ECE 603 Final Project

Yiren Zhou 21124234
Ahmad Kamal 20767259

Repository: https://github.com/ahmadkama/ECE603_Project

Introduction

Wireless communication is a very important technology that is becoming ubiquitous in our life. It has transformed the way people connect and interact, enabling seamless data transmission over vast distances without physical connections. This field has continuously evolved from the early days of 2G and 3G to the cutting-edge technologies of 5G and the emerging 6G systems. These advancements aim to meet growing demands for higher bandwidth, ultra-low latency, and enhanced reliability. Wireless communication supports various applications such as mobile internet, smart cities, autonomous vehicles, and remote healthcare systems, reflecting its pervasive role in modern life.

The emergence of **5G networks** represents a significant leap in wireless communication technology, offering capabilities such as enhanced mobile broadband (eMBB), ultra-reliable low-latency communication (URLLC), and massive machine-type communication (mMTC). These services cater to diverse applications, from high-definition media streaming and immersive virtual reality experiences to mission-critical systems like autonomous driving and industrial automation. At the same time, in realizing these capabilities, the network has become more and more complicated and heterogenous.

Heterogeneous networks enhance flexibility and coverage, making them essential for accommodating the exponential growth in connected devices and data traffic in 5G and beyond. However, the complexity to scale and hardness of organization are to be carefully managed. A critical enabler of 5G's adaptability is **network slicing**, which allows multiple virtualized network instances to operate on shared physical infrastructure. This innovation ensures that each slice is optimized for specific quality of service (QoS) requirements, and simplifies the work of operators through manipulating only software instead of connecting wires and physical components.

The incorporation of **machine learning (ML)** and **deep learning (DL)** into network management and optimization has unlocked new possibilities for intelligent decision-making. ML algorithms analyze network data to predict traffic patterns, optimize resource allocation, and identify anomalies. In this report, we are reproducing a ML-based system for 5G networks that can dynamically assign network slices, balance loads, and mitigate slice failures. Advanced DL models, such as convolutional neural networks (CNNs) and long short-term memory

(LSTM) networks, further enhance these capabilities by enabling real-time analysis and prediction of network conditions. These tools are crucial for ensuring the efficiency, reliability, and scalability of 5G systems.

The synergy between 5G heterogeneous networks, network slicing, and machine learning not only improves network performance from physical metrics but also drives innovation in various aspects. From supporting autonomous systems and smart cities to enhancing healthcare delivery and fostering industrial automation, these technologies collectively shape the future of connectivity and digital transformation.

The following sections are organized as follows: Background section talks about related technologies such as network slicing and machine learning in detail. Experiment and reproducing section specifies how we reproduced the work. Extension section talks about how we improve the model and optimize it.

Background

I. Network Slicing

Network slicing is one of the most significant improving technologies used in modern 5G networks, enabling efficient, scalable, and customized use of physical infrastructure. Network slicing allows multiple logical networks to operate on a shared physical infrastructure. It provides end-to-end isolation, customization, and dynamic allocation of resources (with the aid of virtualized network resources such as communication/computing/caching) to meet diverse use cases such as massive mMTC, eMBB, and URLLC. As a prospective technology to play an important role in 6G and further, network slicing is designed to provide better QoS, improved resource efficiency, and to support heterogeneous device networks.

Key enabling technologies of network slicing include Software-Defined Networking (SDN) and Network Function Virtualization (NFV). SDN provides centralized control for dynamic resource allocation through the use of software controllers/application programming interfaces (APIs). On the other hand, NFV decouples network functions from hardware, enabling flexible orchestration and management of network functions/slices without interacting/modifying physical components, thus promoting programmability and scalability.

The use of ML in network slicing is beneficial because we are having more and more heterogeneous and complicated networks. Model-driven and

optimization methods might find hardships in encapsulating network dynamics and providing optimal resource allocation/network prediction results. Also, data-driven methods usually win over model driven methods on inference time because optimization problems take longer to be solved. The integration of ML and DL enhances decision-making in network slicing, enabling predictive resource allocation and real-time adjustments.

II. Major contribution from the paper to reproduce

The paper proposes DeepSlice, a deep learning neural network to manage slicing based network load efficiency and network availability. Based on relevant Key Performance Indicators (KPI) from both the network and device, the model is trained to predict appropriate selection of a network slice, also considering network load balancing and makeup for network failure. The model is trained and validated on large datasets representing diverse scenarios, showcasing its generalizability and applicability to real-world networks.

The paper emphasizes the resilience of DeepSlice in handling slice failures and unpredictable loads. Backup strategies, such as a master slice, are proposed to mitigate disruptions and ensure service continuity. The framework supports efficient load balancing and scalability, essential for the diverse and dynamic nature of 5G applications, including mMTC, eMBB, and URLLC.

In evaluating their model, the paper provided several key results. First, simulating network traffic over a time period of 24 hours, using the model to categorize traffic into slices and plotting the active device count for each slice. This is serving as a baseline for other evaluations. The paper evaluated categorization accuracy, as well as the situation of unseen device types/randomly selected input (feature value) and cross validation accuracy. Then using the network traffic simulation as a baseline, the paper evaluated their model in the situation of load balancing, where some slices are almost saturated with traffic, to see how their model will react. Similarly, they also tested the situation of a total slice failure. Last but not least, they plotted their model training accuracy and loss function value against training epoch to show fast and guaranteed convergence of their model.

Experiment and Reproducing

The DL model and the simulation specified in the paper were reproduced and tested with varying parameters. The reproduction was initially done to be similar to the procedure specified in the paper. Then, some minor modifications were implemented to follow best practices and optimize results.

I. Reproducing the DL model

The 'DeepSlice' DLNN (Deep Learning Neural Network) model takes as input the KPIs and produces an estimate of the appropriate slice type to use. The input is 8 neurons (one per KPI). There are four hidden layers, as specified by table 1. Note that the size of the output of layer 1 is 8 to match the size of the input of layer 2. In the paper, there was a mismatch between the output of layer 1 and the input of layer 2, which would result in an error.

Table 1: 'DeepSlice' DLNN Layer Specifications

Layer number	Input size	Output size	Activation Type
1	8	8	ReLU
2	8	4	ReLU
3	4	3	tanh
4	3	3	SoftMax

The output layer is a classification layer consisting of 3 neurons with a softmax activation. This guarantees that each output z_i, $0 \leq z_i \leq 1$, and that the sum of the outputs $\Sigma z_i = 1$. The output can be perceived as a 'probability' of the appropriate slice type being the one corresponding with each output. Thus, the slice type to use will be determined by the largest output, with the largest 'probability'.

The dataset specified by the paper was used, consisting of 65,000 example KPIs and corresponding slice type. The data was randomly split in a 70 - 20 - 10 split between training, validation, and testing. Since the paper did not mention any preprocessing on the data, no preprocessing was applied.

The DLNN was implemented in Python with Tensorflow Keras. A Sequential model was created using Dense layers with the appropriate size and activation type. The kernel initializer used was 'normal'; each kernel was randomly sampled from a zero mean Gaussian Distribution with a standard deviation of 0.05. The model was evaluated on its accuracy, and uses the Categorical Crossentropy

loss along with the Adam optimizer. The model was trained for 16 epochs. A plot of the model can be seen in Figure 1.

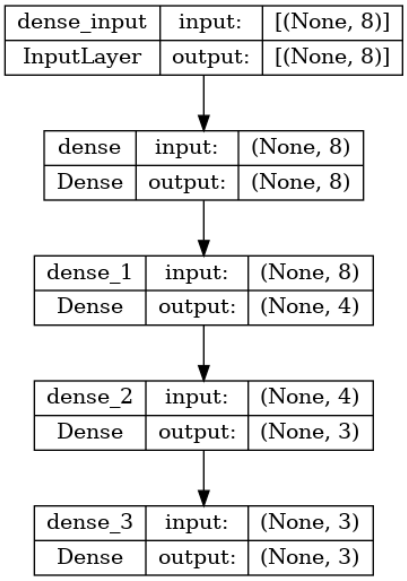

Figure 1: Plot of DLNN model

The performance of the model varied significantly based on the initializations. This was determined to be due to a lack of feature standardization, resulting in certain features dominating the learning. So, the data was standardized to zero mean and standard deviation of 1. This causes the range of the input data to be similar for all KPIs.

The model was trained with the training data and evaluated using the validation data. The loss and accuracy curves are plotted in figures 2 and 3. It was determined that the model reaches perfect accuracy in very few epochs, so early stopping was implemented to prevent overfitting.

Furthermore, the model was evaluated on the test dataset and it was determined to have an accuracy of 100% and a loss of 0.1. Thus, the DLNN correctly determines the appropriate Slice Type based on the KPIs.

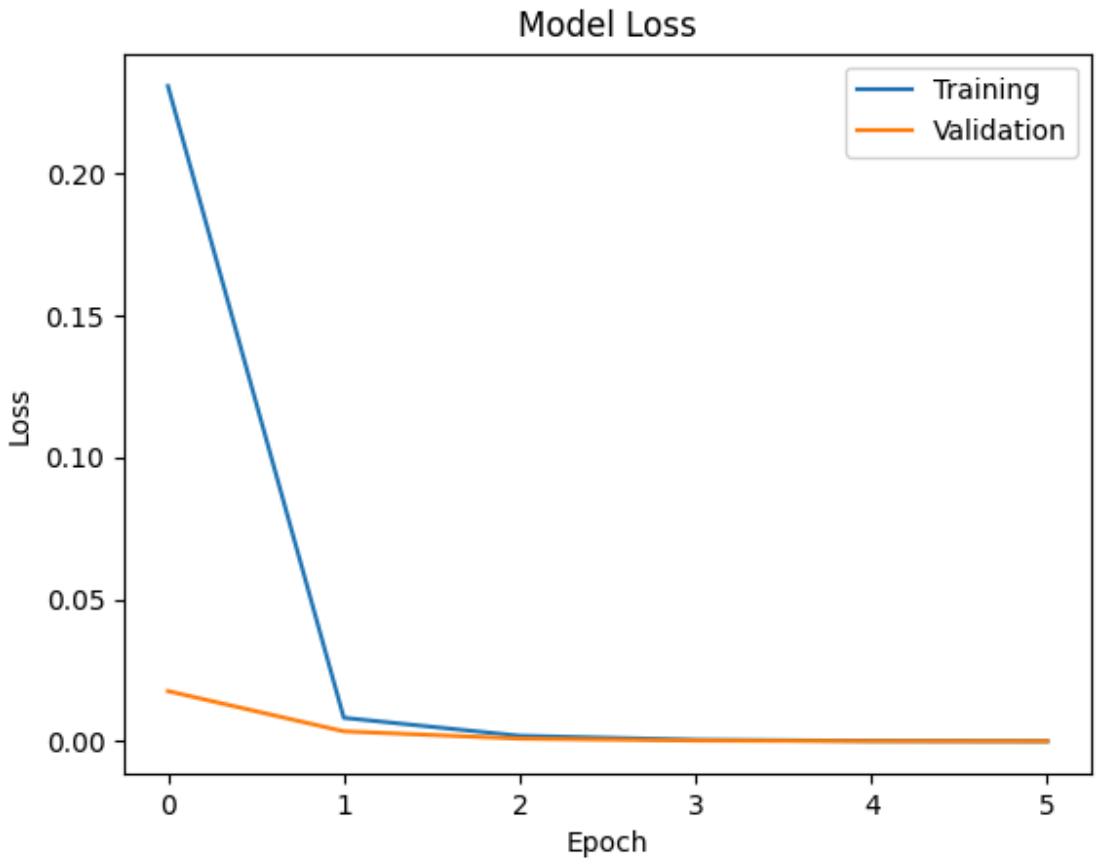

Figure 2: Model Loss

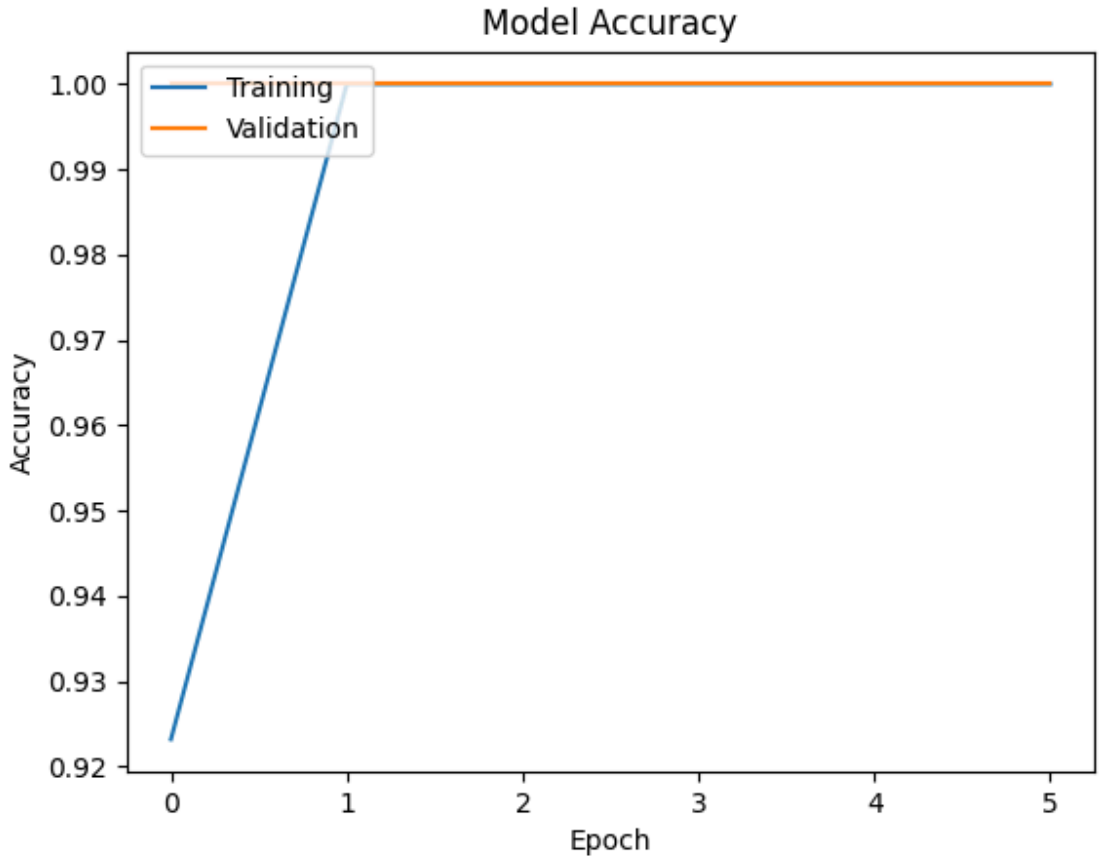

Figure 3: Model Accuracy

II. Reproducing the Simulation

The simulation of active users specified in the paper was reproduced using a Python script. As specified in the paper, approximately a quarter million user connection requests were generated in a 24-hour simulation, of which 40% was eMBB, 25% mMTC and 35% URLLC. The simulation uses a one second time discretization and maintains a queue for each Slice Type.

The TTL (Time To Live) for incoming traffic for each Slice Type is pre-defined. The paper did not specify the TTL used. So it was assumed to be 200 seconds for eMBB, 64 seconds for mMTC, and 125 seconds for URLLC. Note that these numbers are varied in the extension portion of this project.

The simulation iterates over the seconds in a 24-hour period. For every second, it randomly selects a number of new connections from a poisson distribution with lambda equal to the mean number of requests per second. It

randomly assigns the new connection to a Slice Type based on the distribution specified above. The simulation adds the new connection to the appropriate queue and records the time the connection expires (as specified by the TTL). Every second, the simulation also removes any expired connections.

Figure 4 shows a graph of the active user count in the network in a 24-hour period. The average user count for each Slice Type is specified in table 2 below. As specified in the paper, the number of users every second varies, and the network is able to appropriately allocate the number of required Slice Types.

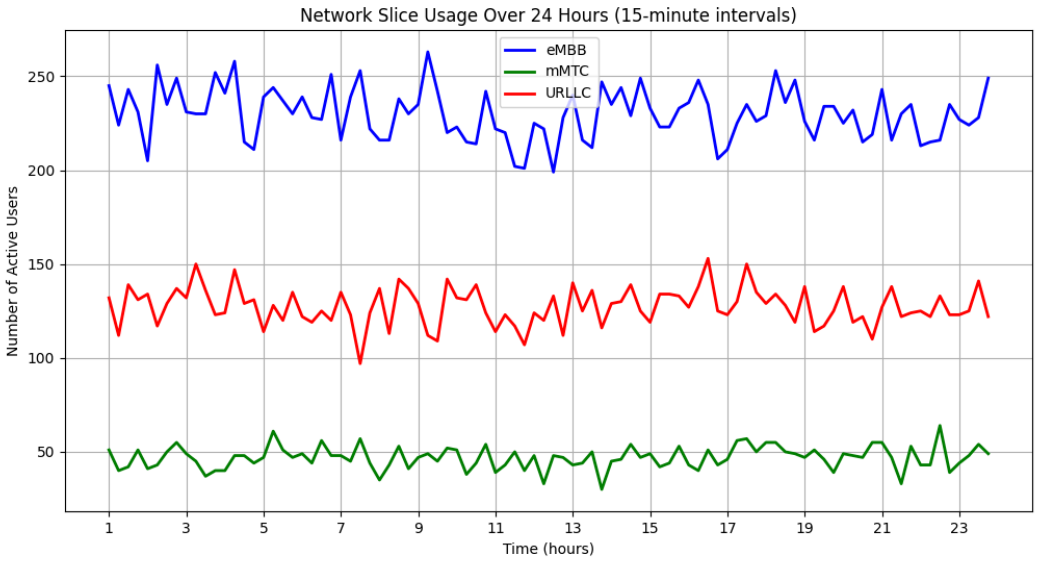

Figure 4: Active user count in the network in a 24-hour period

Table 2: Average user count

Slice Type	TTL (seconds)	Average User Count
eMBB	200	232
mMTC	64	46
URLLC	125	126

Extending the Paper

I. Standardizing the input

As specified in the “Reproducing the DL model” section, the input data was standardized to have a mean of zero and unit variance. This was implemented because different random initialization values resulted in significantly different accuracy results. The effect of standardization is ensuring that all features (KPIs) have a consistent scale resulting in a proportionate impact on model learning, and faster convergence and avoiding numerical instability. For reproducibility, the results without standardization are not included in this report.

II. Testing on a different dataset

A large dataset consisting of 466 thousand samples of KPIs and corresponding slice type was used to ensure that the model was not overfitting on the existing dataset and can be applied to a wider collection of inputs.

The exact same procedure specified in the “Reproducing the DL model” section was replicated with the larger dataset. When evaluated on the test dataset, the model had an accuracy of 100% and a loss of 0.01. The accuracy was identical to that of the model using the smaller dataset, indicating that the model perfectly predicts the appropriate Slice Type. The loss was significantly less than that of the model trained using the small dataset (0.1); this indicates that more data ensures that the model has more confidence in predicting the correct Slice Type.

Figure 5 shows the model loss every epoch when trained using the larger dataset, and figure 6 shows the model accuracy every epoch when trained using the larger dataset.

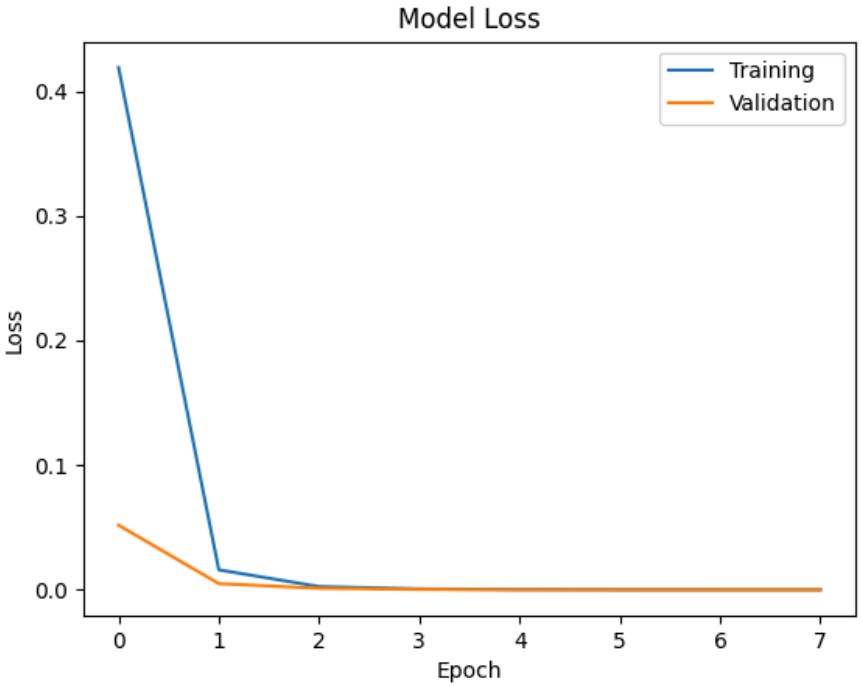

Figure 5: Model Loss using larger dataset

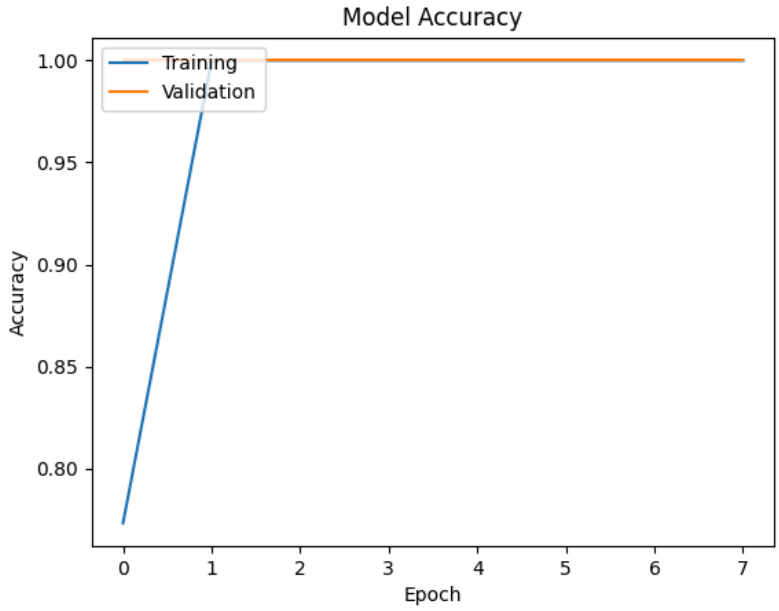

Figure 6: Model accuracy using the larger dataset

III. Simulating with different TTL values

Since the paper did not specify the TTL values used, the simulation was repeated with different TTL values. This was to determine the effect of TTL on the number of active users. The average number of active users can be used as a lower bound for the minimum required capabilities of the system for each slice type; the system must be able to at least handle the average number of active users. The maximum number of active users can indicate the size of a spike in users for a Slice Type.

Table 3 shows the specifications for the TTL values used in this experiment. Figures 7 and 8 show the average and maximum number of active users for the corresponding TTL values. For example, it can be seen that for TTL 5, the minimum system requirements should be at least able to handle 350 eMBB slices, 25 mMTC slices, and 200 URLLC slices, with the ability to handle spikes of up to 400 eMBB slices, 35 mMTC slices, and 240 URLLC slices (or a master slice that is able to handle the load till the spike ends).

Table 3: TTL Values used in experiment

Name	eMBB TTL (seconds)	mMTC TTL (seconds)	URLLC TTL (seconds)
TTL 1	200	64	125
TTL 2	150	80	100
TTL 3	250	50	150
TTL 4	100	100	100
TTL 5	300	30	200
TTL 6	180	90	110
TTL 7	220	70	130
TTL 8	130	110	90

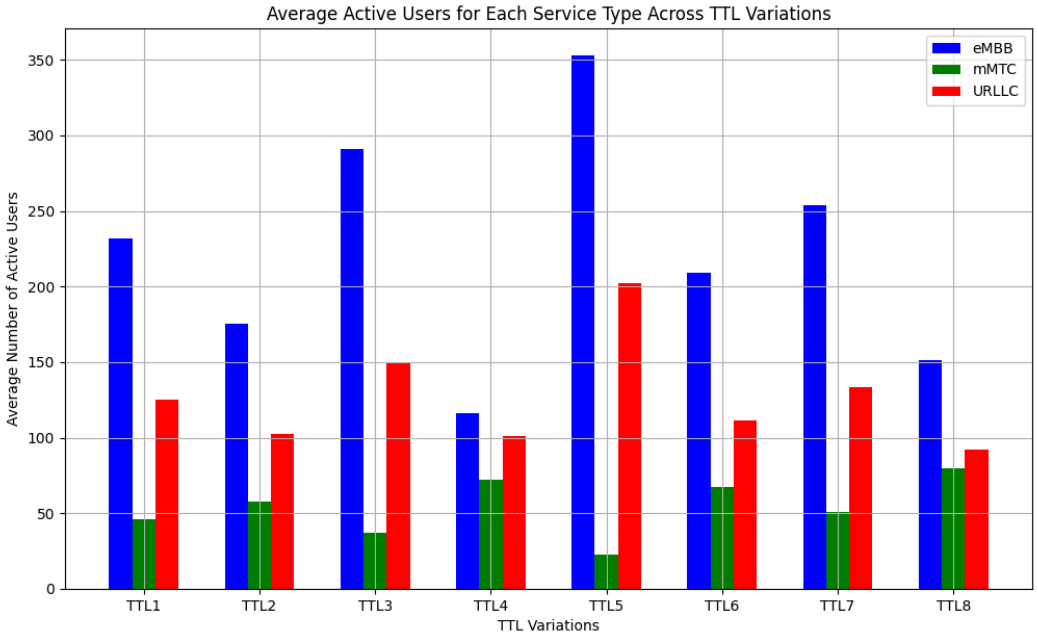

Figure 7: Average number of active users for different TTL values

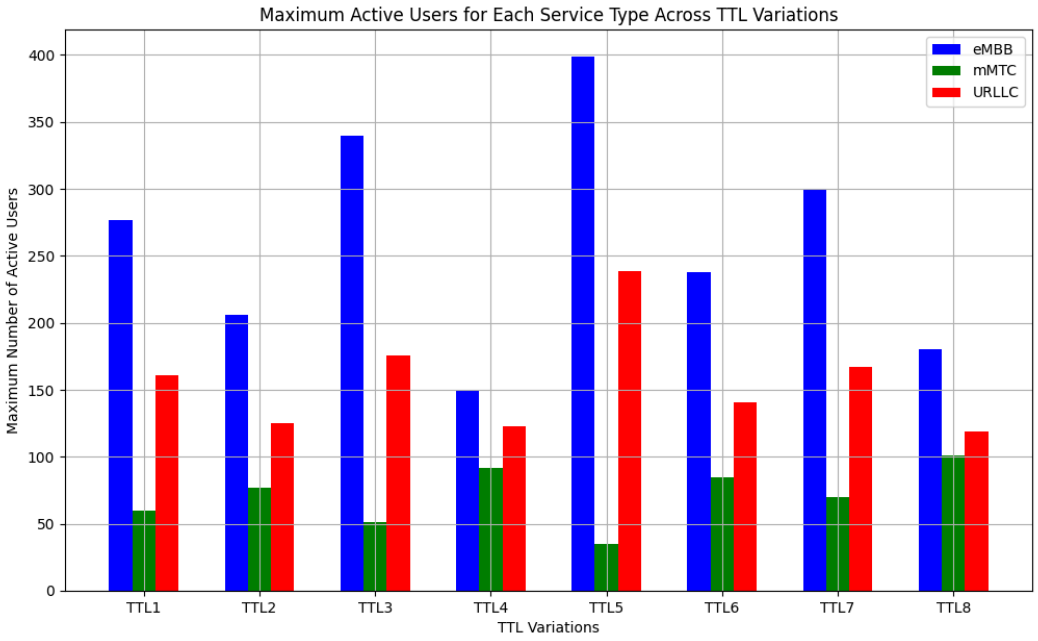

Figure 8: Maximum number of active users for different TTL values

Reference

[1] A. Thantharate, R. Paropkari, V. Walunj and C. Beard, "DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0762-0767