Federated learning framework for prediction based load distribution in 5G network slicing

Nitul Dutta*
Department of CSE, SRM University
Guntur, Andhra Pradesh, India
nituldutta@gmail.com

Rajesh Mahadeva
Department of CSE, MIT-MAHE
Manipal, Karnataka, India
rajeshmahadeva15@gmail.com

Shashikant P. Patole
Department of Physics, Khalifa University
Abu Dhabi, UAE
shashikant.patole@ku.ac.ae

Gheorghita Ghinea
Department of Computer Science, Brunel University
London, United Kingdom
George.Ghinea@brunel.ac.uk

ABSTRACT

The 5G technology brings transformative changes across sectors like healthcare, automotive, and entertainment by integrating massive IoT networks and supporting dense device connectivity. Network slicing in 5G further ignites the capability by allowing tailored virtual networks for specific applications, enhancing operational efficiency and user experience across diverse scenarios. In this paper we propose a framework to use Federated Learning (FL) in 5G network slicing to support service assignment. The aim is to optimize the network traffic allocation among various slices. It first predicts the load on each network slice and then the incoming traffic is allocated to a slice which is most suitable and not heavily loaded. The DeepSlice dataset on 5G slicing is horizontally splited into multiple segments to train a federated CNN model which are deployed across multiple clients. The model is analyzed with varying number of clients and parameters such as accuracy and loss are observed. The performance of federated approach is compared with centralized approach of prediction keeping essential hyper parameters unchanged. Outcomes in terms of training and testing is presented for better interpretation of the proposed framework. Observation shows that the federated learning outperform the centralized technique in accuracy as well as loss.

CCS CONCEPTS

• **Computing methodologies** → **Federated Learning**; **Neural networks**; **Machine learning algorithms**.

KEYWORDS

5G, Network slicing, Federated learning, Resource allocation in 5G.

ACM Reference Format:
Nitul Dutta, Rajesh Mahadeva, Shashikant P. Patole, and Gheorghita Ghinea. 2024. Federated learning framework for prediction based load distribution in 5G network slicing. In *2024 Sixteenth International Conference on Contemporary Computing (IC3-2024) (IC3 2024), August 08–10, 2024, Noida, India.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3675888.3676085

*All authors contributed equally to this research.

IC3 2024, August 08–10, 2024, Noida, India

ACM ISBN 979-8-4007-0972-2/24/08
https://doi.org/10.1145/3675888.3676085

1 INTRODUCTION

The advent of 5G technology marks a significant milestone in the evolution of telecommunications, promising substantial enhancements over its predecessor, 4G [12]. Characterized by higher data rates, reduced latency, increased connectivity, and greater bandwidth, 5G networks are set to revolutionize various sectors including healthcare, automotive, entertainment, and industrial automation. The ultra fast speed supported by 5G network has transformed the real-time data processing by integrating a massive scale of IoT devices deployed in various environments including smart cities and industries. Moreover, it also capable of managing thousands of connected devices per square kilometer thereby leveraging the urban management and sustainability. The enhanced mobile broadband (eMBB) offered by 5G can support new multimedia applications, virtual reality experiences, and advanced gaming technologies, thereby enriching the user experience and creating new business models. Additionally, 5G networks are poised to boost the economy by unlocking new economic opportunities through innovative services and applications.

On the other hand, the network slicing in 5G enables a single physical network to support multiple virtual networks called network slices. Each of these slices could be tailored to serve different applications, or user groups [14]. For example, a slice could be configured with ultra-reliable low-latency communication (URLLC) for critical applications like remote surgeries or autonomous driving. Another slice could prioritize eMBB to cater high-speed data services needed for video streaming or large-scale broadcasts. Hence, the network slicing significantly increases operational efficiency of network operators by allowing them to deploy and manage multiple virtual networks with varying service levels, security, and performance characteristics. Even it does not demand multiple physical networks. Moreover, slicing improves user experience by seamlessly meeting network specific demands of different applications.

In the recent past, the research on 5G networks slicing has been targeting to optimize various operations focusing in lifecycle management, ensuring security, isolation between slices, and developing dynamic slicing algorithms [7]. Many research have also explored the integration of AI and machine learning methods [8] to predict

network demands and allocate resources proactively. Work is also reported on provisioning network slicing with edge computing to reduce latency and improve data handling capabilities. Recently, many researchers have demonstrated the use of FL in 5G network slicing, particularly for resource allocation and data security. Few of such works are discussed in the next section. The FL can also be used to predict network load and manage resources dynamically across slices, thereby improving scalability and responsiveness. This decentralized learning model through FL aligns with the distributed nature of 5G architectures and makes it ideal for real-time applications in diverse and sensitive environments. This fact motivates us to carry out this research.

Rest of the paper is organized as follows. In Section 2 few of the recent works on applications of Federated learning in network slicing is explained. Section 3 describes the proposed FL model for network slicing. Simulation results are found in Section 4 and paper is concluded in Section 5.

2 RELATED WORK

Although ample of research on 5G network slicing is found in the literature, the exploration with federated learning is limited. Few of the most prominent work with respect to the federated learning in 5G network slicing is presented in this section.

The work described in [3] is an implementation of FL technique to design a distributed defense systems for 5G network slicing. The proposed architecture is capable of detecting internal as well as external attacks targeting the critical components involved in network slicing. The framework is tested for certain well known attacks such as distributed DoS, botnets and cache poisoning. As stated by the authors, the system produces large overheads. A federated learning influenced digital twin architecture is reported in [4]. Authors have proposed to create a digital replica of the physical slicing network to mimic its complicated infrastructure. The federated approach of the stated scheme enables forecasting the network's dynamic performance. In proposed digital twin of network slicing is designated as a non-Euclidean graph representation which supports estimation of QoS metrics of the underlying network slices. The intelligent self-supervision method used in the technique demands accurate and appropriate hyperparameters for the success of the model.

In [5] a Stackelberg multi-leader and multi-follower game model for network slicing in peer-to-peer (P2P) network is proposed. Authors suggest a decentralized resource trading system integrating blockchain and federated deep reinforcement learning. It assists the network tenants to perform inter-slice resource sharing securely. Integration of block chain in the system introduces additional computational complexity in the framework. Another federated deep reinforcement learning approach for network slicing is described in [2] for RAN. A collaboration among multiple DL models deployed for aggregation in the federated learning phase. The synchronization and coordination among multiple agents is a critical operation for the success of the model.

There are few FL motivated slicing approaches found in various applications such as IoT, Edge computing etc. One of such work is discussed in [11] that integrates FL in IoT network. It is a two-tier resource allocation method that involves coordination between InP and IoT devices. Authors have designed a federated deep reinforcement learning-based resource allocation algorithm to explore the optimization. The proposed algorithm converges to the optimal solution and effectively maximizes utilization. Extension of federated model to slicing in Edge computing is reported in [13]. They adopted SDN to learn the local model's data distribution. The edge devices with its local model communicate with a global SDN federated model to meet the demand of dynamic network slicing. This model requires an accurate prediction by a global SDN federated controller for successful prediction accuracy for network slices. The research reported in [10] is a FL technique for slicing in optical network. They suggest that each client needs to periodically upload local model parameters and download global parameters. As suggested by the authors, this approach is also suitable for fog computing in passive optical network. Due to periodic upload and download of global parameters the algorithm consumes comparatively large bandwidth.

From the above discussions, it is noticed that there is a research gap in the area of 5G network slicing that incorporates distributed approach through FL. Proper prediction of network traffic load may significantly improve the performance of 5G network in terms of service provisioning. Furthermore, the FL can perform considerably better than the centralized DL models in analysing the hidden pattern in datasets. However, before considering FL for deployment in real applications a thorough analysis of the technique is necessary. In that context this research is aimed at exploring the FL model in 5G network slicing.

3 FEDERATED LEARNING FOR NETWORK SLICING

Each slice is considered as an independent, logical network tailored to meet specific requirements of a service or application. This allows for optimized use of the network's resources by dedicating different slices to different types of traffic. A network infrastructure which supports virtualization technique, divides the network functions and resources into isolated slices and characterise by their own performance parameters. This customization is managed through software-defined networking (SDN) and network function virtualization (NFV), which allows dynamic allocation and reallocation of network resources based on real-time demand.

3.1 Problem description and dataset

The problem of this research work is to optimize the network traffic allocation among all the slices using federated learning approach. The designed model first predicts the network load on each network slice based on the previous information. It assists in determining the utilization of output 'network slices'. Based on this analysis, the incoming traffic is allocated to a slice which is most suitable and not heavily loaded. The DeepSlice dataset on 5G slicing technology as described in [15] is used in this work. It comprises extensive data used by researchers and developers to train machine learning models that optimize and manage network slices. It features diverse data points including network traffic patterns, service requirements, quality of service metrics, and other operational parameters of network slices. For detailed information about the dataset readers may refer to [16] and [6].

3.2 Methodology

After defining the objective (optimized allocation of traffic to different slices), and performance parameters (maximize the accuracy and minimize the loss) as in previous subsection, the CNN model is planned to use through federated learning approach. Following the principles of FL, the proposed framework operates in two different levels, namely client and server levels. The training of the CNN is performed in both these levels. For the ease of computation, only shallow layer parameters are communicated to clients. After a certain rounds of iteration the clients performs deep parameter update. The ratio of the shallow layers to total neural layers and interval of deep update are considered as hyper-parameters. The server initializes the model with basic parameters and distribute it to the participating clients. Clients perform local data cleaning and pre-processing to ensure the suitability of data for training. Then each client trains the model locally with its own data. This step models that all training data remains on the local device and it preserves the security. Local training adapts the model to the specific conditions and patterns of each slice. After certain rounds of training each client perform a deep update where server is informed with the results of each client. On receipt of the update, the server aggregates various weights and train its own model with its holding data. Likewise, the server accordingly adjust its weights and update clients for the second round of training. This update reflects the learned patterns across all participating nodes in the network. Detailed description of the model is explained in Section 4.B. Further, aggregation of training results of clients can be performed in a decentralized manner among peers however, in this work centralized aggregation is considered. The CNN model works by means of repeating the above described process of local training, aggregation, and global updating iteratively. Each cycle refines the model's performance. Incorporating new data and feedback into the model during the training process can further enhance the prediction of the model.

3.3 Proposed algorithm

The algorithm works with two different procedures. One for the client and another for the server. The client receives hyperparameters from the server and perform the training on its local dataset. After the training it performs deep update with the server. The procedure is given in Algorithm 1. On receipt of the deep update from all clients, the central server aggregate all the received parameters. Based on the aggregated parameters, it trains its own model. The performance of the model is then evaluated and check for required error level. If the error of the current model is within the permissible limit then server stops model training and sends the updated parameters to all clients. Otherwise the training of the model continues in the server. The procedure is described in Algorithm 2.

The proposed expert system if implemented in a real network, can assists the system intelligently in learning and adapting to changes or new requirements. It does not need any clear rules for handling incoming service types (such as handovers, voice transmission, data etc.) This module can also support in identifying and accommodating previously unknown demands in the network.

Algorithm 1: clientTrain (local model M_l)

```
Input: hyperperameters
Output: trained model with deep parameters for M_c
for i = 1 to noofrounds do
    Train M_u locally
    if i == noofrounds then
        Perform deep update M_c
    end
end
Return M_c;
```

Algorithm 2: serverTrain(central model M_s)

```
Input: Deep parameters from clients
Output: trained model M_s
for each client do
    Perform aggregation on deep parameters
end
Update model M_s
Send Feedback updated model to all clients
if (currLoss ≤ minLoss) then
    break
end
Return M_s;
```

4 SIMULATION

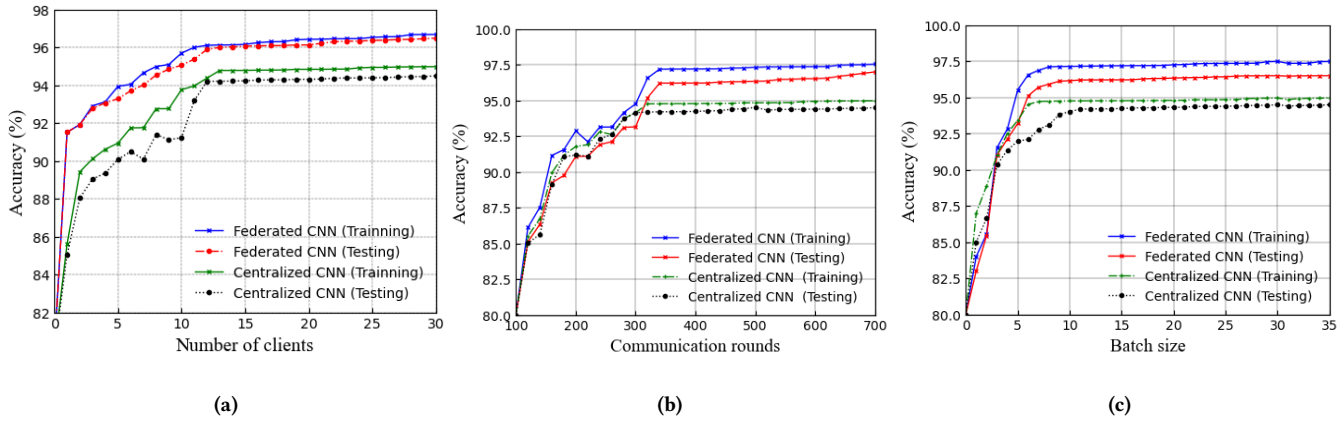
In order to evaluate the performance of federated learning on the network slicing data a simulation is carried out in python. Details of the simulation setup and setup for federated learning environment is described in this section.

4.1 Simulation environment

The deep neural network namely a CNN model is designed for the simulation of the proposed federated scheme. The model comprises of single input and single output layer and three fully connected hidden layers with 256, 128, and 120 neurons, respectively. In the first two layers ReLU activation function is used with a discount factor to γ=0.9 and in the last layer softmax activation is integrated. The Keras library for deep learning is used within Python. The episode and batch sizes for the FL model are experiment specific and is described ialong with the results in Section 4.C. All simulations are executed in a NVDIA DGX Server 5.5.1 which comprises of 8 NVIDIA Tesla V100 GPUs, has computation capacity of 1 Peta Flops and 20 number of Intel Xeon E5 CPU cores. The system has the GPU memory of 128GB and system memory of 512GB.

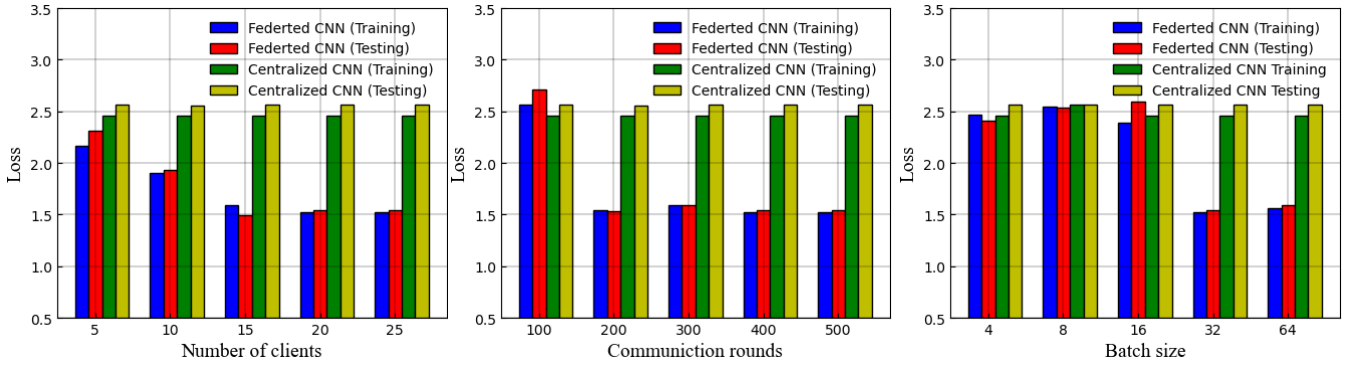
4.2 Federated computation

The federated members in FL's real world application is distributed in the network with their own data. Each of these members work in isolation with their local dataset. In the simulation of FL, shards of clients are created to replicate the real world environment. In this simulation training is performed with 10 to 30 shades of clients. The training dataset is partitioned horizontally and distributed to all clients after scuffling them. All these clients interact with a central server for federated computation. The shuffled training



(a) (b) (c)

Figure 1: Accuracy analysis of training and testing for federated and standard SGD. (a) Impact of volume of clients (the global epoch is 500, local epoch 200, batch size 32). (b) Impact of global epoch (number of clients 10, local epoch 200, batch size 32). (c) Impact of batch size (number of clients 10, the global epoch is 500, local epoch 200).



(a) (b) (c)

Figure 2: Loss analysis of training and testing for federated and standard SGD. (a) Impact of volume of clients (the global epoch is 500, local epoch 200, batch size 32). (b) Impact of global epoch (number of clients 10, local epoch 200, batch size 32). (c) Impact of batch size (number of clients 10, the global epoch is 500, local epoch 200).

data set is assumed to be a the local copy of data for the client. In the initial round of the computation the server sends few hyper-parameters to all clients and each client train their multi layer perceptron DNN model on their local dataset. The model uses SGD as a optimizer with a horizontal data partitioning for all clients. The clients executes the model for number of local epochs (assumed 100) to complete one iteration, also called a communication round. After completion of the number of *comm_round* (such communication rounds varying between 100 to 900 in this implementation also denoted as global epochs) an aggregation on local parameters is performed by the client. At this point the client initiates a deep update by sending computed wights to the to the server. The server then aggregates all the client parameters and train its own model. Once the server meets the loss requirement of the system it stops and update weights to all clients based on certain criteria set by the model. This step triggers the learning rate of the FL model. Here, rather than decaying the learning rate with respect to the number of local epochs (in client) like in general centralized DL, in FL based DL, decay happens with respect to the number of global aggregations. This is certainly determined by the value of the *comm_round* which is a hyper parameter for the model.

The central point of success of FL is the parameter aggregation process of the model. The method of federated aggregation as defined in [9] is used in this work and stated below,

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{1}$$

where, $F_k(w)$ is defined as

$$F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$$

The Eqn. (1) is the component-wise sum of all scaled parameters generated by various clients. The $F_k(w)$ on the other hand is the

estimation of weight parameters for each client based on the loss values recorded across every data point every client has trained upon. These calculations are influenced by the proportion of a client's local training data with the overall training data held by all clients. Further, the parameter is influenced by the client's batch size which determines the total number of data points a client model is trained upon. In real world applications training data is disjoint and hence no single client can correctly estimate the quantity of the combined set. So, in such scenario each client has to send the the number of data points they trained upon while performing a deep update to the server. The client receives global hyper-parameters from the server at the beginning and train on the local data (shared shad) for a period determined by the local epochs. After the training, the new weights are scaled and scaled weights are send to the server. It completes single local training session in all clients. All the scaled weights are summed up by the server ans scaled the received local trained weights and updated the global model to this new aggregate. That ends a single full global training epoch. This process continues for number of times denoted by *comm_rounds*. In order to compare the results of federated approach with standard centralized CNN model, same set of training dataset is used. All the hyper parameters used for the FL training is used in the centralized model. However, batch size in this case is taken as the sum of all clients in FL model. This setting ensures that the centralized model possess exactly the same number of training samples per epoch as the global model did per communication round in FL model. A single MLP model is trained in a single batch. Performance of both the approaches are evaluated in terms of accuracy and loss. A cross entropy loss model is used in the evaluation. Simulated results are presented in the following subsection.

4.3 Results and analysis

(i) Accuracy. Accuracy of both the approaches with respect to training and testing is presented in Fig. 1 to understand the effectiveness of the model. Overall performance of FL model is better than centralized model. The testing accuracy is nearly 3-4% less than training accuracy. However, for centralized model, parameters like number of clients and batch size does not effect the performance. Fig. 1(a) shows the data against number of clients. With the increase in clients, accuracy of training and testing both increases. In Fig 1(b) the performance with respect to global epoch is shown. Accuracy increases till 400 and after that it remains consistent in 96%. The impact of batch size is depicted in Fig. 1(c). After a batch size of nearly 28, the accuracy becomes stable around 96%.

(ii) Loss rate. Fig. 2 shows the loss in both the models using the cross entropy loss. The loss suffered by FL framework is better than the centralized model. For Federated model the loss is nearly 1.5 and that other is 2.4. Detailed loss analysis is presented in various graphs. Fig. 2(a) shows the impact of number of clients on loss. Figure depicts that the loss get stabilized for clients more than 10 and above. Loss rate is presented for varying global epochs in Fig. 2(b) and against batch size in Fig. 2(c). Loss get stable at 1.5 near the global epochs of 500 and batch size of 30.

5 CONCLUSION AND FUTURE WORK

In this paper an approach to analyze 5G network slicing data using the federated learning is described. To simulate the FL model a set of clients and a server is created and the dataset is splited to all clients and server. After every round of computation the clients send back the weights to the server. The server aggregates weights and modifies the parameters. The modified parameters are then again sent to clients and it repeats another round of training. This process repeats several rounds till the loss reaches a required threshold. The performance of the FL model is analyzed in terms of accuracy and loss and is compared with Standard Gradient Descent (SGD) without FL. The results show that federated learning surpasses SGD in both accuracy and loss. The results are presented for both training and testing phases.

Although in simulated scenario FL performs better than standard DL approach, it may slightly vary in real world applications. It is because of the fact that in real world scenarios federated data held by clients are mostly NON independent and identically distributed (IID). Because of the heterogeneity of Data (non-IID data) in Federated Learning, when each client node updates its local model, its local objective may be far from the global objective. The averaged global model which is obtained by averaging the weights of the client nodes (following vanilla federated learning model algorithm used in the paper) do not meet the global optima. Hence, the Model-contrastive learning (MOON) [1], may be used to tackles Non-IID Data distribution by correcting the local updates. However, this concept is not examined in this version of the paper and left as a future work.

ACKNOWLEDGMENTS

Authors acknowledge all the anonymous reviewers for their valuable suggestions to make this work a presentable one.

REFERENCES

[1] 2021. Model-contrastive federated learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 0762–0767.
[2] 2023. Federated Deep Reinforcement Learning for Open RAN Slicing in 6G Networks. *IEEE Communications Magazine* 61, 2 (2023), 126–132.
[3] 2024. When Two-Layer Federated Learning and Mean-Field Game Meet 5G and Beyond Security: Cooperative Defense Systems for 5G and Beyond Network Slicing. *IEEE Transactions on Network and Service ManagementVolume* 21, 1 (2024), 1178–1189.
[4] Mohamed Abdel-Basset, Hossam Hawash, Karam M. Sallam, Ibrahim Elgendi, and Kumudu Munasinghe. 2023. Digital Twin for Optimization of Slicing-Enabled Communication Networks: A Federated Graph Learning Approach. *IEEE Communications Magazine* 61, 10 (2023), 100–106.
[5] Daniel Ayepah-Mensah, Guolin Sun, Gordon Owusu Boateng, Stephen Anokye, and Guisong Liu. 2024. Blockchain-Enabled Federated Learning-Based Resource Allocation and Trading for Network Slicing in 5G. *IEEE/ACM Transactions on Networking* 32, 1 (2024), 654–669.
[6] Khaled Bedda, Zubair Md Fadlullah, and Mostafa M. Fouda. 2022. Efficient Wireless Network Slicing in 5G Networks: An Asynchronous Federated Learning Approach. In *2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*. 285–289.
[7] Chamitha De Alwis, Pawani Porambage, Kapal Dev, Thippa Reddy Gadekallu, and Madhusanka Liyanage. 2024. A Survey on Network Slicing Security: Attacks, Challenges, Solutions and Research Directions. *IEEE Communications Surveys & Tutorials* 26, 1 (2024), 534–570.
[8] Adnei Donatti, Sand L. Corrêa, Joberto S. B. Martins, Antonio J. G. Abelem, Cristiano Bonato Both, Flávio de Oliveira Silva, José A. Suruagy, Rafael Pasquini, Rodrigo Moreira, Kleber V. Cardoso, and Tereza C. Carvalho. 2024. Survey on Machine Learning-Enabled Network Slicing: Covering the Entire Life Cycle. *IEEE Transactions on Network and Service Management* 21, 1 (2024), 994–1011.

[9] McMahan H., Moore Eider, Ramage Daniel, Hampson S., and Arcas less Y. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Int. Conf. on Artificial Intelligence and Stat.*

[10] Jun Li, Xiaoman Shen, Lei Chen, and Jiajia Chen. 2020. Bandwidth Slicing to Boost Federated Learning Over Passive Optical Networks. *IEEE Communications Letters* 24, 7 (2020), 1492–1495.

[11] Ruijie Ou, Guolin Sun, Daniel Ayepah-Mensah, Gordon Owusu Boateng, and Guisong Liu. 2023. Two-Tier Resource Allocation for Multitenant Network Slicing: A Federated Deep Reinforcement Learning Approach. *IEEE Internet of Things Journal* 10, 22 (2023), 20174–20187.

[12] Vuyo S. Pana, Oluwaseyi P. Babalola, and Vipin Balyan. 2022. 5G radio access networks: A survey. *Array* 14 (2022), 100170.

[13] Rakkiannan and al. et. 2023. An Automated Network Slicing at Edge with Software Defined Networking and Network Function Virtualization: A Federated Learning Approach. *Wireless Pers Commun* 131 (2023), 639–658.

[14] Guolin Sun, Daniel Ayepah-Mensah, Gordon Owusu Boateng, Noble Arden Kuadey, Mohamed Basher Omer, and Guisong Liu. 2023. Holistic Roadmap of Trends in Radio Access Network Slicing: A Survey. *IEEE Communications Magazine* 61, 12 (2023), 118–124.

[15] Anurag Thantharate, Rahul Paropkari, Vijay Walunj, and Cory Beard. 2019. DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks. In *Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*. 0762–0767.

[16] Anurag Thantharate, Rahul Paropkari, Vijay Walunj, Cory Beard, and Poonam Kankariya. 2020. Secure5G: A Deep Learning Framework Towards a Secure Network Slicing in 5G and Beyond. In *Annual Computing and Communication Workshop and Conference*. 0852–0857.