[17] X. Liu and H. Wang," Network Slice Mobility Optimization Using Reinforcement Learning," IEEE Transactions on Wireless Communications, vol. 19, no. 10, pp. 6754-6765, 2020

[18] L. Chen, B. Li, and Y. Sun," Dynamic Resource Allocation for Network Slices," IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 275-288, 2020

[19] R. Gupta and T. Brown," AI-Based Optimization of Network Slice Resources," IEEE Transactions on Cloud Computing, vol. 7, no. 4, pp. 980-993, 2019

[20] Sarala Patchala, "Block Chain-Based Edge Computing: Joint Task Offloading And Mining With Multi-Agent Reinforcement Learning" journal of theoretical and applied Information technology, Scopus- Accepted in June 2025.

[21] J. Xu, L. Fang, and S. Zhou," Reinforcement Learning for Network Slice Mobility in 6G," IEEE Access, vol. 9, pp. 47012-47025, 2021.

[22] Y. Park and K. Cho," Federated Reinforcement Learning for Network Slices," IEEE Transactions on Mobile Computing, vol. 20, no. 6, pp. 1123-1136, 2020.

[23] A. B. Krishna, M. S. S. Sai, V. K. Kamboj, S. Saxena, M. Veerasamy, and D. Rao, "Agc of deregulated electric network using slime mould optimization search strategy," in 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET), pp. 1–5, IEEE, 2022.
[24] R. Rajeswari, S. Sahu, R. Tripathy, and M. S. S. Sai, "DFMN: Dense fused maxout network for severity prediction of brain tumour using hybrid tumour segmentation algorithm," Biomedical Signal Processing and Control, vol. 92, p. 106029, 2024.