An ML-based Method for Efficient Network Utilization in Online Gaming using 5G Network Slicing

by

Peyman Saleh

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master
in
Computer Science

Examining Committee

The following served on the Examining Committee for this thesis.

External Examiner:	Chris Joslin Professor, Dept. of Engineering & Design, Carleton University
Internal Member(s):	Emil Petriu Professor, Dept. of Electrical Engineering and Computer Science, University of Ottawa
Supervisor(s):	Shervin Shirmohammadi Professor, Dept. of Electrical Engineering and Computer Science, University of Ottawa

Abstract

Online video gaming has become a ubiquitous aspect of modern-day video gaming. It has gained immense popularity due to its accessibility and immersive experience, resulting in millions of players worldwide participating in various online games. Depending on the type of gameplay, the players' quality of experience (QoE) in online video gaming can be significantly affected by network factors such as high bandwidth and low latency. As such, providers of online gaming services are competing to offer the highest quality of experience to their users at reasonable prices.

To achieve this objective, online game providers face two main challenges. Firstly, they must accurately estimate the network throughput capacity required to meet the servers' demands and ensure that the QoE is not compromised. Secondly, they must be able to secure the required throughput with network providers, which, in the current conventional network infrastructure, is neither agile nor dynamic. Thus, online game providers have to prepay for extra network throughput capacity or choose a cost-effective capacity that may result in potential QoE losses during peak usage.

To address these challenges, this thesis proposes a deep neural network-based model that utilizes a QoE-aware loss function for predicting the future network throughput demand. The model can accurately estimate the network throughput capacity required to maintain QoE levels while minimizing the cost of network resources. By doing so, online game providers can achieve optimal network resource allocation and effectively meet servers' demands.

Furthermore, this thesis proposes a slice optimizer module that employs 5G network slicing and a machine learning model to optimize network slices in a cost-efficient manner that satisfies both the online game provider's and the network provider's requirements. This module can dynamically allocate network resources based on the game provider's QoE requirements, the network provider's resource availability, and the cost of network resources. As a result, online game providers can efficiently manage network resources, optimize network slicing, and effectively control the cost of network resources.

Acknowledgements

I am humbled to express my immense gratitude to Professor Shervin Shirmohammadi, my supervisor, for his invaluable support, encouraging guidance, and motivational encouragement throughout my master's study. His unrelenting commitment and mentorship have shaped my academic career.

I also wish to thank Dr. Mahmoud Reza Hashemi for his insightful comments and collaborative research efforts. His contributions have significantly enriched my thesis, and I am deeply grateful for his partnership.

Furthermore, I would like to thank the Natural Sciences and Engineering Research Council (NSERC) and Swarmio Media for their generous funding and support of my research over the past two years. Their unwavering support has allowed me to pursue my academic goals with passion and dedication.

Finally, I must acknowledge the unwavering love, support, and patience of the most important person in my life, Zara. Her constant presence and encouragement have been the greatest help and source of motivation I could ask for. I am forever indebted to her and grateful for her love and unwavering support.

Table of Contents

List of Tables

List of Figures

Abbreviations

3GPP 3rd Generation Partnership Project 3, 4, 13

5QI 5G QoS Identifier 14

AI Artificial intelligence 3, 21

APH Average of the Previous Hour 58

ARMA Autoregressive integrated moving average 6, 7, 11

ARMA Autoregressive moving average 6

CDF Cumulative Distribution Function 40

CNN Convolutional neural networks 9

DL Deep learning 7

DNN Deep neural network 4, 8, 9, 16

DTR Decision Tree Regression 7

eMBB enhanced Mobile Broadband 14

GRU Gated recurrent unit 9, 10, 26, 28

HD The Highest Demand 58

HDPW The Highest Demand in the Previous Week 58

IIoT Intelligent Internet of Things 10

ISOC Inner Slice Over-capacity 19, 34, 41

ISP Internet service provider 9

KNN K-Nearest Neighbors 7

LSTM Long Short-Term Memory 7, 9, 10, 16, 26

MAE Mean Absolute Error 8, 29

MIoT Massive IoT 14

ML Machine learning 3, 4, 7, 11, 16, 17, 21

MLP Multi-Layer Perceptron 6–8

MLPWD Multi-Layer Perceptron with Weight Decay 6, 7

MSE Mean Square Error 29

NETSPRO 5G NETwork Slicing by PREdicting upcoming required link conditions in Online gaming 18

NFV Network function virtualization 13

OSPF Open Shortest Path First 10

QoE Quality of experience 1, 2, 4, 29

QoS Quality of service 1–3, 13

ReLU Rectified Linear Unit 29

RF Radio Frequency 16

RL Reinforcement learning 10, 16

RMSE root mean square error 8

RNN Recurrent neural networks 7, 10

SD Slice Differentiator 13

SDN Software-defined networking 10, 13

SGD Stochastic Gradient Descent 29

SLA Service-level agreement 2, 9, 16

SST Slice/Service Type 13, 34

STPW Same Time as the Previous Week 58

SVM Support Vector Machines 6–8

SVR Support Vector Regression 7, 8, 11

TanH Hyperbolic Tangent 29

TC Throughput Costlo 41

TDD Throughput Demand Distribution 39

URLLC Ultra-reliable Low Latency Communication 14

VMs Virtual machines 13, 16

VNFs Virtual network functions 13, 16

List of Symbols

α The ration of β/γ 31

c Assigned lable to a data point 36

$r(t)$ Available throughput capacity at time t 22

h' Candidate hidden state 28

k Number of clusters/slice options 35

C Number of configurable slice attributes 19

β Constant loss of under-provisioning 31

g Cost of throughput capacities 22

$d(t)$ The ground truth network demand's metrics of the server at time t 18

F Forecast horizon 18

h Hidden state 28

$W^{(z)}$ Input weights 28

X Input data for ML models 18

γ Monetary loss 31

$\sigma(a)$ The cost of a network with a set of attributes a 19

$\tilde{x}$ Prediction error 30

$p(t_F)$ Predicted network demand's metrics with forecast horizon F 18

Q Number of QoS metrics 18

μ Random initial clusters' centroids, 36

$U^{(z)}$ Recurrent weights 28

r_t Reset gate 27

S_i The ith slice option where $i \in \{1, \ldots, N\}$ 19

$\mathbf{L}$ List of throughput capacities 22

t Time slot 18

z_t Update gate 27

Chapter 1

Introduction

1.1 Motivation

1.1.1 Online gaming

Engaging in online video games with other players has become an integral aspect of many video games. Players from all around the world connect to game servers to partake in online video games. However, providing game servers presents a primary challenge, namely maintaining a consistently excellent Quality of experience (QoE). Moreover, the affordability of 4G and 5G cellular networks has increased in recent years, and Quality of service (QoS) reports indicate that the cellular network can deliver acceptable latency for online gaming. Consequently, players now have the option to acquire a 4G or 5G modem, enabling them to engage in gameplay without being confined solely to a fixed home internet connection.

It is important to note that QoE is a subjective assessment, and even within the context of online gaming, its definition can vary across different layers. In broader terms, QoE can be defined as the level of user satisfaction and enjoyment derived from playing a game. However, distinct factors within each layer of online gaming, including the game itself, the game server, and the network, can exert influence on a player's QoE [34]. The present study concentrates specifically on QoE from the network perspective. Online video games are susceptible to the prevailing network conditions, wherein factors such as delay, loss, and throughput at the network layer directly impact the player's QoE. For instance, in first-person shooting games, the downlink can reach up to 41.6 Mbps [8]. Moreover, even a minor increase in latency from 0 to 100 msec results in a substantial 35% decrease in the player's performance [12].

Various solutions have been introduced to enhance network latency and throughput, utilized by game providers and players. For instance, game providers distribute their servers geographically, enabling players to connect to the closest server and minimize latency [20]. However, depending on the time, game, and number of players, the servers' throughput demand may fluctuate.

One conventional approach to tackle changing throughput demands is to select a high-quality, high-throughput capacity dedicated network connection that guarantees the required network QoE. Nonetheless, this approach is inefficient and costly since the server may not always utilize the total available throughput capacity. Alternatively, game providers may choose a network connection with lower QoS and lower throughput capacity, such as the average server's throughput demand, which is less expensive. However, this may negatively impact the user's QoE when throughput demand exceeds the provided throughput capacity.

1.1.2 5G network slicing

The fifth generation of cellular technology, commonly known as 5G, has been designed to increase speed and reduce latency. These features offer an efficient communication platform with extended reach for online game providers. However, the advantages of 5G for game providers are more comprehensive than these features alone. 5G introduces a new architectural landscape with network slicing capabilities.

5G network slicing enables the multiplexing of virtualized and independent logical networks on the same physical network infrastructure. This unique capability allows network providers to offer a diverse range of services for different use cases on a physical network. For instance, a network tenant, such as a game provider, can request and purchase a specific type of network slice tailored to its service. The network provider adds the requested network slice and adapts and configures its physical network as needed.

By offering the network slice to the network tenant, the network provider guarantees a level of QoS through a Service-level agreement (SLA). Therefore, the network tenant can be sure that the network QoE is satisfied by the purchased network slice. Moreover, if the network requirements change, the tenant can request a new slice. For example, in the case of game servers, when the network demand increases, the game provider can purchase a slice with a higher throughput capacity.

1.2 Challenges and research problem

As stated in section 1.1.1, the server's throughput demand is subject to fluctuations based on various factors such as game state, level, concurrent players, e-sport events, and promotions. Accurate prediction of throughput demand enables game providers to make necessary adjustments to their network in anticipation of upcoming demand. This, in turn, leads to a more cost-efficient network with reduced QoS loss through the use of 5G network slicing. To model throughput demand, game providers can employ Artificial intelligence (AI) and Machine learning (ML) techniques using available data. Such a model would take into account several factors, including network demand, delay, jitter, packet loss rate, and other metrics that impact QoS. By analyzing past time slots using a sliding window approach, this model can predict these values for an upcoming time slot.

Once the future server's throghput demand has been determined, game providers can order a network slice with sufficient resources to cover the predicted demand. However, it is worth noting that from a network provider's standpoint, the enhanced flexibility offered by 5G network slicing comes at the expense of a more intricate network, which can be challenging to manage at scale. Although the 3rd Generation Partnership Project (3GPP) standard [17] provides for 2^{16} slice instances for each type of network slice, network providers can offer only a limited number of different slice instances due to management complexity. Hence, network providers restrict network tenants (i.e., game providers) to predefined network slices to reduce the complexity of the network.

The technical requirements for an ML-based model to determine the optimal network slice in the gaming context can be articulated as follows:

- *Prediction Accuracy*: This factor plays a crucial role in determining the effectiveness and cost-saving potential of the proposed model. However, unlike traditional traffic prediction models, an effective predictor for network slicing in the gaming context must account for the impact of under-provision on QoS.
- *Flexible Sliding Window Size*: The size of the sliding window has a direct bearing on the size and computational complexity of the model, as well as its accuracy. It also depends on the metrics employed for optimal slice prediction.
- *Flexible Forecast Horizon*: The forecast horizon, i.e., the future period for which the model should generate predictions, is determined by the specific application and the infrastructure of the 5G network provider (including the frequency and speed of new slice instantiation).

- *Context-aware loss function*: The loss function should meet the QoE-aware and cost-efficient network utilization requirements of the game provider using 5G network slicing.
- *Flexible number of 5G network slices*: The ability of a game provider to request a larger number of customized slices from the 5G network operator leads to greater efficiency in network slicing, translating to operational cost savings. The number of slices that can be provided is determined by the 5G network operator and the limitations of the maximum number of slices specified in the 3GPP standard.
- *Practical Computational Complexity*: The ML model is utilized to predict the next optimal slice for the server managed by the game provider. A near-real-time slice prediction necessitates a practical computational complexity, which implies a reasonable size for the ML model.

1.3 Contributions

The proposed scheme offers several significant contributions and innovations in the three main components of the proposed architecture. These can be summarized as follows:

- An innovative, parametric, and flexible problem formulation for the utilization of network slicing in online gaming that satisfies the technical requirements from a game provider's perspective.
- A Deep neural network (DNN) architecture optimized for time-series prediction, leveraging recent developments in neural network research.
- A selected context-aware loss function tailored to the QoE requirements of online gaming and e-sport, which improves the accuracy of network slice predictions.
- A method for determining the optimum set of slice configurations, taking into account the maximum available number of slices, to achieve efficient and cost-effective 5G network slicing. This approach improves the overall effectiveness of the proposed scheme.

1.4 Organization of thesis

The rest of this thesis is organized as follows.

Chapter 2- Background and Related offers an exposition of the foundational principles behind network condition prediction utilizing conventional approaches as well as more modern ML algorithms. Additionally, it provides an in-depth account of the advancements in 5G and network studies, and subsequently presents related research in this domain.

Chapter 3 - Network Slicing for online gaming formulates the challenge of utilizing network slicing for online game servers and proposes a high-level solution architecture to address this problem. Furthermore, it elucidates the modules of the proposed solution and their respective responsibilities.

Chapter 4 - QoE Aware prediction and network slicing expounds upon the proposed GRU-based DNN network condition prediction model and its benefits over traditional DNN solutions. It also introduces a QoE aware loss function that is optimal for online gaming scenarios where providing the minimum QoE is paramount. Subsequently, it describes the network slice optimizer module and explains the advantages of utilizing weighted K-means in this module.

Chapter 5 - Performance Evaluation proposes a cost model based on the network and QoE cost. Furthermore, it introduces the dataset used for this study and elucidates the feature engineering process for incorporating new features into the dataset, including time. Additionally, it describes the experiment setup and conducts a comprehensive experiment for optimizing the proposed module. Finally, it conducts a performance evaluation and demonstrates the cost efficiency of the proposed solution while providing the required QoE as compared to other widely used techniques in the industry.

Chapter 6 - Conclusion and Further Works include a conclusion and a vision of future works.

Chapter 2

Background and Related Works

2.1 Throughput prediction

Network throughput prediction can be classified into two main methodologies: model-driven and data-driven approaches. Model-driven methods, such as Autoregressive moving average (ARMA) [26] and Autoregressive integrated moving average (ARMA) [18], rely on mathematical models derived from the network and its surrounding environment. However, these models are built upon certain assumptions that may not accurately represent real-world conditions and might struggle to adapt to dynamic changes in the network environment [42]. Consequently, model-driven approaches are commonly utilized for predicting network throughput in scenarios where the statistical properties, including mean, variance, and autocovariance, remain constant over time [13]. In essence, these methods assume that the distribution and characteristics of the data remain unchanged as time progresses.

On the other hand, data-driven techniques such as machine learning and deep learning are successful in identifying nonlinear attributes of network throughput. Authors in [40] conducted an experiment comparing the performance of Multi-Layer Perceptron (MLP), Multi-Layer Perceptron with Weight Decay (MLPWD), Support Vector Machines (SVM) in network traffic prediction. The authors had access to an extensive hourly logged dataset of wireless network cells, which provided them with over 24 features for each cell over 168 hours. The results indicate that SVM outperforms both MLP and MLPWD. The authors also found that "using multidimensional traffic datasets significantly increases the prediction accuracy of MLP, MLPWD, and SVM algorithms" [40]. However, the use of MLP, MLPWD, and SVM is limited to throughput prediction, and the shortest forecast

horizon in their experiment was only one hour. This may not be ideal for many latency-sensitive applications.

On the other hand, data-driven techniques such as ML and Deep learning (DL) are successful in identifying nonlinear attributes of network throughput. Authors in [40] conducted an experiment comparing the performance of MLP, MLPWD, SVM in network traffic prediction. The authors had access to an extensive hourly logged dataset of wireless network cells, which provided them with over 24 features for each cell over 168 hours. The results indicate that SVM outperforms both MLP and MLPWD. The authors also found that "using multidimensional traffic datasets significantly increases the prediction accuracy of MLP, MLPWD, and SVM algorithms" [40]. However, the use of MLP, MLPWD, and SVM is limited to throughput prediction, and the shortest forecast horizon in their experiment was only one hour. This may not be ideal for many latency-sensitive applications where they require constant network optimization to prevent any service disruption. Authors in [1] explore the use of machine learning techniques to predict downlink throughput in 4G-LTE networks. The authors aim to address the challenge of accurately estimating the expected throughput experienced by users in mobile networks. The study begins by analyzing the factors that influence downlink throughput in 4G-LTE networks, such as signal strength, signal-to-noise ratio, and network congestion. They then propose a machine learning-based approach to predict downlink throughput based on these factors. The study then propose a machine learning-based approach to predict downlink throughput based on these factors. The results indicate that the machine learning models can effectively predict downlink throughput, with certain algorithms such as K-Nearest Neighbors (KNN) and Decision Tree Regression (DTR) outperforming others in terms of accuracy. The authors in [13] address the increasing demand for network enhancements due to the growing network usage and emphasize the importance of analyzing network quality and performance using real-world data, with throughput serving as a key performance indicator. The study explores two main approaches for throughput analysis: classical machine learning and time series forecasting. In the machine learning approach, several models are evaluated for throughput prediction. These models include Support Vector Regression (SVR), KNN for regression, Ridge Regression, and Random Forest for regression. The performance of each model is assessed in terms of their ability to accurately predict throughput. Furthermore, the paper explores time series forecasting models for predicting future throughput values. The time series forecasting techniques employed both traditional statistical methods and advanced deep learning architectures. These include ARMA models, Recurrent neural networks (RNN)s, and Long Short-Term Memory (LSTM) networks. The performance of these time series models is then compared to the classical machine learning models to assess their efficacy in throughput prediction. Among the tested models, the Random For-

est model emerges as the most effective, achieving the highest prediction performance in terms of throughput. However, the authors indicated that, lack of enough training data was one of the main reasons that the DNN models performed poorly. The authors in [32] explore the prediction of mobile bandwidth and handoff events in the context of 4G/5G networks. The authors aim to develop real-time prediction models to improve the quality of service and user experience in mobile communication systems. The study begins by discussing the importance of accurate bandwidth and handoff predictions in dynamic and heterogeneous 4G/5G networks. They highlight the need for timely and reliable predictions to enable seamless communication and efficient resource allocation. The authors propose a prediction framework called Temporal pattern attention LSTM (TPA-LSTM) that combines historical network measurements and machine learning techniques. They collect real-world data comprising network parameters such as signal strength, signal-to-noise ratio, and user mobility patterns. Various machine learning algorithms, including SVR and random forest (RF), are employed to develop predictive models. The performance of the models is evaluated using metrics such as Mean Absolute Error (MAE) and root mean square error (RMSE). The results demonstrate the effectiveness of the proposed framework in accurately predicting mobile bandwidth and handoff events in real-time scenarios. Furthermore, the authors investigate the impact of different features and model configurations on the prediction performance. They analyze the trade-offs between prediction accuracy and computational complexity, providing insights into the optimal design choices for real-time predictions in 4G/5G networks.

Authors in [35] conducted a data collection exercise across various use cases such as city and suburban driving, a subway ride and mining, and home and office environments, for both LTE and 5G non-standalone networks. Subsequently, they proposed a model to evaluate popular machine learning algorithms such as XGboost, MLP, and SVM for throughput prediction, using the collected data. The parameters measured in this study included reference signals such as RSRQ [1], CQI [2], SINR [3], and past throughput measurements, which are typically not accessible to network tenants and thus, impractical for game providers. Additionally, the forecast horizon in this study was less than two seconds, which is inefficient, if not impractical in the context of 5G network slicing.

Machine learning methods rely heavily on feature engineering and perform poorly when working with high-dimensional data. Deep learning prediction methods are superior to machine learning methods, as they tend to retain learning characteristics, ensure relevance between tasks, and effectively address time series problems. The work in [50] proposed

[1]Reference Signal Received Quality

[2]Reports the current channel quality, which is used to determine the transport block size

[3]Signal to Interference plus Noise Ratio of the signal carrier

a DNN throughput prediction method that demonstrated the advantages of various deep learning architectures. Using Convolutional neural networks (CNN), fully connected neural networks, and recurrent neural networks (RNN), the authors showed that DNN can determine the importance of inputs in large-scale data points.

Regarding throughput prediction, spatial and temporal features are crucial in enhancing prediction accuracy, depending on the nature of the problem and available data points. Spatial features denote the correlation between two attributes of the data point influenced by their location, while temporal features define the time-varying attributes of the data point. By employing image processing techniques, CNN, authors in [54] processed the throughput data of a city, extracted the spatial and temporal features, and obtained good performance in throughput prediction.

Authors in [53] proposed a precise cellular traffic forecasting architecture, referred to as a Double Spatio-Temporal Neural Network (D-STN), which employs a lightweight mechanism for combining the STN output with historical statistics, thereby improving long-term (1-10 hour) prediction performance. The proposed model was trained with 40 days of collected data, after which it was given 2-10 hours of data and predicted up to a 10-hour forecast horizon. Although the model provided more accurate predictions than traditional approaches such as ARIMA, MLP, and SVM, it failed in short-term predictions. The CNN models are capable of extracting the spatial characteristics of network throughput. However, they are generally not practical for predicting the throughput of a single area, such as a server where geographical data correlation are unrelated.

Authors in [5] proposed a deep learning-based data analytics tool, DeepCog, for predicting network capacity or throughput at the slice level instead of aggregated. The study's main advantage was its loss function, which "could reflect a variety of economic cost strategies via a single parameter". [5]. The authors designed the loss function to prevent SLA violation and resource overprovisioning in each network slice, which corresponds to a service and its requirements. Using a dataset from an Internet service provider (ISP) and a LSTM neural network, the work in [21] proposed a method for short-term (less than 30 seconds) traffic prediction. The authors first introduced a set of features for throughput clustering (TCP, UDP, etc.) and then used an LSTM model for throughput prediction for each cluster. Their experiment demonstrated that LSTM is a competitive DNN model for throughput prediction. However,their solution was designed for predicting throughput for only one slice, which limits the solution for multiple slice usage.

The Gated recurrent unit (GRU) is a type of gating mechanism used in RNNs, which addresses the issue of vanishing gradients commonly encountered in standard RNNs. It shares similarities with the LSTM unit but differs in the absence of an output gate. In

2014, Kyunghyun Cho et al. [10] introduced GRUs as an alternative to LSTMs, featuring a forget gate while having fewer parameters due to the lack of an output gate. The work in [42] proposed a RNN architecture that outperformed LSTM, and GRU models in a small sliding window of 5 seconds, making it a promising candidate for near real-time throughput prediction. Nonetheless, the RNN model was found to be more susceptible to overfitting in their evaluation compared to other models. Consequently, the authors suggest that LSTM and GRU are better suited for throughput prediction.

The authors in [14] propose a deep learning-assisted traffic prediction model for hybrid Software-defined networking (SDN)/Open Shortest Path First (OSPF) backbone networks. The authors first introduce the hybrid SDN/OSPF backbone network architecture and the challenges of traffic prediction in such networks. They then propose a deep learning-assisted traffic prediction model that uses a LSTM and GRU to compare their performance. Their experimental evaluation shows that LSTM performs better in long-term predictions but GRU can outperform LSTM in short-term predictions.

In recent years, Reinforcement learning (RL) has emerged as a popular approach in ensemble learning research, with the aim of maximizing returns or achieving specific goals through learning strategies. Authors in [39] proposed an RL solution that models the network throughput prediction problem as a Markov decision process. The paper also proposes a residual-based adaptive dictionary learning algorithm to reduce computational complexity, resulting in lower energy consumption for the Intelligent Internet of Things (IIoT) network. The authors in [4] presents a novel approach to enhance network traffic prediction using an advanced deep reinforcement learning algorithm. The authors aim to improve the accuracy and efficiency of predicting network traffic patterns, which is crucial for network management and resource allocation. The study introduces a deep reinforcement learning-based method that combines the power of deep learning techniques with reinforcement learning algorithms. The proposed method leverages historical network traffic data to train a deep neural network model, which is then used by a reinforchement learning agent to make predictions about future network traffic patterns. The authors conduct experiments to evaluate the performance of their proposed method and compare it with traditional traffic prediction approaches. They analyze various metrics, including prediction accuracy and computational efficiency, to assess the effectiveness of the enhanced deep reinforcement learning algorithm. The results demonstrate that the proposed method outperforms traditional approaches in network traffic prediction. The enhanced deep reinforcement learning algorithm achieves higher accuracy in forecasting future traffic patterns, enabling better network management decisions and resource allocation. Additionally, the method showcases improved computational efficiency, allowing for faster predictions without sacrificing accuracy. The authors in [6] propose a novel approach to predict bandwidth

availability in real-time communication systems. The authors aim to address the challenge of accurately estimating future bandwidth conditions to optimize the quality and performance of real-time applications. The study introduces a hybrid approach called BoB, which combines heuristic methods and reinforcement learning techniques for bandwidth prediction. The heuristic component leverages historical network measurements to identify patterns and trends in bandwidth availability. The reinforcement learning component uses an agent to interact with the network and learn optimal actions to predict future bandwidth. The authors design experiments to evaluate the performance of BoB in different scenarios, including varying network conditions and application workloads. They compare the accuracy of BoB with other existing bandwidth prediction methods, such as ARMA and SVR. The results demonstrate that the BoB approach outperforms traditional methods in predicting bandwidth for real-time communications. It achieves higher prediction accuracy, leading to improved quality of service and user experience. The reinforcement learning component of BoB adapts to changing network conditions and learns effective strategies for accurate bandwidth estimation. Despite the promising results demonstrated by RL models in predicting throughput, they possess two notable disadvantages. Firstly, these models are limited in their ability to predict only the next time slot, making them practical solely for short-term throughput prediction scenarios. Secondly, their performance is relatively poor during the initial stages of prediction due to the lack of sufficient available data.

Table 2.1 presents a comparison of the proposed scheme with state-of-the-art methods based on the technical requirements of a ML-based model for efficient network utilization in online gaming using 5G network slicing.

Table 2.1: Comparison of related works in throughput predicton

Related works	Input parameters for prediction	Predicted Parameters	ML Method	Sliding Prediction Window Size	Forecast Horizon	Design Perspective	Flexible number of slices
[5]	Vary, depending on the location of the prediction	Throughput	CNN	Flexible	Flexible	Network Provider	No
[53]	Throughput, geographical region	Throughput	ConvLSTM, 3D-ConvNets	2-10 h	2-10 h	Network Provider	N/A
[42]	Throughput	Throughput	RNN	N/A	5 sec, 5 min	Network Provider	N/A
[39]	Throughput, geographical region	Throughput	RL	10-15 min	10-15 min	Network Provider	N/A
[30]	Throughput	Throughput	ARIMA/RNN	N/A	5 min- 1 day	Network Provider	N/A
[40]	Reference signals (RSRQ, CQI, etc.), throughput, GPS coordinates	Throughput	MLP, MLPWD/SVM	2-10 h	1 h	Network Provider	N/A
[35]	Reference signals (RSRQ, CQI, etc.)	Throughput	XGBoost, MLP, SVR	500-1500 ms	500-1500 ms	Network Provider	No
[27]	Throughput	Throughput	LSTM	90 Sec	30 Sec	Network Provider	N/A
[6]	packet-level data, including the receiving rate, packet delay, packet loss, throughput	Throughput	DRL	200 ms	5-200 ms	Packet Receiver	N/A
[14]	Throughput	Throughput	LATM/GRU	150 min	15 min	Network Provider	N/A
[13]	Reference signals (RSRQ, CQI, etc.), throughput, GPS coordinates	Throughput	SVR/KNN/RR	8 h	8 h	Network Provider	N/A
[1]	Reference signals (RSRQ, CQI, etc.), throughput, GPS coordinates	Throughput	SVR/LR/KNN/DTR	2 sec	2 sec	Network Provider	N/A
[4]	Variance of packet, actual data size, client port number,	Throughput	DRL	N/A	N/A	Network Provider	N/A

2.2 Network Slicing

Network slicing is a noteworthy innovation of 5G that is expected to carry over to the upcoming generation of mobile cellular networks, 6G [15, 51]. This technology allows for the creation of multiple completely isolated virtual networks using the same physical infrastructure, thereby enabling network providers to offer customized network slices with different QoS requirements. The implementation of network slicing is made possible through the use of techniques such as SDN and Network function virtualization (NFV). In the cloud computing paradigm, SDN utilizes a centralized controller for resource orchestration and network flow management [52]. On the other hand, NFV implements network functions, such as firewalls and load balancers, as software, and 5G architecture treats them as Virtual network functions (VNFs) that operate as Virtual machines (VMs) on servers.

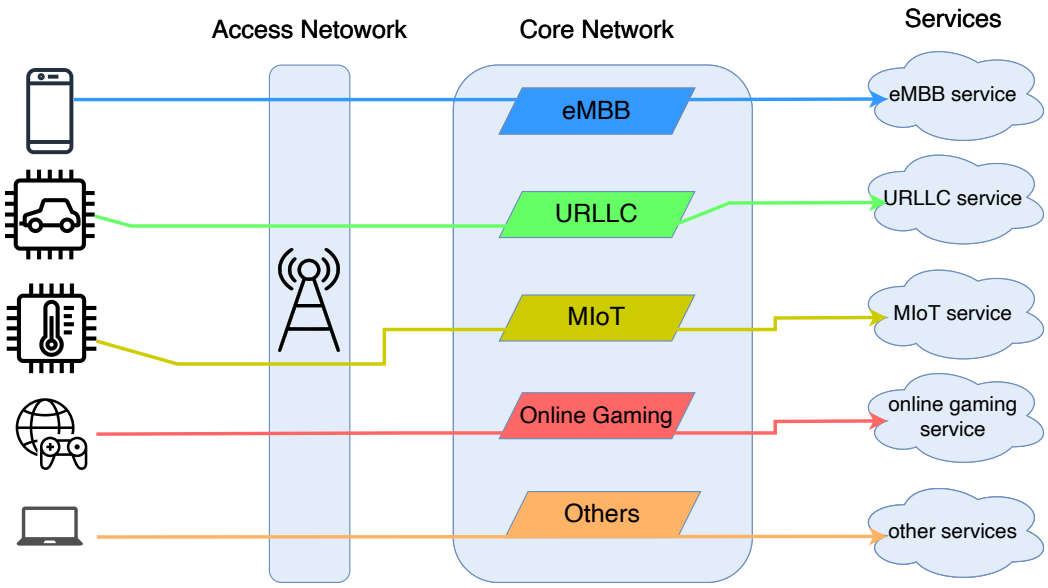


Figure 2.1: Network Slicing in 5G – High Level Architecture

The 3GPP [17] has recognized network slicing as a significant characteristic of 5G networks. The definition of a network slice includes two components: the Slice/Service Type (SST) and the Slice Differentiator (SD). The SST establishes the set of features for a given network slice, and 3GPP has allocated an 8-bit value to define the SST, providing for a total of 256 SSTs. The SD provides additional information that differentiates a network slice. To differentiate network slices based on the same SST, 3GPP has designated 16 bit bits for the SD. Out of 256 SST, 128 SSTs are reserved, and some have been standard-

ized by 3GPP, including enhanced Mobile Broadband (eMBB), Ultra-reliable Low Latency Communication (URLLC), and and Massive IoT (MIoT). Each SST represents a common use case in the industry [3], and a set of 5G QoS Identifier (5QI) numbers are associated with each standardized SST. The 5QI values indicate a set of minimum requirements, such as default priority level, packet delay budget, and packet error rate. A NEtwork slice type (NEST) is a SST and SD filed with values.

A network provider may offer a NEST that satisfies or exceeds these requirements. Table 4.3 presents an example of standardized and non-standard NESTs, while Table 2.3 depicts a portion of the "Standardized 5QI to QoS characteristics mapping" that 3GPP has defined as NESTs QoS blueprints. For instance, if we consider the online gaming SST, a network provider can offer ten gaming NESTs, each with different maximum throughputs capacity (10 Mbit, 100 Mbit, etc.), and all would meet the specified 5QI. Figure 2.1 depict a high level architecture of network slicing in 5G.

Table 2.2: NEST examples and their 5QI values

NEST	Standardized	Slice quality of service	Value
eMBB	Yes	3GPP 5QI	1,2,5,6,7,8, 9
URLLC	Yes	3GPP 5QI	82
MIoT	Yes	3GPP 5QI	9
Online Gaming	No	3GPP 5QI	3

Table 2.3: Sample of standardized 5QI to QoS characteristics mapping [17]

5QI Value	Default Priority Level	Packet Delay Budget	Packet Error Rate	Example Services
1	20	100 ms	10^{-2}	Conversational Voice
2	40	150 ms	10^{-3}	Conversational Video (Live Streaming)
3	30	50 ms	10^{-3}	Real Time Gaming, V2X messages (see TS 23.287) Electricity distribution – medium voltage, Process automation monitoring
4	50	300 ms	10^{-6}	Non-Conversational Video (Buffered Streaming)
5	10	100 mm	10^{-6}	IMS Signalling
6	60	300 ms	10^{-6}	Video (Buffered Streaming) TCP-based (e.g. www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
7	70	100 ms	10^{-3}	Voice,Video (Live Streaming) Interactive Gaming
8	80	300 ms	10^{-6}	Video (Buffered Streaming) TCP-based(e.g. www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.)
9	90			
10	90	1100 ms	10^{-6}	Video (Buffered Streaming) TCP-based (e.g. www, e-mail, chat, ftp, p2p file sharing, progressive video, etc.) and any service that can be used over satellite access type with these characteristics
82	19	10 ms	10^{-4}	Discrete Automation

The authors in [38] contend that the commencement and configuration of VNFs on new VMs can consume several minutes. Hence, to attain a dynamically scalable network slicing in the context of 5G, the network must possess the ability to predict future resource requirements. To overcome this issue, the authors introduced a Bi-directional LSTM (Bi-LSTM)-based model to forecast VNFs' resource usage in network slicing. They assessed their methodology using Open source Mano, an implementation of NFV [31]. The study in [41] proposed a dynamic network slicing technique to enhance network utilization and profitability. To prevent network congestion or resource wastage, the authors introduced two thresholds for a slice leased to a tenant. The initial threshold alerts the network management system of the escalating slice usage, resulting in the allocation of additional resources to the slice and avoiding any SLA violation. The second threshold notifies the network management system of the possibility of resource wastage due to a sudden reduction in slice usage. The authors in [49] split the network slicing problem into two subproblems. The first subproblem operates at a large timescale for reserving a resource for a slice, known as the network slicing planning stage. The paper employs a RL approach to solve this subproblem. The second subproblem operates at a small timescale to dynamically allocate reserved resources to tenants, known as the network operation stage. The authors resolved this problem using an optimization algorithm. In [46], three use cases were investigated for network slicing. The first scenario is to assign the appropriate slice type to an unknown device. The authors proposed a Radio Frequency (RF) model that utilizes the device type and the device's network requirements to make slice selection for the unknown device. The second scenario is load balancing and network slice utilization, and the third is slice failure. The paper introduced a DNN model to monitor slice usage and assign a backup slice for load balancing or replace the failed slice. The authors in [47] posited that wireless channel capacity is constrained, and several variables, such as the distance between nodes or physical barriers, can affect capacity. The authors proposed a ML model to predict medium usage and ascertain if a new slice with the requested parameters can be deployed.

Chapter 3

Network slicing for online gaming

3.1 Introduction

As mentioned in section 2.2, although network providers have the capability to offer 10^{24} individual network slices for each NEST, managing such a complex network presents a significant challenge [49]. Therefore, this thesis assumes that a set of network slices is negotiated between the 5G network provider and the tenant, which can then switch between slices as needed. Specifically, in the context of online gaming, a game provider must negotiate a set of 5G network slices for each game server and utilize each network slice at the appropriate time. To accomplish this, the online game provider must address two key problems.

Problem 1: Network condition prediction, which involves predicting the network condition for the next time slot with minimal loss of QoE. The frequency and preparation time required for switching between network slices depends on the capability of the network provider. In addition, the length of game sessions is another attribute that can impact the length of the time slots. The average gaming session is between 4-6 minutes, depending on the type of game [16]. Therefore, addressing this problem requires designing a context-aware cost-efficient DL model that can predict the network conditions for the next time slot.

Problem 2: Slice optimization, which entails minimizing the monetary cost of the network slices without sacrificing QoE. In this problem, the configuration of slices is affected by a server's demand history and the number of slice options available. To solve this problem, an ML model is designed that can optimize a dynamic number of slices.

After addressing the aforementioned subproblems, network conditions for the next time slot and available network slices can be predicted. Accordingly, a slice selector module can then choose one of the slice options for the next time slot based on the prediction. The proposed solution architecter is a 5G NETwork Slicing by PREdicting upcoming required link conditions in Online gaming (NETSPRO). The main components of the proposed 5G network slicing scheme and their relationships are illustrated in figure 3.1

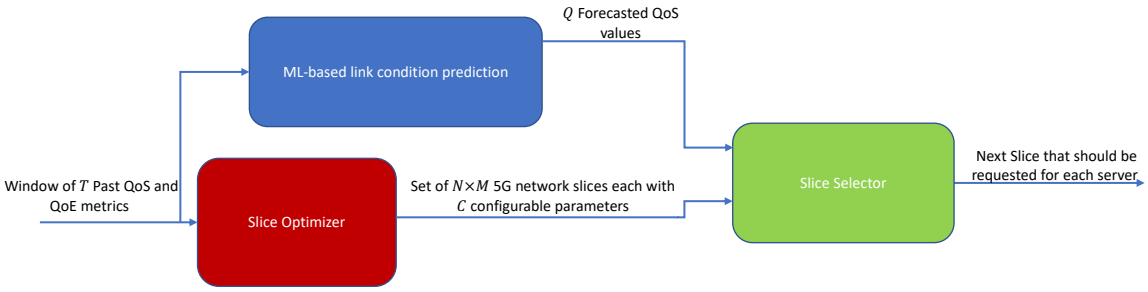


Figure 3.1: The high-level architecture of NETSPRO

3.2 Network condition prediction

Let us divide time into discrete time slots, represented by the variable t. The duration of each time slot is a configurable parameter, allowing the proposed scheme to accommodate the constraints of any 5G operator. Let $d(t)$ be a $1 \times Q$ vector comprising network demand metrics of the game server at time slot t, where Q denotes the number of QoS metrics employed as input X for the predictor. The set of demand metrics may comprise throughput, delay, jitter, packet loss rate, or other metrics that impact QoE. For the purpose of evaluation in section 5, we assume that $d(t)$ represents the server's throughput demand at time slot t, without loss of generality. The objective is to design, train, and employ a new deep learning-based model, featuring a QoE-aware loss function, to solve the network condition prediction problem with high accuracy and minimal QoE loss. The proposed network prediction scheme's primary parameters are presented in 4.3.1.

Let $p(t_F)$ be the vector of predicted network demand metrics of the server at time slot t_F. Here, F denotes the forecast horizon, which is the future period for which the model is making predictions. The network condition prediction problem can then be formulated as computing $p(t_F)$ based on knowledge of the current and T previous network condition

snapshots $d(t), d(t-1), \ldots, d(t-T)$. The accuracy of the predicted $p(t_F)$ relative to the ground truth $d(t_F)$ is measured by a loss function $l(p(t_F), d(t_F))$.

3.3 Slice Optimization

The cost of a network slice is determined by its set of attributes such as throughput, delay, and packet loss. Let $S_i, i \in \{1, \ldots, N\}$ denote the network slices with C number of attributes for the game server based on the request, and $a_i^j, j \in \{1, ..., C\}$ represent a slice's attributes.The cost of a network slice is computed using the cost model $\sigma(a)$ proposed in section 4.2, which is based on the slice's attributes. As stated previous section, without loss of generality, we assume that a represents slice throughput capacity. Figure 3.2 depicts a scenario where the actual throughput demand is $d(t) = 15\,\text{Mbit/s}$ and the assigned slice (S_1) provides throughput capacity of $a = 20\,\text{Mbit/s}$. Thus, the slice throughput capacity exceeds the actual demand, resulting in Inner Slice Over-capacity (ISOC). The ISOC cost is determined as $(\sigma(a) - \sigma(d(t))$. The goal of the slice optimizer is to provide a set of slices that:

1. Ensure that at least one slice can provide the required capacity for each server throughput demand $d(t)$.
2. Minimize the ISOC cost.

The optimum solution depends on the maximum number of available network slices N, the cost structure of the slices, the frequency at which slices can be changed, and the configurable attributes of each slice, all of which are determined by the network operator and are known priori. The optimized set depends also on the pricing policies of the game provider and the nature of its traffic. The Slice Optimizer should be implemented with an ML model. Depending on the type and how it changes with time, the slice set may be updated periodically by reusing the slice optimizer.

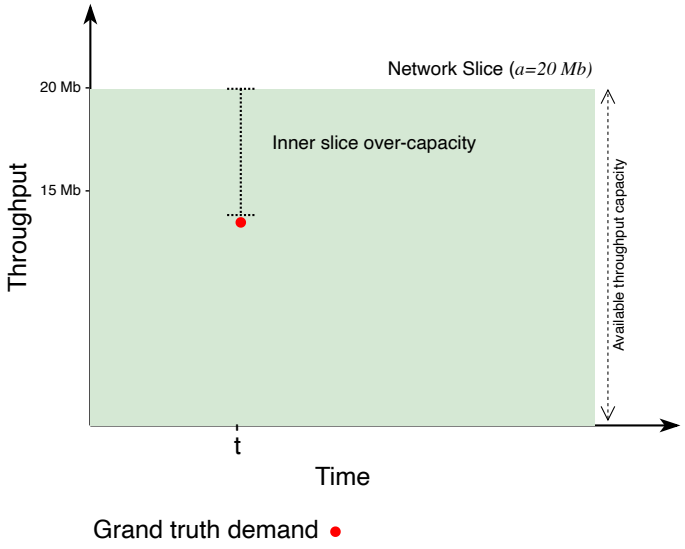


Figure 3.2: An example of ISOC occurs when the available network capacity provided by a slice is more than the demand.

Chapter 4

QoE Aware prediction and network slicing

4.1 Introduction

AI tools and platforms have exhibited significant potential and are extensively utilized for predictive analytics. AI solutions can be implemented across various facets of 5G network domains, including network planning, network diagnostics and insights, and network optimization and control [22], as demonstrated in the literature. Nonetheless, no existing method satisfies all the technical prerequisites of an efficient and practical ML-based solution to determine the optimum 5G network slices in the context of online gaming. This thesis proposes novel ML-based solutions to predict game server network demands and select a suitable 5G network slice, as discussed and illustrated in section 3.2. The proposed solution comprises of three primary components: an DL-powered server link condition prediction, an ML-powered slice optimizer, and a slice selector. In this section, first, we propose a new cost model for optimizing and evaluating the performance of the proposed and existing solutions. Then we explain each component in greater detail, highlighting their unique features and innovations.

4.2 Cost model

The evaluation of machine learning solutions has traditionally been focused on determining the difference between the true values and the predicted values. However, when it

comes to network slicing in online gaming, the issue becomes much more complex. The cost of network services provided by network providers is not linear, which makes it insufficient to calculate the difference between the true throughput demand and predicted throughput demand using distance-based methods. Additionally, network slicing increases the complexity of calculating the network cost due to considerations such as slice type, infrastructure usage, and time. In addition to network cost, QoE cost is another factor that can impact the overall cost. QoE cost is primarily influenced by network capacity and network delay. Therefore, in this section, we have developed a cost model for calculating the total cost, denoted as $TC(\cdot)$, taking into account both network cost and QoE cost.

Network Cost

The monetary cost of the network is denoted as $\sigma(\cdot)$, which represents the cost incurred by the game provider for the network services. Several network attributes, such as maximum throughput capacity, delay, jitter, and packet loss rate, can impact the final monetary cost of a network.With the implementation of 5G network slicing, a certain level of QoS can be guaranteed for a network slice (2.3). Hence, it can be safely assumed that the slice's throughput capacity is the primary network attribute that can influence the cost of a NESTs. However, other non-network attributes, including the time of the request, NEST demands, and region, can also affect the slice cost [9]. Unfortunately, there is no available database that provides network slicing costs for the present study. Additionally, no public information exists on how network providers compute network costs based on throughput and and as we can see from Figure 4.1 the relation between the network cost and the throughput capacity in not linear. Therefore, to assess network cost, we have used AWS Direct Connect pricing [2]. A portion of this cost table is presented in the table 4.1. Let assume that L is a list of throughput capacity specified by network provider and g is the associated cost of each item in the list. Therefore, the cost of a network with throughput capacity of a is calculated using the following equation:

$$\sigma(\cdot) = g_{\min\{x \ in \ L \ | \ x>a\}} \tag{4.1}$$

Recall that we denoted by $r(t)$ the available throughput capacity at time slot t. Therefore, $\sigma(r(t))$ determines the network cost of each time slot.

Table 4.1: AWS Direct Connect pricing, 2023

Capacity	**Port rate**
50 Mbps	$0.03/time slot
100 Mbps	$0.06/time slot
200 Mbps	$0.08/time slot
300 Mbps	$0.12/time slot
400 Mbps	$0.16/time slot
500 Mbps	$0.20/time slot
1 Gbps	$0.33/time slot

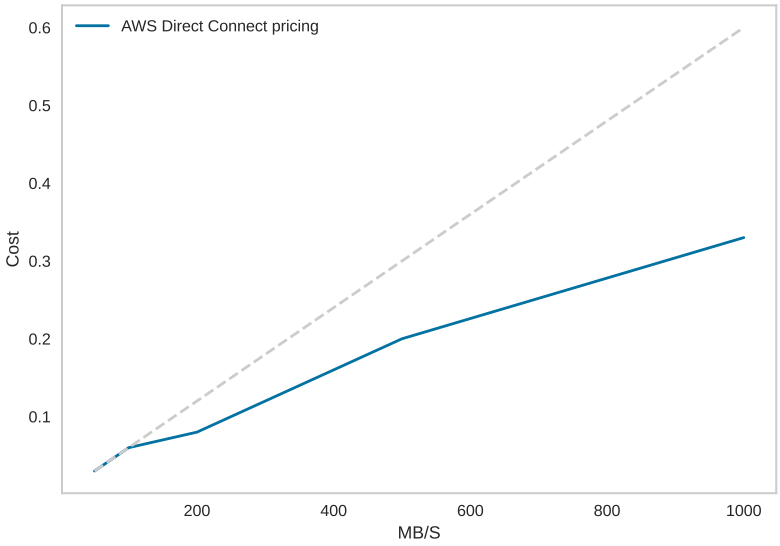


Figure 4.1: AWS Direct Connect pricing comparison

QoE Cost

The QoE cost is an expense that game providers must bear. Online gaming is influenced by various factors that impact QoE, including frame rate, throughput, packet loss, and delay [25]. As per the Network Cost section's explanation 4.2, we can exclude packet loss and delay as network-related factors. Consequently, throughput becomes the primary factor in estimating QoE cost. The only situation in which network throughput can affect QoE is when the server's throughput demand exceeds the available throughput capacity. In such cases, the game provider may incur QoE costs. The QoE cost model denoted by

$\delta(\cdot)$ and based on the network cost $\sigma(\cdot)$ is formulated to calculate a tangible estimation of this cost. The QoE cost at time t is calculated as follows:

$$\delta(t) = (\sigma(d(t)) - \sigma(r(t))) \times \lambda \tag{4.2}$$

λ defines the QoE importance factor which is varied between online game providers. In this evaluation we considered the value of $\lambda = 10$.

In accordance with the available network's throughput capacity at time t and the server's throughput demand, the total cost can be quantified as follows:

- If the actual server's throughput demand $d(t)$ surpasses the provided throughput capacity, i.e., $r(t) < d(t)$, then both network cost and QoE cost are incurred. In this scenario, the QoE cost is equal to $\delta(t)$, and thus, the game provider must bear the network cost of $\sigma(r(t))$ in addition to the QoE cost of $\delta(t)$.
- Alternatively, if the actual server's throughput demand can be met by the available network's throughput capacity, i.e., $r(t) > d(t)$, the game provider only needs to cover the network cost of $\sigma(r(t))$.

The aforementioned model can be represented as follows:

$$TC(\cdot) = \begin{cases} \sigma(r(t)) + \delta(t) & r(t) < d(t) \\ \sigma(r(t)) & r(t) > d(t) \end{cases} \tag{4.3}$$

We may transfer the aforementioned model to a version adapted for network slicing. Let $s(t)$ represent the slice number $S_i, i \in \{1, \ldots, N\}$, at time slot t, having a throughput capacity of a. The cost of slice S_i at time t is computed based on its throughput a and denoted as $\sigma(s(t))$.

The overall cost, considering the allocated slice at time t and the server's throughput demand, is determined by the following expression:

- In the event that the throughput demand of the actual server, denoted as $d(t)$, exceeds the provided throughput capacity of slice $s(t)$, that is, when $d(t) > s(t)$, a slice underestimate occurs, resulting in a loss of QoE. The QoE cost in this scenario would be represented by $\delta(t)$, which in addition to the network cost of $\sigma(s(t))$ represents the total cost.

- On the other hand, if the actual throughput demand falls within the range of provided throughput by the assigned slice, or if the demand can be met by a network slice with lower throughput capacity, then we have a scenario of ISOP (or slice over-provisioning), which means that the throughput provided by the network slice is greater than the demand. Despite the fact that the network provider is responsible for paying for the slice, a portion of the cost is in the form of a surcharge calculated as $(\sigma(s(t)) - \sigma(d(t)))$.

The aforementioned conditions are summarized in tabled 4.2 and can be observed in Figure 4.2. The cost model stated above may be expressed in the following manner:

$$TC(\cdot) = \begin{cases} \sigma(s(t)) + \delta(t) & d(t) > s(t) \\ \sigma(s(t)) & d(t) < s(t) \end{cases} \tag{4.4}$$

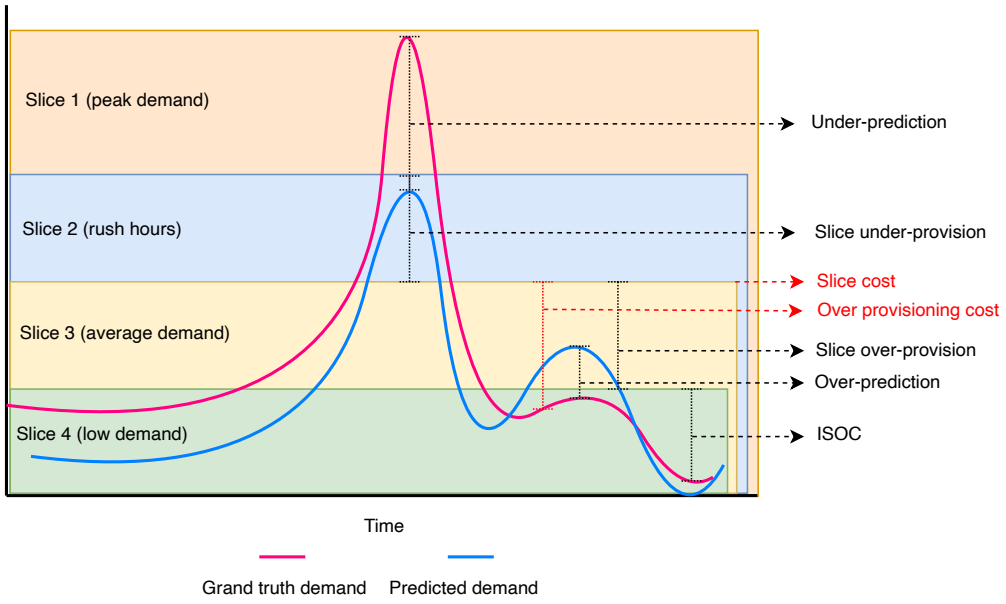


Figure 4.2: Throughput and slice prediction scenarios, based on four optimized network slices

Table 4.2: Cost table summary

Cost Name	Description
Network cost, $\sigma(.)$	The monetary cost of reserving a network with the available throughput of a
QoE cost, $\delta(.)$	An estimate of the cost in case of QoE loss
Slice cost	The cost of a slice with a throughput capacity of $a, \sigma(a)$
ISOC cost	The surcharge cost of using a slice in time t where the throughput demand is less than available capacity, $(\sigma(s(t)) - \sigma(d(t)))$
Over provisioning cost	The difference in network cost between available throughput capacity and throughput demand
Total cost, TC(.)	The compound cost of network and QoE cost, $\sigma(.) + \delta(.)$

4.3 Deep Neural Network based Predictor

4.3.1 DNN selection

The prediction of network conditions represents a time-series problem, necessitating predictions based on historical time-stamped data. Within the field of DNNs, LSTM models [19] have demonstrated efficacy in time-series prediction tasks with long-term dependencies [33, 36, 48]. Nonetheless, recent empirical evaluations of the GRU network [10] indicate that GRU can surpass LSTM units on some datasets by employing a fixed number of parameters for all models, in terms of convergence in CPU time, parameter updates, and generalization [11]. Consequently, we have opted to base our link condition prediction module on a GRU network. The following section elaborates on the GRU network.

4.3.2 GRU

The GRU model has been specifically designed to address machine learning problems that involve memory, and was first proposed by Cho et al. in their research [10]. This model employs two vectors to determine whether to update or reset its gate, thereby enabling it to avoid the problem of vanishing gradients. A detailed schematic of one GRU is shown in Figure 4.3.

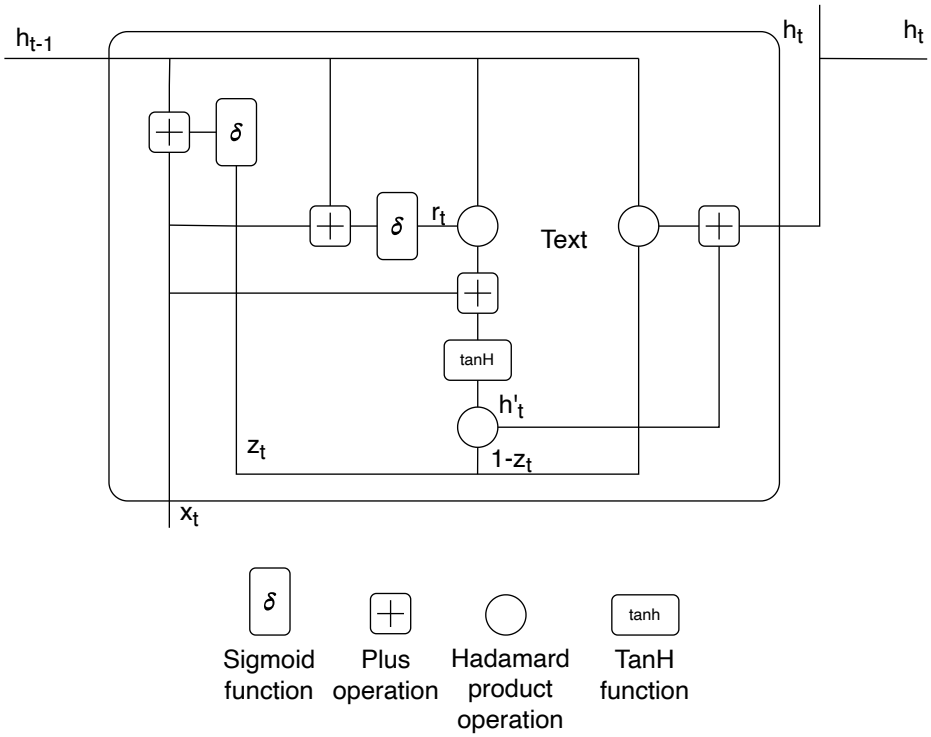


Figure 4.3: Illustration of one gated recurrent unit

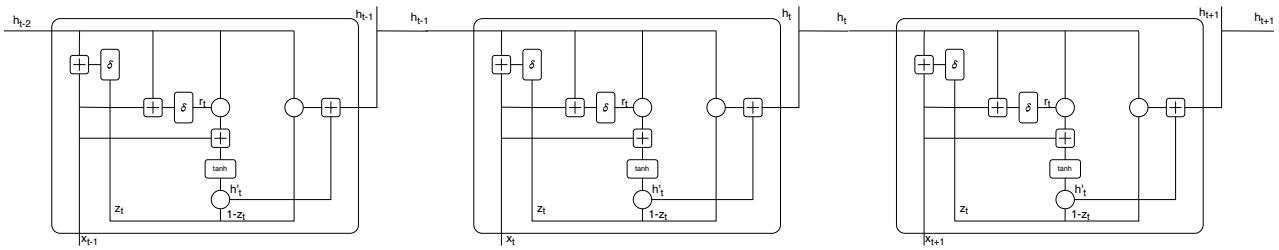


Figure 4.4: Illustration of GRU in RNN
Illustration of a repeating module in an recurrent neural network using gated recurrent units

The reset and update gates, denoted as r_t and z_t respectively, determine the information that needs to be passed to the output. As depicted in Figure 4.4, the units in this design are recurrently connected. Both gates utilize the sigmoid activation function σ to determine which data should be kept and which data should be ignored. Equation 4.5 describes the

mathematical process for computing the value of the update gate, where x_t, $W^{(z)}$, and $U^{(z)}$ refer to the input, input weights, and recurrent weights respectively.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{4.5}$$

The output of the sigmoid function is a value between 0 and 1. A value of 0 indicates that all information from the previous hidden state, denoted by h_{t-1}, should be disregarded, while a value of 1 signifies that all information should be considered. The reset gate value is calculated using Equation 4.6, which has its own set of weights, similar to the update gate.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{4.6}$$

Upon receiving the reset gate value, Equation 4.7 generates the candidate hidden state, which utilizes the reset gate value to determine the amount of data from the previous hidden state (h_{t-1}) that should be considered.

$$h'_t = tanh(Wx_t + r_t \odot Uh_{t-1}) \tag{4.7}$$

Finally, using the candidate hidden state and update gate, Equation 4.8 computes the value of the current hidden state h_t, which determines the amount of data that should be propagated through the network. It is worth noting that Equation 4.8 represents one of the key advantages of GRU over LSTM, as it enables GRU to simultaneously control both historical data (h_{t-1})) and new data (h'_t) using only the update gate.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{4.8}$$

When the update gate value z_t is close to zero, the first term of Equation 4.8 is also close to zero, which means that the new hidden state does not consider much information from the previous hidden state, and its value is largely determined by the new data from the candidate hidden state. Conversely, when the update gate value approaches one, the second term disappears, implying that the new hidden state value is entirely determined by the previous hidden state value (h_{t-1}).

4.3.3 DNN structure

In this research, we have chosen a GRU-based neural network architecture comprising one input layer, three hidden layers, and three fully connected layers. The first fully connected

layer employs the Rectified Linear Unit (ReLU) activation function, while the second layer utilizes the Hyperbolic Tangent (TanH) activation function. We have set the learning rate to 5×10^{-4} and the batch size to 192 for training the data. Furthermore, we have employed *Adam*, a Stochastic Gradient Descent (SGD) optimization technique, as the optimizer for this model. For further details regarding the DNN layers and hyperparameters, please refer to table 4.3. The proposed DNN network architecture is outlined in section 4.5.

Table 4.3: Deep neural network structure and its configuration

ML model layers	1- GRU (150 units) 2- GRU (50 units) 3- GRU (25 units) 4- Dense (10 units) 5- Dense (5 units) 6- Dense (1 unit)
Activation function	Layer 4,5: ReLU Layer 6: TanH
Learning rate	5×10^{-4}
Optimizer	Adam
Batch size	192

4.4 Loss function

The loss function is an essential component of any machine learning model, as it determines the effectiveness of the model in predicting the dataset. In conventional deep learning models, prediction accuracy is evaluated using Mean Square Error (MSE) and MAE as shown in equations 4.9 and 4.10, respectively. These metrics measure the distance between the actual and predicted values, wherein over-predicting and under-predicting carry equal weightage towards the final loss. However, in the context of online gaming, it is crucial to avoide under-predicting as it directly impacts the user's QoE and subsequently, the game provider's reputation. Unlike network operators who may prioritize link usage optimization over peak-time QoE, game providers aim to maintain a constant high QoE at the expense of occasional over-predicting of network resources. Hence, one of the innovative aspects of our proposed approach lies in the carefully chosen loss function.

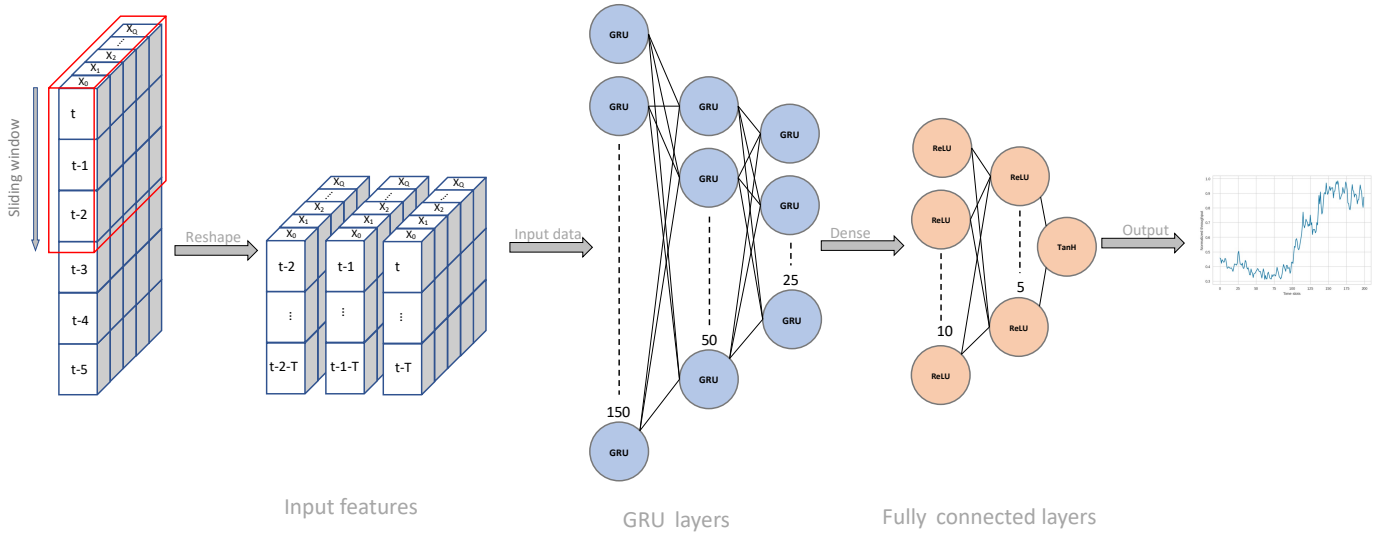


Figure 4.5: Structure of the GRU-based predictor

$$MSE = \frac{1}{N}\sum_{1}^{N}(y_i - \hat{y})^2 \tag{4.9}$$

$$MAE = \frac{1}{N}\sum_{1}^{N}|(y_i - \hat{y})| \tag{4.10}$$

The conceptual foundation of the proposed loss function was originally derived from DeepCog's loss function [5], which was developed to predict network throughput demand at the slice level from a network operator's perspective. In our proposed DNN, we have modified and employed the DeepCog loss function to predict server throughput demand. The actual value of the server's throughput demand at time t is represented by the vector $d(t)$, while the predicted demand is denoted by $p(t)$. The prediction error is defined as $\tilde{x}= p(t) - d(t)$. As depicted in equation 4.11, the loss function $l(\tilde{x})$ measures the loss of

resource under-predicting and incorporates a constant loss of β, which represents the QoE loss experienced by the game provider. The formulation of this equation is illustrated in Figure 4.6.

$$l(\tilde{x}) = \begin{cases} \beta & if\ \tilde{x} \leq 0 \\ \gamma \cdot \tilde{x} & if\ \tilde{x} > 0 \end{cases} \tag{4.11}$$

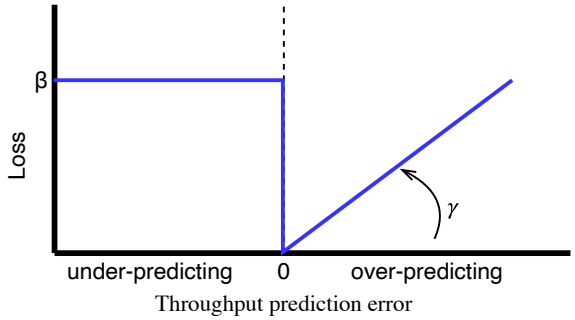


Figure 4.6: Loss model $l(\cdot)$, ideal model

The loss associated with over-predicting resources is expressed as a linear function in terms of a monetary loss denoted by γ for each network capacity unit, multiplied by the prediction error. As illustrated in equation 4.11, the loss function serves the purpose of achieving a balance between the loss of QoE and the loss of over-predicting. The ratio of γ to β determines this balance, and can be expressed as $\alpha = \beta/\gamma$ to simplify the expression. A higher value of α indicates a greater QoE loss, and defines the maximum loss of over-predicting for one capacity unit. Since the Adam optimizer is not capable of handling constant or step functions, we have introduced a small value ϵ to enable SGD to function as intended. Equation 4.12 and Figure 4.7 depict the updated and implemented loss function.

$$l(\tilde{x}) = \begin{cases} \alpha - \epsilon \cdot x & if\ \tilde{x} \leq 0 \\ \alpha - \frac{1}{\epsilon}\tilde{x} & if\ 0 < \tilde{x} \leq \epsilon\alpha \\ \tilde{x} - \alpha\epsilon & if\ \tilde{x} > \epsilon\alpha \end{cases} \tag{4.12}$$

To assess the impact of the loss function on the DNN prediction, we utilized the proposed DNN architecture presented in the previous section to forecast server throughput demand over 1000 time slots using MSE, MAE, and the loss function that was selected. The data set description is available in Section 5.2 A portion of the server demand prediction, using MSE as the loss function, is illustrated in Figure 4.8b. Although the predicted

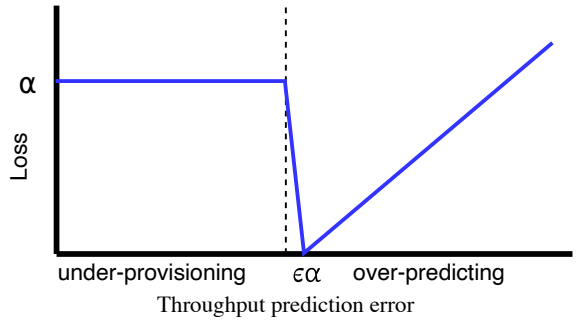
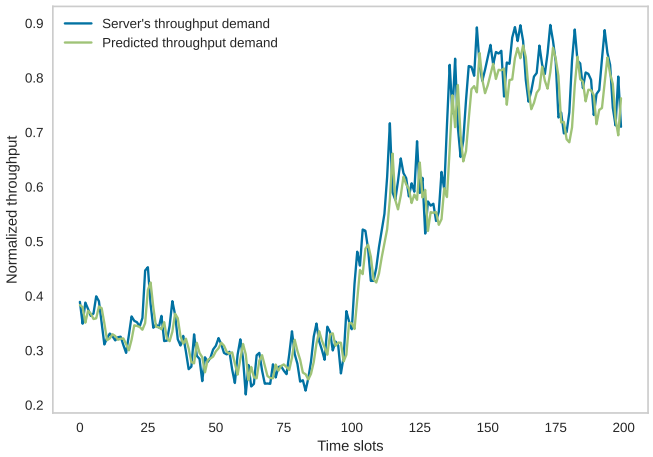
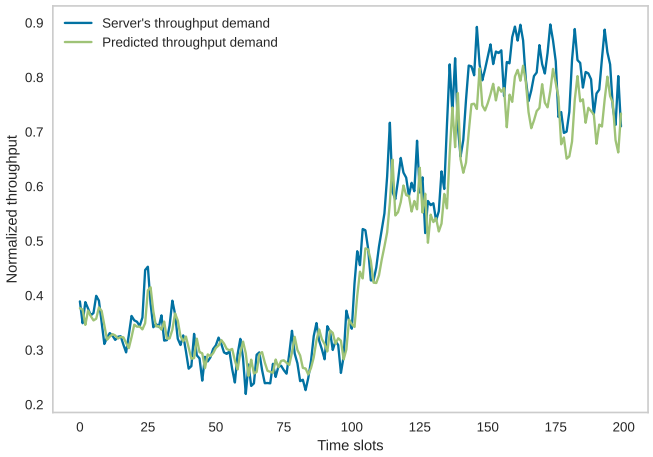


Figure 4.7: Loss model $l(\cdot)$, implemented model

demand is in close proximity to the ground truth demand, an under-predicting trend can be observed in the prediction, which is suboptimal for QoE. Figure 4.8c portrays demand prediction using the chosen loss function, which successfully assisted the DNN in avoiding under-predicting. Moreover, we computed the number of occurrences of under-predicting for the three loss functions. Figure 4.9 compares the total percentage of server demands that were under-provisioned. As indicated, the selected loss function resulted in a reduction of over 50% in the number of under-predicting occurrences in this evaluation.



(a) Predicted throughput demand using MSE as a loss function



(b) Predicted throughput demand using MAE as a loss function

(c) Predicted throughput demand using the selected loss function

Figure 4.8: Predicted server throughput demand using the proposed DNN for 1000 time slots. (a): using MSE as loss function. (b) using MSE as loss function. (c): using the selected loss function, server i=3, α=0.2.

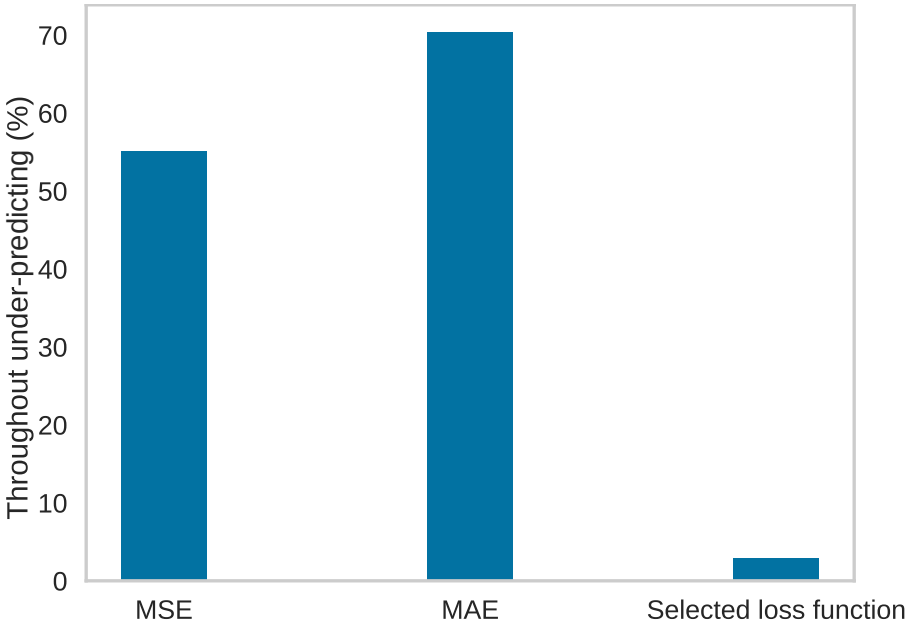


Figure 4.9: Comparing the impact of loss function on the number of throughput under-predicting predictions in 1000 time slots using the proposed DNN. Our proposed loss function reduced the number of under-predicting cases that directly impact the QoE, by more than 50%, server i=3,α=0.2.

4.5 Slice Optimizer

The Slice Optimizer module is responsible for providing slice options based on requested numbers. A slice's cost is influenced by two primary factors: its SST and its throughput capacity. The slice type is a constant factor as game providers are required to offer users a minimum level of QoS. In contrast, the slice's throughput capacity is a dynamic value determined based on tenant requests and approved by the network provider. Managing multiple slices with varying throughput capacities can be insufficient for network providers due to resource orchestration complexity. As a result, for this study, we assume that a slice tenant and a network provider agree to select a set of slices, and the slice tenant will not have any issues switching between them. Therefore, the Slice Optimizer module should optimize slices based on the requested number and keep ISOC costs to a minimum for each time step. Several factors must be considered when selecting a solution for the Slice Optimizer module:

Large dataset: Depending on the sample rate, the server's throughput demand history

may include thousands of sample data points. Therefore, the solution must handle large datasets.

Processing time: Game providers continually create, upscale, and downscale their servers and services using cloud infrastructures. Therefore, the chosen solution should optimize network slice options within a short period by utilizing available data.

Granted convergence: The Slice Optimizer should suggest slices based on the request. As a result, the algorithm should always converge and find the optimum slice options.

Low computation cost: Game servers are deployed in cloud infrastructures, and utilizing resources will result in monetary costs. Therefore, the solution should use relatively fewer resources.

Prioritization: During rush hours, peak hours, and special events, the server throughput demands are more significant and have higher priority. Hence, the module must prioritize these instances during network slice optimization to ensure a positive user experience.

Considering all these factors, we have decided to employ a ML solution to optimize network slices. As we do not know how we should categorize and group these throughput demands, our ML options are limited to unsupervised learning algorithms. Therefore, we have chosen to use K-means clustering [29] as our base ML model, which satisfies all the mentioned conditions.

4.5.1 K-means clustering

K-means clustering is an unsupervised learning algorithm that employs a centroid-based approach to determine the distance between data points within an unlabeled dataset. The algorithm partitions data points into a predetermined number of clusters. Given a dataset $X = \{x_i \dots x_n\}$, the K-means algorithm clusters n data points into k clusters by following these steps:

1. **Specify the number of clusters:**
   Since K-means is a centroid-based algorithm, before the algorithm is initiated, the number of clusters (k) must be defined. For instance, in the following example, we choose k=3 clusters.

2. **Initialize the centroids by selecting random k data points:**
The algorithm selects k random initial clusters' centroids, represented by μ_j, $j \in k$. Figure 4.10 illustrates a set of data points, and figure 4.11 shows three initial clusters' centroids, with k=3.

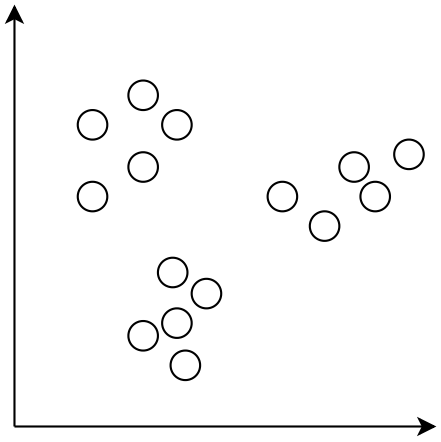


Figure 4.10: A selection of sample data

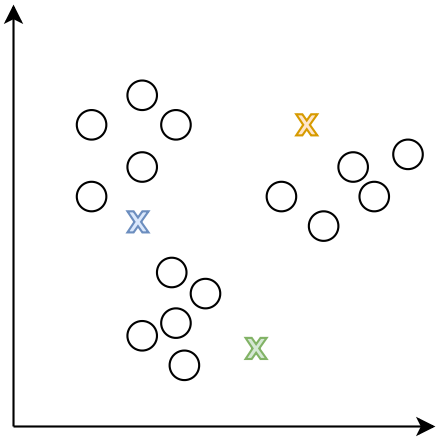


Figure 4.11: Initializing clusters' centroid with rendom possitions

3. **Assign each data point to the nearest centroid:**
Equation 4.13 is employed to measure the distance between each data point and the centroid. The data point is subsequently assigned to the centroid with the smallest distance. After the distances have been calculated, data points are assigned a label denoted by c^i, which indicates the nearest centroid. The cluster assigned to each data point based on distance is illustrated in Figure 4.12.

$$c^i := \underset{j}{arg\,min} \left\| x^{(i)} - \mu_j \right\|^2 \tag{4.13}$$

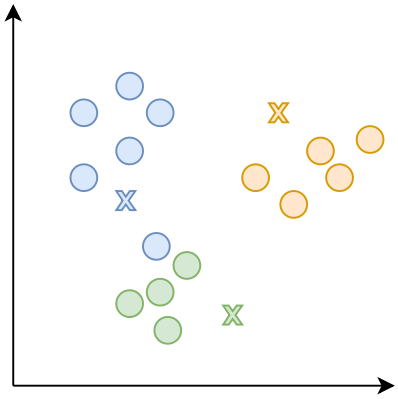


Figure 4.12: Assinging clusters to datapoints based on their distance from the centroids.

4. **Reinitialize the centroids:**
   After the algorithm assigns a label to each data point, each cluster includes data points with the same label. In the following step, the algorithm computes the central gravity of the clusters using equation 4.14 and relocates the centroids to the central gravity point. Figure 4.13 illustrates the central gravity of the two clusters, and 4.14 displays the updated position of the clusters' centroids.

$$\mu_j = \frac{\sum_{i=1}^{n} 1\left\{c^{(i)} = j\right\} x^{(i)}}{\sum_{i=1}^{n} 1\left\{c^{(i)} = j\right\}} \tag{4.14}$$

5. **Repeating steps 3 and 4:**
   The algorithm continues to repeat steps three and four until it converges, which means that the centroids' positions remain unchanged, or it reaches a specific stopping condition. Figure 4.15 depicts the final state of the K-means algorithm and the final clusters.

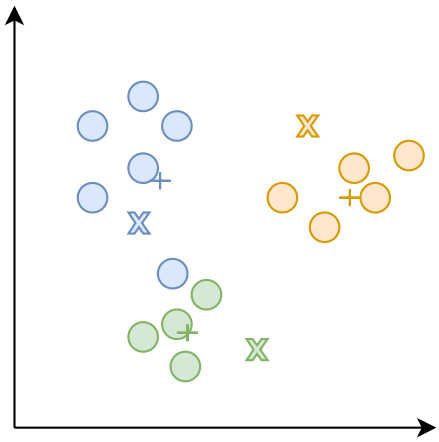


Figure 4.13: Claculated clusters' central gravity, indicating by +.

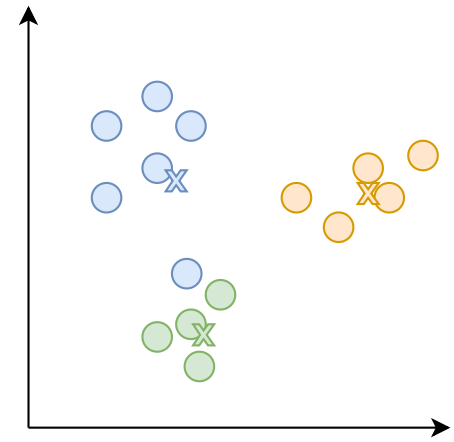


Figure 4.14: Updated possition of clusters' centroids.

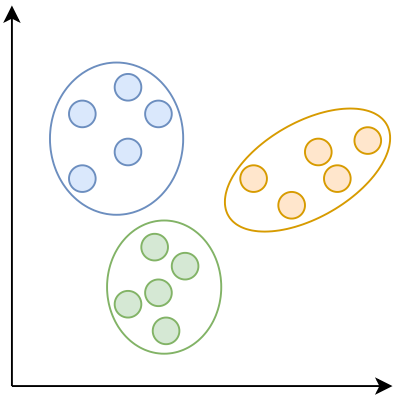


Figure 4.15: Final labled data after formation of clusters

Weighted K-means

The prioritization of data points related to certain times is a crucial requirement for the selected solution, as highlighted in section 4.5. This requirement encompasses various scenarios, such as special events, rush hours, and peak usage. In the present study, the novel variant of K-means, known as weighted K-means [24], was utilized to address this requirement. Weighted K-means is an adaptation of the conventional K-means algorithm, which involves assigning an importance value or weight to each data point. The incorporation of

weights in the clustering process provides more influence to the significant data points in determining the cluster centers. This approach is particularly beneficial in situations where the data has varying levels of relevance or reliability. Furthermore, it can be advantageous when specific data points should have a more profound impact on the final clusters than others. The algorithm's implementation involves modifying the distance calculation between data points and cluster centers to consider the weight of each point. Equation 4.15 presents the updated formula for calculating central gravity in step four, which reflects the incorporation of weights in the algorithm.

$$\mu_j = \frac{\sum_{i=1}^{n} 1\left\{c^{(i)} = j\right\} w(x^{(i)}) x^{(i)}}{\sum_{i=1}^{n} 1\left\{c^{(i)} = j\right\} w(x^{(i)})} \tag{4.15}$$

4.5.2 K-means weight options

In order to apply weighted K-means in this study, it is important to introduce weight options and subsequently choose the one that yields the lowest total cost (4.2). The following provides a comprehensive explanation of the three weight models that were selected for assessment purposes.

Throughput Demand Distribution (TDD): The first weight model under consideration is the TDD. This approach involves describing the dataset using frequency distribution, which entails dividing the dataset into smaller, equal units known as bins, and counting the number of data points in each bin. A histogram is then used to visually represent this distribution. Figure 4.16 illustrates the histogram of the server's throughput demand using 50 bins, based on the dataset described in section 5.2. In order to assign weights to the data points, the bin values were normalized and each data point within a bin was given the same weight value.

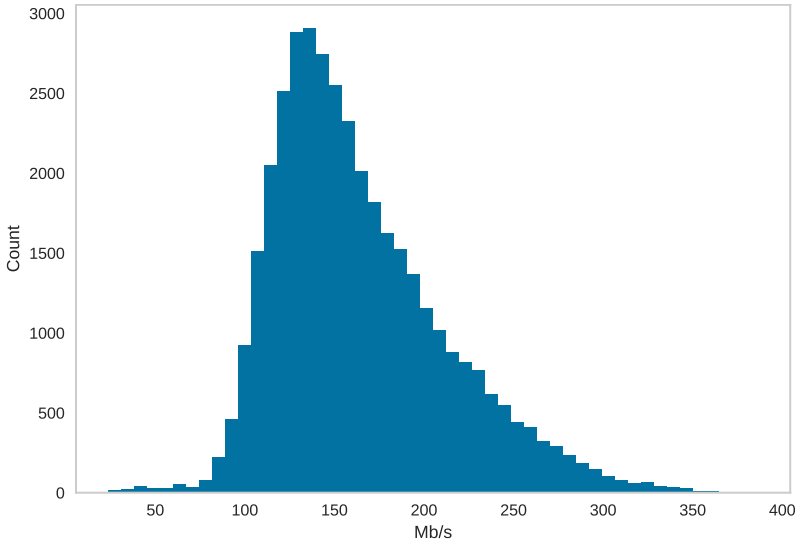


Figure 4.16: TDD of selected server

Cumulative Distribution Function (CDF): The second weight model considered is the CDF. The CDF defines the probability of a variable's accumulated values. Specifically, the CDF $F(x)$ of a variable y is defined as equation 4.16, which calculates the probability that a variable is equal to or less than x. Equation 4.17 expresses the CDF for a continuous distribution. Figure 4.17 presents the normalized CDF plot of the server's throughput demand. In this weight model, the normalized CDF probability of each data point is considered as its weight.

$$F(x) = Pr[X \leq x] \tag{4.16}$$

$$F(x) = \int_{\infty}^{x} f(\mu) d\mu \tag{4.17}$$

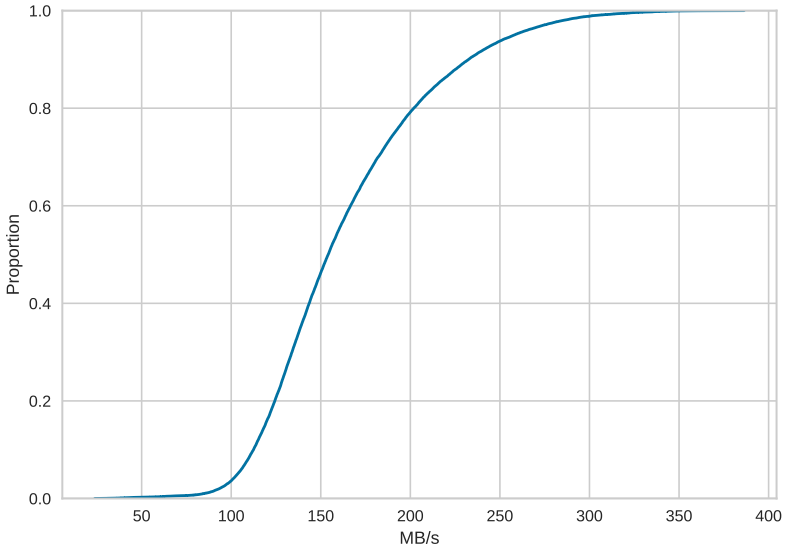


Figure 4.17: CDF of server throughput demand

Throughput Costlo (TC): The throughput capacity of a slice has a direct impact on its final cost. Since we have the server's throughput demands at each time slot, using the network's cost model $\sigma(\cdot)$, proposed in Section 4.2, we can calculate the cost of a network slice with same throughput capacity as the server's demand for each time slot. The predicted cost was assigned as a weight.

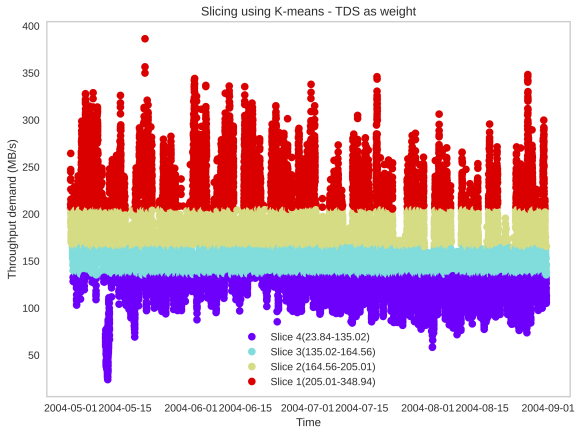
4.5.3 Weight performance

In the present study, an evaluation was carried out to assess the performance of the weighted K-means model by utilizing the training data as described in Section 5.2. The optimal weight was selected through a rigorous evaluation process, following which the K-means model was employed, incorporating weights, to label the time slots with clusters based on their values and corresponding weights. Each cluster, also referred to as a network slice, was characterized by the time slot with the highest throughput demand, which defined the slice's throughput capacity (a). The cost of each slice was calculated using the network's cost model $\sigma(\cdot)$. Subsequently, the compound slice cost of the time slots was determined, defining the total slice cost.

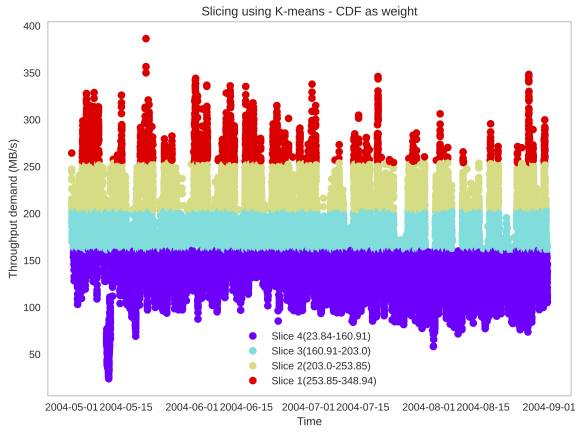
In some instances, the assignment of network slices to time slots resulted in ISOC. Therefore, to aid in selecting the optimal weight, the total ISOC cost was calculated to indicate the portion of the total slice cost that was surcharged. Figure 4.18 presents a

comparison of the K-means and weighted K-means slice optimization under three weight options for four slices. Moreover, Figure 4.19 provides a cost comparison of K-means models, illustrating that using "throughput cost" as a weight can reduce the total slice cost when compared to TDD and CDF weights. Furthermore, a comparison of traditional K-means (without weights) and weighted K-means shows that the latter performs better.

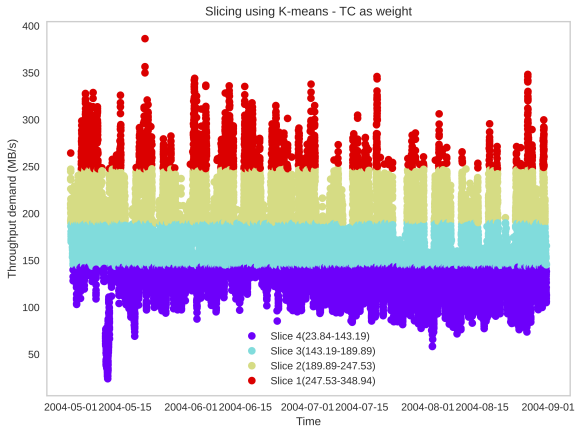
(a) Traditional K-means (no weight)



(b) Weighted K-means (using TDD as weight)



(c) Weighted K-means (using CDF as weight)

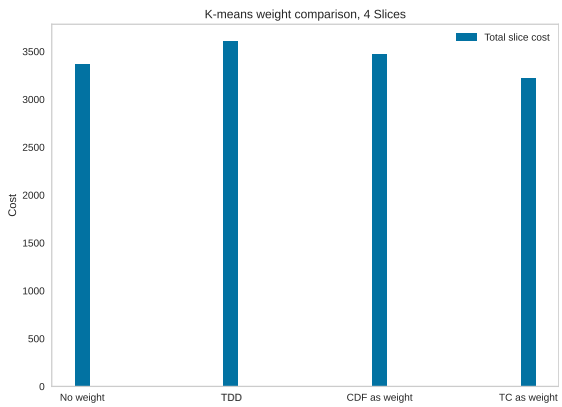


(d) Weighted K-means (using TC as weight)

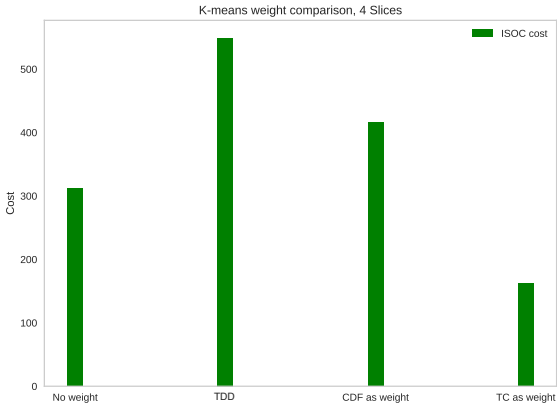
Figure 4.18: Slicing using K-means clustering with respect to four weights

Since the results of this study are dependent on the behavior of server and its users, the test was conducted on the data of ten servers available in the dataset. Figure 4.20 presents the average result of four slice optimizations for the ten available servers, wherein

using "throughput cost" as a weight demonstrates superior performance and relatively lower total slice cost.

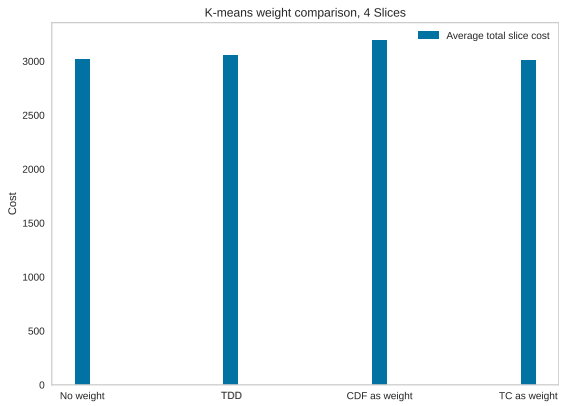


(a) Total slice cost

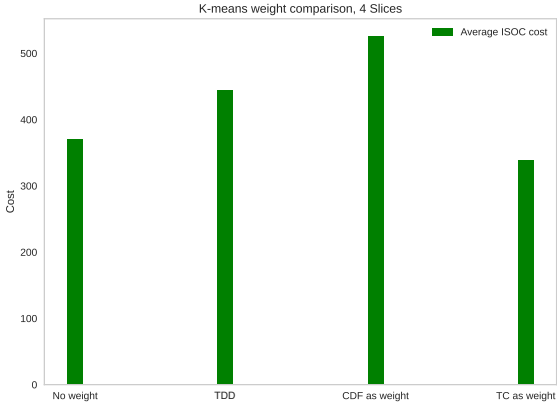


(b) ISOC cost

Figure 4.19: Cost comparison, under slicing by K-means algorithm using different weights for server number three. (a) total slice cost. (b) ISOC cost.



(a) Average total slice cost



(b) Average ISOC cost

Figure 4.20: *Average* of server cost comparison, under slicing by K-means algorithm using different weights for ten servers. (a) total slice cost. (b) ISOC cost.

4.6 Slice selector

The proposed architecture includes a key component known as the slice selector module, which plays a crucial role in the overall functioning of the system. This module is responsible for selecting the optimal slice for the next time slot, in order to ensure that the Quality of Experience (QoE) is not compromised.

The slice selector module receives input in the form of predicted network conditions $p(t_f)$, denoted as p, and the set of available slices $\{S_1, ..., S_N\}$. Using this information, the module performs a detailed analysis to determine the most suitable slice for the server. The goal is to minimize any potential loss in QoE, while also keeping the operating costs for the game provider to a minimum. The process of calculating the operation cost is discussed in detail in Section 4.2.

In order to make an informed decision, the slice selector module takes into account various attributes of the slices and the server. In this particular study, the primary attributes considered include the throughput capacity of the slices and the throughput demand of the server. By considering these key attributes, the module is able to make a decision that balances the needs of the game provider and users.

The slice selector module functions as a lookup table, offering a rapid and efficient mechanism for selecting the optimal slice that can fulfill the anticipated throughput demand while minimizing costs. For instance, if the slices optimizer recommends four slices with respective maximum capacities of 50 MB/s, 75 MB/s, 120 MB/s, and 200 MB/s and the link condition prediction module forecasts a throughput demand of 95 MB/s for the subsequent time interval, the "slice selector" module will choose the third slice with a throughput capacity of 120 MB/s to satisfy the demand and prevent unnecessary network cost.

Chapter 5

Performance Evaluation

5.1 Introduction

In this study, we aim to investigate the potential of network slicing as a means of improving cost efficiency while maintaining a satisfactory Quality of Experience (QoE) for online gaming. To this end, we propose a new cost model that accounts for both cost and QoE, as opposed to traditional performance evaluations that are primarily focused on legacy metrics. Furthermore, we intend to evaluate the efficacy of our proposed solution using real-world data and newly-extracted features to improve prediction accuracy.

The proposed solution comprises two main components: a network condition prediction module and a slice optimizer. Each of these components has specific attributes that must be optimized for each individual server. Therefore, we dedicate two sections of this chapter to the optimization of these attributes.

In the final section of this chapter, we compare the performance of our proposed solution with traditional and existing solutions in the industry. Through this comparison, we aim to highlight the advantages and disadvantages of our proposed solution and demonstrate its potential to improve the cost-efficiency and QoE of online gaming. Additionally, we will also identify the areas where further research is needed to fully realize the potential of network slicing in online gaming.

5.2 Data preparation

Dataset

In order to assess the efficacy of the proposed architecture, we utilized the Abilene dataset [55], which furnishes genuine traffic measurements of the Backbone network situated in the United States. The aforementioned dataset comprises 12 servers and 15 links. The data was sampled at five-minute intervals, starting from 00:00 hours on March 1, 2004, until 23:55 hours on September 10, 2004. Figure 5.1 illustrates the topology of the Abilene network.

Figure 5.1: Abilen Network Topology [44]

Data Cleaning

The selection of the data used to train, optimize, and test the various components of the proposed architecture holds paramount importance as it has a great impact on the final performance. As such, for this particular undertaking, a comprehensive examination of the dataset was carried out in order to identify and isolate any anomalous records. Of the twelve servers that were available in the dataset, the data originating from servers three and eight was excluded from the analysis as it contained excessive noise and breaks. Moreover, given that the data from all the remaining servers was utilized, it was important to identify a timeframe in which none of the nodes' data suffered any interruptions. Consequently, the recorded data between May 1, 2004, at 00:00 hours and September 4, 2004, at 23:55 hours was employed for the analysis.

Feature Engineering

We propose the integration of additional features into our network condition predictor module to increase its predictive accuracy. Such features would be supplementary to the existing dataset, which already consist of servers' throughput demand and time. Typically, time-series prediction involves indexing data by date-time, and the identification of cyclical patterns within the data can significantly enhance the prediction's efficacy. By leveraging the date-time information, we can extract various categorical features such as days of the week, days of the month, and holidays. Consequently, we employed the one-hot encoder [28] method to incorporate these novel features into our dataset.

As mentioned in the preceding section, the throughput demand of the server is susceptible to change at certain times of the day, such as rush hours. One method of discerning a correlation between throughput demand in time-series data is through autocorrelation. "Autocorrelation is a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals" [43]. As depicted in Figure 5.2, the autocorrelation result of a server is shown, with lags ranging from 1 to 350. The correlation at each lag is represented on the Y-axis. Notably, the initial 50 lags exhibit a considerable degree of correlation, which is reasonable given that the demands in consecutive time slots are similar. In this figure, the shaded grey region denotes the confidence interval, implying that the correlation for the bars outside this region is statistically significant. Upon further analysis of the figure, we can observe that the correlation increases again, reaching its maximum at lag 288. It is worth mentioning that the dataset's sampling rate interval is 5 minutes, and the following 288 timeslots correspond to precisely the same time the next day. This observation suggests that the server's throughput demand follows a daily cycle.

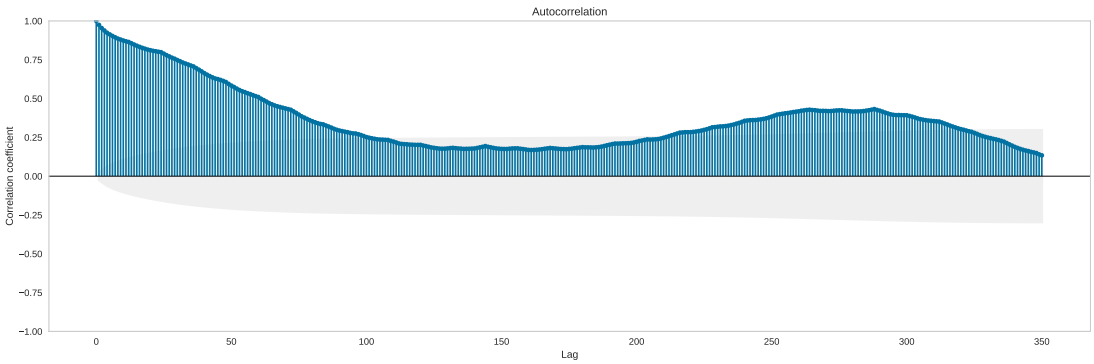


Figure 5.2: Autocorrelation graph of server's throughput demand

Recent studies [23,37,56] have demonstrated that time can serve as a feature to enhance the performance of a model. Our investigation has further revealed that the time of day plays a significant influence on the server's throughput demand. Thus, incorporating the time of day as a new feature can effectively improve the model's performance. However, it is imperative to note that we cannot simply add the time as it is. Nevertheless, it is crucial to acknowledge that the direct addition of time intervals is not feasible in this context. For illustrative purposes, let us suppose that we adopt a 24-hour format for the daytime and assign a unique numerical value to each minute, such as designating 0 for 00:00 and 1439 for 23:59. Under this mapping, if we were to calculate the difference between 23:30 PM and 00:05 AM, it would yield 1405. However, in reality, there is only a 35-minute gap between the two time points. Thus, it becomes evident that a linear relationship does not provide useful insights in this particular scenario. As such, we must transform time into a new format to mitigate this issue. As proposed in [7], it is possible to convert time into a cosine value. As shown in Figure 5.3, this conversion maps time onto a continuous scale ranging from -1 to 1. However, an issue arises due to the periodicity of the cosine function, as it produces the same value for different times. This is illustrated in Figure 5.4a. To overcome this obstacle, we can introduce sine as the second function and include it as an additional feature in our dataset, Figure 5.4b . By employing both cosine and sine values as coordinates in a two-dimensional system, the model can differentiate between times that have identical cosine values, as shown in Figure 5.5.

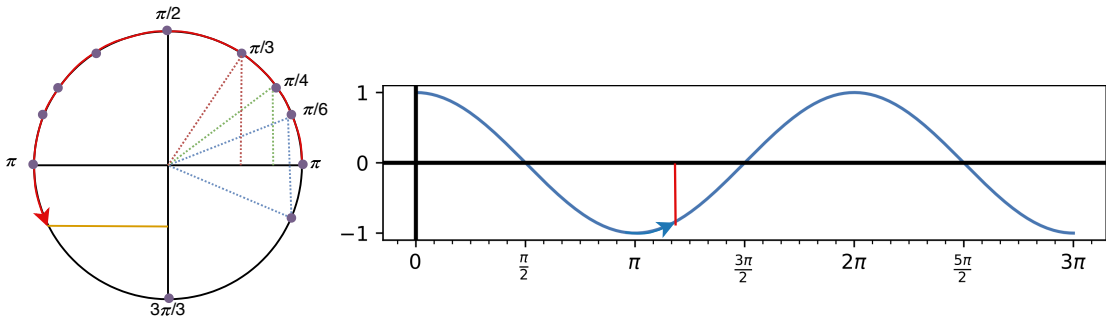
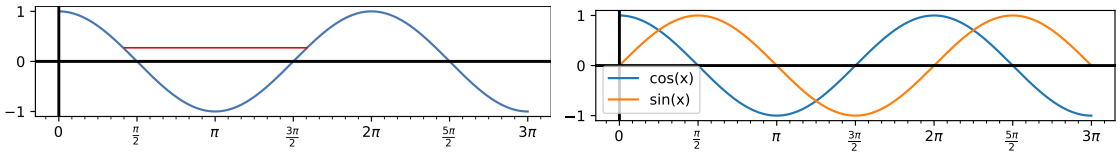


Figure 5.3: Cosine function [21]



(a) (b)

Figure 5.4: Mapping time to cosine and sine functions. (a) Mapping time to the cosine function and the problem of having an identical value for two separate times. (b) Adding sine as the second feature to have a unique combination of cosine and sine values to represent time.

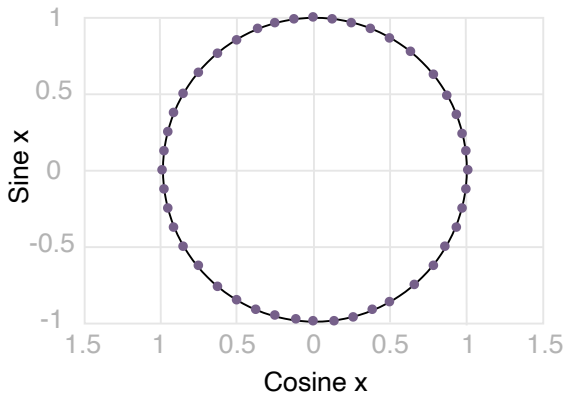


Figure 5.5: The cosine-sine coordinates system [7]

5.3 Experiment setup

In order to train, optimize, and assess the proposed architecture, we utilized the real-world dataset that was described in section 5.2. Out of the ten available servers, we chose server four as the primary server for evaluation. This decision was based on the fact that server four's throughput demand falls within the range of available real-world network costs that were introduced in section 4.2.

For training the models, we employed 30,000 time-stamped data points, which were recorded between May 1, 2004, at 00:00 and August 30, 2004, at 04:00. We then utilized the subsequent 3,000 records for the purpose of validation. To mitigate any potential biases,

Table 5.1: Deep neural network structure and configuration

Batch size	192
epochs	1000
Sliding window size	8
Forecast horizon	1
Training set	30,000
Validation set	3,000
Test set	1,000
Early stopping patience	20

we opted to incorporate a gap between the training and testing datasets. Accordingly, we employed 1,000 data points that were recorded between August 31, 2004, at 12:45 and September 4, 2004, at 00:05 as the test dataset.

The dataset was restructured in accordance with the scheme that was outlined and depicted in Section 3.2.The configuration of the network condition prediction module involved the utilization of a batch size of 192. In this experiment, a sliding window size of eight samples was employed, with a forecast horizon of five minutes. Specifically, the proposed solution employed the previous 40 minutes of demand history to predict the subsequent 5 minutes of throughput demand. For a more comprehensive understanding of the DNN setup, please refer to table 5.1.

5.4 Impact of the alpha

As expounded upon in this thesis, the principal aim of this study is to harness the potential of network slicing in the context of online gaming, specifically in relation to network and QoE costs. Hence, we undertake an assessment of the influence of each component, by means of analyzing its respective impact on the performance of the proposed architecture. As described in Section 4.4, the chosen loss function was specifically devised to fine-tune throughput demand prediction while taking into account QoE and network cost strategies. A game provider has the ability to adjust their QoE/Network cost strategy through modification of the loss function parameter, α. Such a parameter modification allows for the implementation of diverse strategies based on factors such as region, market, event, and game category, among others. For instance, given the considerable market capital of online gaming in North America [45], the game provider may opt to offer the highest feasible QoE, regardless of cost, in order to gain a competitive edge over other rivals and

attract a larger user base. Similarly, if the game provider decides to compete with low-price competitors and expand its market share in this segment, it may choose to adopt more cost-effective network strategies for providing gaming services in this particular tier. In accordance with the adapted strategy, the network provider can assign a lower value to α in order to minimize the network cost, or a relatively higher value to ensure an improved QoE while minimizing the associated cost. Figure 5.6 illustrates different strategies ranging from $\alpha = 0.1$, which offers a tactic for minimizing the network cost, resulting in a lower slice over-estimate cost, to $\alpha = 5$, which provides a method for minimizing QoE loss, and thus allocating more resources towards achieving this goal.

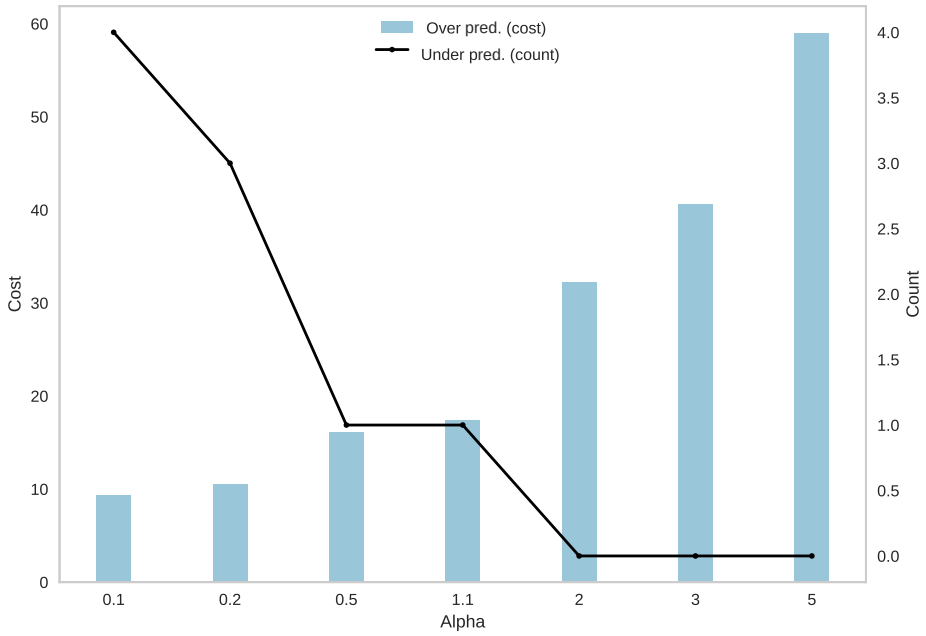


Figure 5.6: Trade-off between the cost of over-predicting and the number of under-predicting occurrences in the presence of α on the test dataset.

Based on the results of the experiment, it can be concluded that the selected loss function was effective in implementing the proposed strategies. Figure 5.6 illustrates that increasing α from 0.1 to 0.5 led to a significant decrease in the occurrences of under-predicting by more than 70%. It is worth noting that a further reduction in under-predicting can be achieved by increasing α. However, this would necessitate the allocation of additional resources, leading to higher network costs for the game provider. During the experiment, it was observed that an α value of 0.5 provided an acceptable balance between the oc-

currences of QoE loss and the cost of over-predicting. As a result, an α value of 0.5 was utilized for the remainder of the evaluation.

5.5 Impact of the slice number

As outlined Section 4.5, a network slice is characterized by two primary attributes: slice type and slice differentiator. The slice type specifies the QoS standards that a given slice must adhere to, while the slice differentiator, in this particular context, denotes the throughput capacity of the slice instances. In the present study, we are primarily concerned with the slice differentiator and will henceforth refer to it as the slice option. The availability of multiple slice options empowers the game provider to select the most suitable option based on their immediate requirements. This, in turn, enables the game provider to reduce their network expenditures without any consequential loss of QoE. However, augmenting the number of slices may potentially hinder this advantage. Figure 5.7 illustrates the impact of increasing the slice options on the total cost and over-provisioning cost, revealing a direct correlation between these two expenses. Another key takeaway from Figure 5.7 is the decreasing trend observed in both costs by increasing the slice options. This trend persists with the addition of nine slice options; however, the trend changes with the inclusion of the tenth slice option. To further investigate the reason for this phenomenon, we have included the server's throughput demand distribution (outlined in Section 5.6) and the optimized capacity values of the slice options in Figure 5.8. Upon examining Figure 5.8a, which features nine slice options, we observe that the throughput demands are not uniformly distributed, and some demands have a higher frequency. Based on the comparison of Figure 5.8a with Figure 5.8b, which features ten slice options, it is apparent that some of the highly requested throughput demands, such as the one indicated with red arrows, would be located close to the slice's lower band. As a result, the number of ISOCs increases, leading to higher over-provisioning costs.

As is apparent from this assessment, the optimal number of slice options varies for each server and hinges upon the characteristics of the server, its network usage, and the network cost. In order to choose the optimal slice options for the server, we employed a modified version of the elbow method based on the total cost. As depicted in Figure 5.7, a curve in the total cost graph appears at the 9th slice option, which is close to the potential number of slice options. Therefore, we chose nine slice options as the optimal number for server three and used this number for the remainder of the evaluation. Figure 5.9 shows the optimal slice options for the available servers in our dataset, thus supporting our hypothesis. Moreover, this figure demonstrates that even though we are not limited

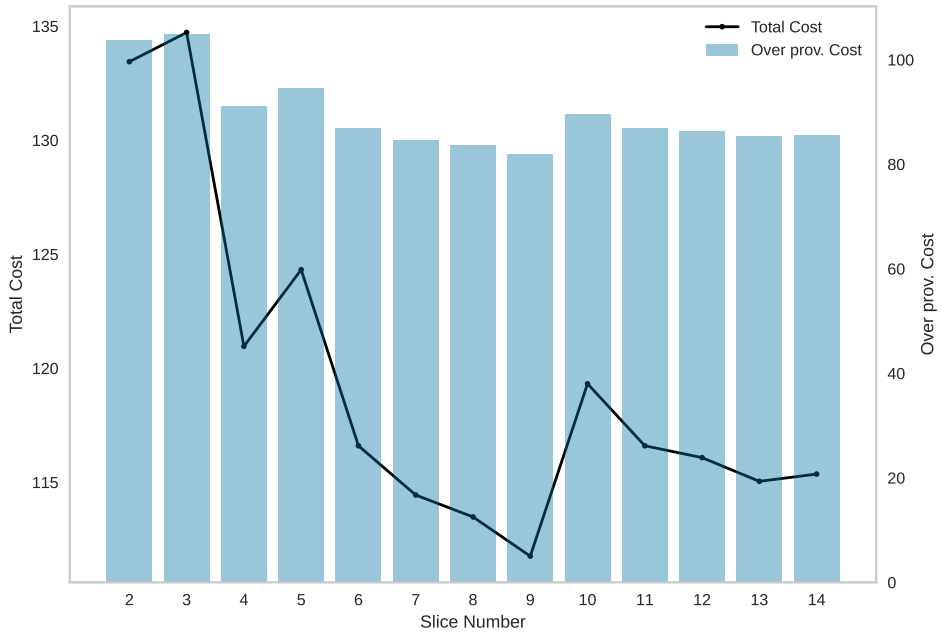
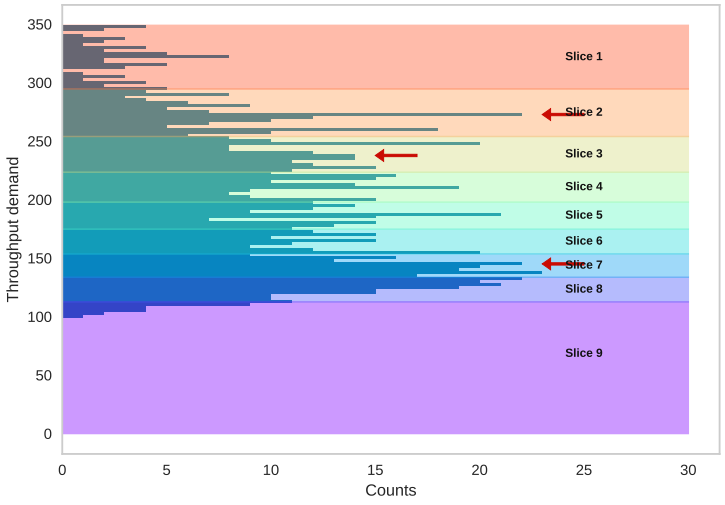


Figure 5.7: Impact of increasing slice number on total cost and over-provisioning cost on the test dataset.

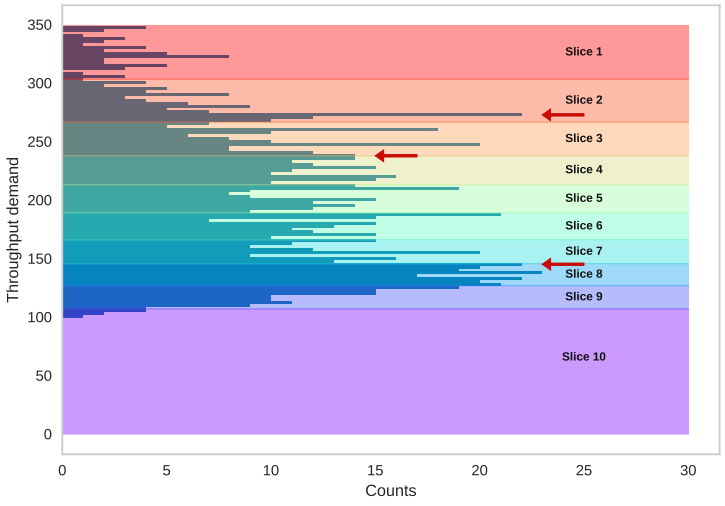
in the number of slice options, the range of slice options for the individual servers ranges between three and nine, further affirming our assumption that an increase in slice options does not necessarily translate to better total cost.

5.6 Orchestration results

Thus far, we have conducted an assessment of the individual modules comprising NETSPRO. In this section, we will analyze the performance of NETSPRO as a comprehensive solution. In order to evaluate the effectiveness of NETSPRO, we utilized the network condition prediction module with the trained data (5.2). The value of α for this section was established as 0.5, and the DDN model was implemented with the previous eight time slots to forecast the subsequent time slot (T=8, F=1). Figure 5.10 provides an illustration of the forecasted values for the initial 200 test time slots, verifying the efficacy of the proposed DNN and selected loss function in effectively reducing under-provisioning in the forecasting process.



(a)



(b)

Figure 5.8: Effect of increasing slice number on slice ranges (a): having nine slice options. (b): having 10 slice options. Adding new slices can put the switching threshold on critical demands resulting in extra over-provisioning costs.

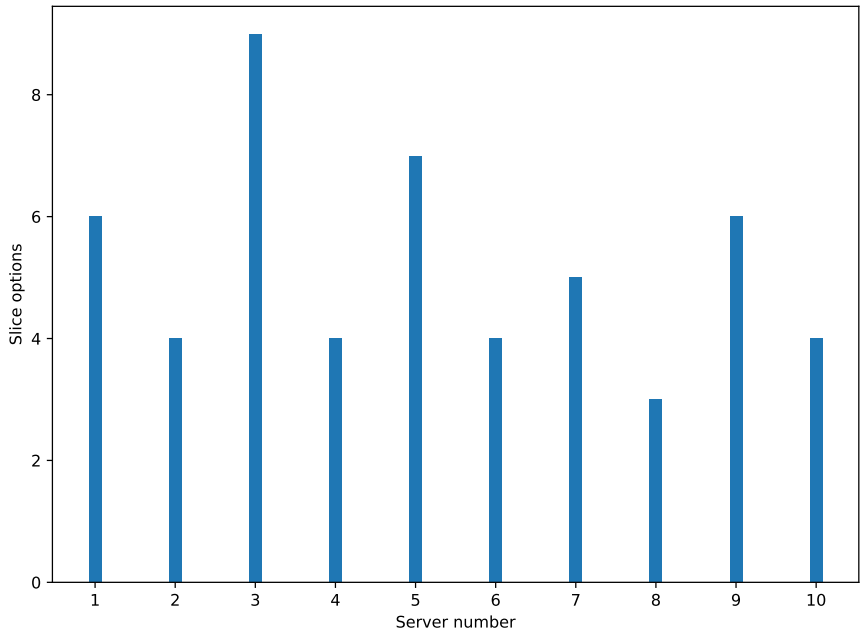


Figure 5.9: Selected slice number for the ten available servers based on the proposed approach

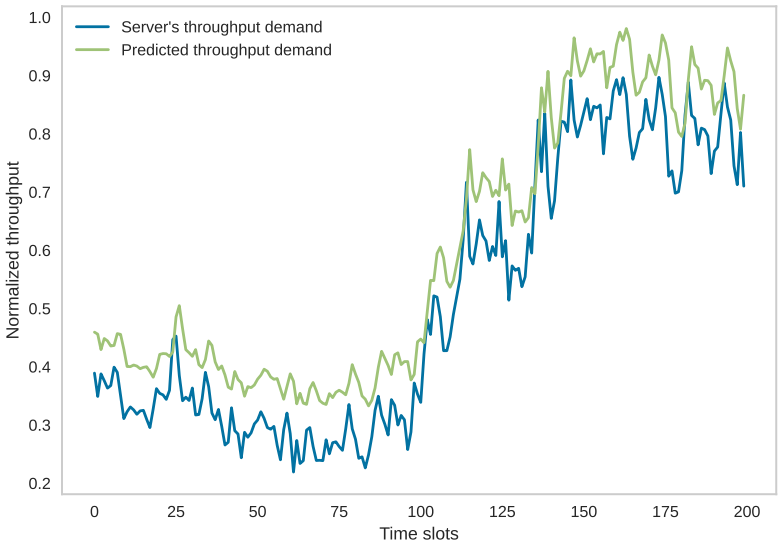


Figure 5.10: Predicted throughput using the proposed ML-based link condition prediction module for one sample link. α=0.5, a slide prediction window size of T=8, and the forecast horizon of F=1 for server three.

The same training data was utilized to optimize the slice options. As demonstrated by the experimental results presented in Section 5.5, the optimal number of slice options for the chosen server was determined to be nine. Accordingly, we configured the slice optimizer module to optimize nine network slices for this assessment. To optimize the slices, we employed the network cost for the K-means weight, based on the evaluation results presented in Section 4.5.3. Figure 5.11 displays the nine optimized slice options and the chosen slices for each time slot, based on the anticipated throughput demand of the server. The results confirm the successful optimization of nine slices, providing enough throughput capacity for all requested demands. Additionally, the slice optimizer module allocated more slice options to frequently requested throughput demands, specifically those between 100 MB/s and 250 MB/s.

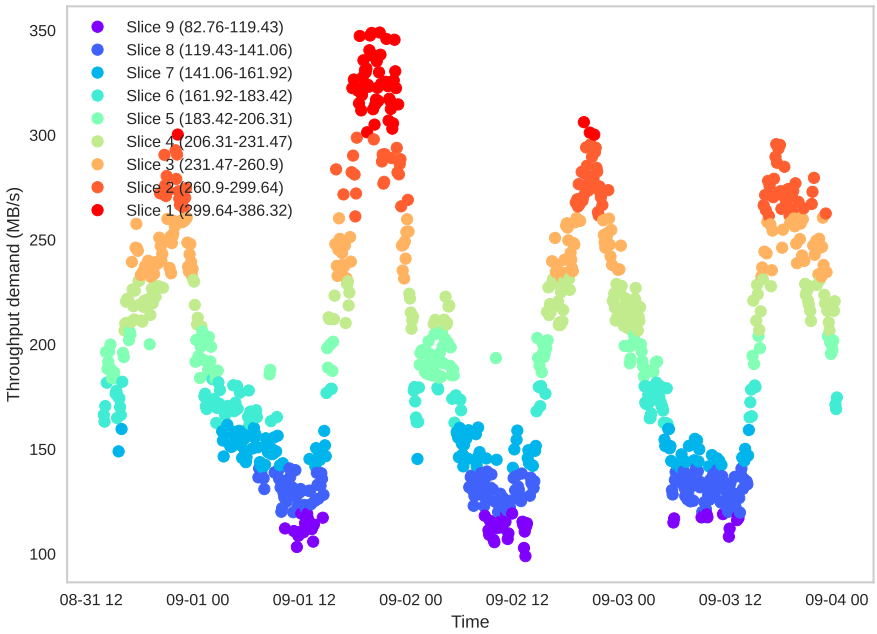


Figure 5.11: Assigned slices to each test time slot based on demand (N=9,Weight=netowrk cost)

In figure 5.12, we assessed the performance of the final provisioned slices in comparison to the true server throughput demand. The results of the evaluation indicate that NET-SPRO effectively eliminates slice underestimate and QoE loss. However, the evaluation reveals that 48 % of slices are overestimated. In the subsequent segment, we shall delve further into the performance of over-provisioning, and evaluate the extent of the additional cost incurred by this slice overestimation.

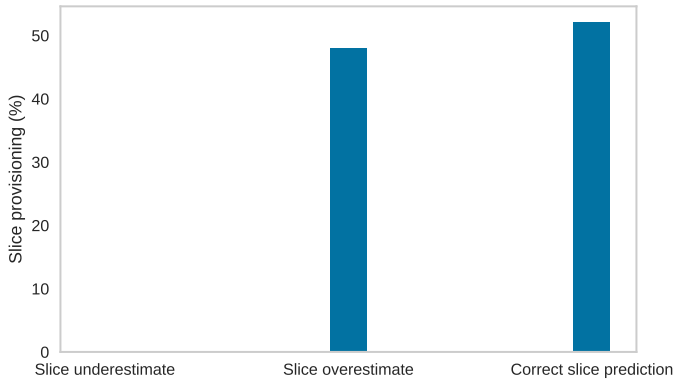


Figure 5.12: Slice provisioning analysis based on the proposed architecture

The NETSPRO system has been designed to demonstrate the potential benefits of network slicing for QoE-sensitive applications, specifically online gaming. The advantage of employing the NETSPRO solution over existing approaches is that it has the potential to mitigate QoE loss while simultaneously minimizing network costs. To support this claim, we conducted a comparative study between NETSPRO and four traditional industry-accepted solutions, which are described as follows:

- **Same Time as the Previous Week (STPW)** This technique predicts future throughput by replicating the previous week's throughput recorded at the same time.
- **The Highest Demand (HD)** This solution identifies the highest throughput demand in the history of a server's throughput demand and uses it for all future predictions in order to avoid QoE loss.
- **The Highest Demand in the Previous Week (HDPW)** Similar to "The Highest demand," this technique also considers the highest throughput demand but limits the search to the previous week.
- **Average of the Previous Hour (APH)** This method predicts future throughput for the next time slot by calculating the average of the last one-hour recorded throughput demand.

In Figure 5.13, we present the results of our comparative study between NETSPRO and the four aforementioned solutions. The plot illustrates the network cost and QoE cost of each approach, and it is evident that NETSPRO outperforms the other solutions

in terms of cost-effectiveness. The reason for this superiority is evident in Figure 5.14, where we compare the under-provisioning and over-provisioning performance of the solutions. Specifically, NETSPRO demonstrates a significant reduction in under-provisioning of throughput demand. On the other hand, the HD and HDPW solutions exhibit comparable total costs (network cost + QoE cost) but suffer from significant under-provisioning events and QoE loss, which are undesirable for our use case. The QoE cost of the HD and HDPW approaches can be attributed to higher throughput demands in the test dataset. Additionally, we have added the best-case scenario in Figure 5.13, which assumes that the server has access to the exact throughput capacity it demands at each time slot. By comparing the best-case network cost with the approaches in Figure 5.13, it is clear that STPW and APH are the most resource-efficient solutions. However, both approaches incur high QoE costs that offset their benefits.

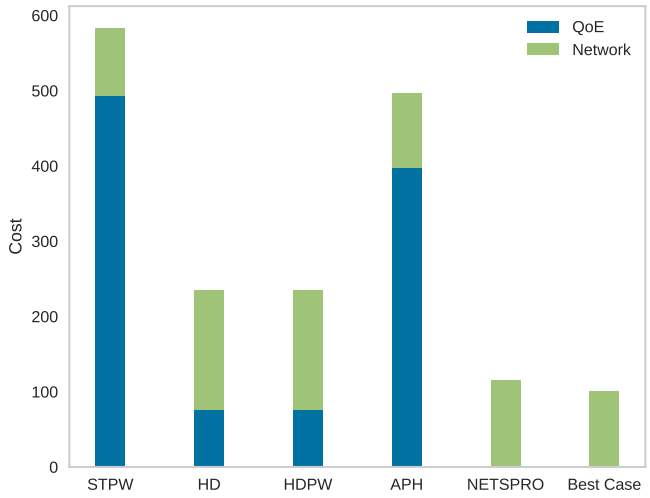


Figure 5.13: Comparative cost evaluation of NESTPRO with four solutions.

As mentioned in the previous section, our analysis indicated that NETSPRO experienced considerable slice overestimation. We have also emphasized throughout the paper that NETSPRO must allocate additional resources to minimize QoE costs. By comparing the best-case network cost with the NETSPRO network cost, we observe that NETSPRO only allocated 14.2 % more resources to prevent QoE loss. Furthermore, we note that the NETSPRO network cost was only 19.4 % higher than the APH network cost, while its QoE cost was 99.8 % less than APH. Therefore, this evaluation supports the fact that NETSPRO was able to achieve the primary objective described in the first chapter, which is reducing QoE loss in a cost-effective manner by leveraging 5G network slicing.

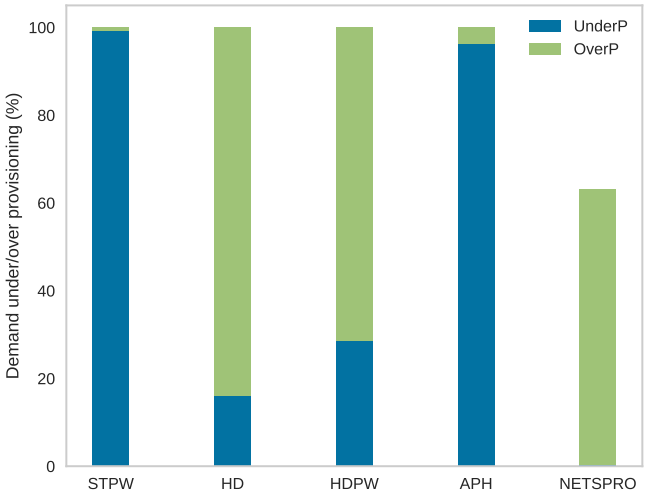


Figure 5.14: Comparative cost evaluation of NESTPRO with four solutions.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

The advent of online gaming has led to its rapid popularity among gamers, making it one of the most popular types of gaming. In this segment, online game providers are fiercely competing to secure a greater market share. QoE plays a crucial role in online gaming, and user satisfaction depends heavily on the experience they have with online game providers. Consequently, online game providers utilize various techniques to improve the QoE for their users. However, providing a high level of QoE comes with a significant cost, which poses a challenge for game providers as it conflicts with their cost objectives.

The next generation of wireless cellular technology, 5G, introduces a novel concept known as network slicing. By employing network slicing, network providers can create virtual networks with varying levels of QoS on top of physical networks. This technology brings new opportunities for online game providers, where they can rent a game-oriented slice based on their needs and reduce network costs while still offering competitive QoE. However, to utilize network slicing, game providers must be able to predict their throughput demand accurately, and network providers must be able to allocate the requested demand accordingly.

In recent years, ML and AI models have provided optimized solutions for addressing many network challenges. This thesis focuses on the application of ML and AI to network slicing to provide a cost-effective and QoE-aware solution for online game providers. The two primary challenges addressed in this study are (1) the prediction of server throughput demand, where the game provider must predict its server's demand to order the slice

accordingly, and (2) the optimization of a set of network slices that allows the game provider to switch between them and gives the network provider the opportunity to orchestrate its network more efficiently.

The investigation of network prediction solutions in Chapter 2 provided valuable insights into the network prediction problem, available solutions, and the appropriate direction to pursue. Drawing upon these findings, we formulated the network prediction problem in Chapter 3. In Chapter 4, we introduced a GRU-based DNN model that predicts feature throughput demand using a range of QoE and cost-driven strategies. The research presented in Chapter 2 also facilitated our exploration of how network slicing could be utilized. The extended study in Chapter 3 provided us with the necessary insights to formulate the network slicing problem from the network tenant's perspective. Based on these studies, we developed an ML solution using the weighted K-means algorithm, and we evaluated and selected the appropriate weight for our model, which we elucidated in detail in Chapter 4.

In Chapter 5, we presented a cost model that combined both network cost and QoE cost. This model allowed us to optimize the modules and compare and evaluate the solution with other existing solutions. We then optimized the network condition prediction module with the help of the introduced parameter, alpha. The chosen alpha value for the loss function enabled the DNN model to achieve balanced throughput demand prediction with respect to over-predicting and under-predicting costs. Furthermore, we assessed the slice optimizer module in terms of total cost and over-provisioning cost to identify the optimal network slice counts that could minimize both costs. Finally, to evaluate the proposed solution, we introduced four solutions that are commonly utilized in the industry and compared them based on network cost, QoE cost, and over-provisioning and under-provisioning ratios. Through this evaluation, our proposed solution exhibited a significant reduction in under-provisioning, which resulted in the minimum QoE cost compared to other solutions, and it also displayed an acceptable network cost.

6.2 Future Work

The selected loss function presented in chapter 4 is designed to have a linear loss in the over-predicting and a static loss in the case of under-predicting. Nevertheless, a more sophisticated loss function can be explored, where two separate and customizable functions could be responsible for calculating the amount of loss for under-predicting and over-predicting. This advanced loss function could enable the prediction model to have more precise predictions and potentially decrease the amount of over-predicting. For the DNN

proposed in the previous chapter, we selected a GRU model as the primary model of the network. As discussed, the GRU model demonstrates better performance compared to other traditional models such as LSTM in short-term prediction. However, further evaluation is necessary to assess the long-term prediction performance. Additionally, several research opportunities exist for modeling new machine learning methods such as reinforcement learning for demand prediction in both short-term and long-term scenarios. The collection and utilization of a more comprehensive dataset that includes a wider range of QoE, QoS, user, and online game metrics can offer numerous new research opportunities. However, gathering such data requires collaborations between online game providers, network providers, and online game users, which can be a challenging task. It is worth noting that incorporating geographical information could also provide more opportunities for utilizing CNNs and extracting local patterns. The slice optimizer module proposed in chapter 4 is an unsupervised solution for optimizing network slicing based on a single future. With the advent of 5G network slicing and the availability of new datasets, there are new research opportunities for employing state-of-the-art supervised machine learning and Reinforcement Learning (RL) methods. The proposed cost model in section 4.2 is based on the existing network cost table that was introduced for traditional networks. However, with the introduction of 5G and network slicing, additional variables that could affect the final cost of instantiating a slice, such as network re-orchestration, resource allocation, container deployment, initial delay, etc., will emerge. Creating a new network cost function based on these new attributes is a significant research challenge. The proposed solution architecture in this study has focused on demand prediction, QoE enhancement, and network slicing utilization and optimization from the game provider (network tenant) perspective. However, adapting 5G network slicing requires network providers to tackle the same challenges. Examining each of these issues from the network provider's viewpoint presents a promising research opportunity.

References

[1] Abbas Al-Thaedan, Zaenab Shakir, Ahmed Yaseen Mjhool, Ruaa Alsabah, Ali Al-Sabbagh, Monera Salah, and Josko Zec. Downlink throughput prediction using machine learning models on 4g-lte networks. *International Journal of Information Technology*, pages 1–7, 2023.

[2] Inc Amazon.com. Aws direct connect pricing, 2022. https://aws.amazon.com/directconnect/pricing/ [Accessed: April 2022].

[3] GSM Association. Official documentng.116 - generic network slice template version 7.0.

[4] Nagaiah Mohanan Balamurugan, Malaiyalathan Adimoolam, Mohammed H Alsharif, and Peerapong Uthansakul. A novel method for improved network traffic prediction using enhanced deep reinforcement learning algorithm. *Sensors*, 22(13):5006, 2022.

[5] Dario Bega, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 280–288, April 2019. ISSN: 2641-9874.

[6] Abdelhak Bentaleb, Mehmet N Akcay, May Lim, Ali C Begen, and Roger Zimmermann. Bob: Bandwidth prediction for real-time communications using heuristic and reinforcement learning. *IEEE Transactions on Multimedia*, 2022.

[7] Pierre-Louis Bescond. Cyclical features encoding, it's about time! https://towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca [Accessed: Feb 2022].

[8] Marc Carrascosa and Boris Bellalta. Cloud-gaming: Analysis of google stadia traffic. *Computer Communications*, 188:99–116, 2022.

[9] Asma Chiha, Marlies Van der Wee, Didier Colle, and Sofie Verbrugge. Network slicing cost allocation model. *Journal of Network and Systems Management*, 28:627–659, 2020.

[10] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. December 2014.

[12] Mark Claypool and Kajal Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, 2006.

[13] Habiba Elsherbiny, Hazem M Abbas, Hatem Abou-zeid, Hossam S Hassanein, and Aboelmagd Noureldin. 4g lte network throughput modelling and prediction. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6, 2020.

[14] Richard Etengu, Saw Chin Tan, Teong Che Chuah, and Jaime Galán-Jiménez. Deep learning-assisted traffic prediction in hybrid sdn/ospf backbone networks. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE, 2022.

[15] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100, 2017.

[16] Mihovil Grguric. Mobile Game Session Length: How to Track and Increase It. https://www.blog.udonis.co/mobile-marketing/mobile-games/session-length [Accessed: March 2023].

[17] 3GPP Working Groups. System architecture for the 5g system (5gs).

[18] Jia Guo, Yu Peng, Xiyuan Peng, Qiang Chen, Jiang Yu, and Yufeng Dai. Traffic forecasting for mobile networks with multiplicative seasonal arima models. pages 3–377–3–380, 2009.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Wenlu Hu, Ying Gao, Kiryong Ha, Junjue Wang, Brandon Amos, Zhuo Chen, Padmanabhan Pillai, and Mahadev Satyanarayanan. Quantifying the impact of edge computing on mobile applications. In *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, New York, NY, USA, 2016. Association for Computing Machinery.

[21] Math1013 Calculus I. Cosine as a function of directed angle in radians, 2017. https://www.math.hkust.edu.hk/~machiang/1013/Notes/tri_func.html, Accessed Feb. 01, 2023.

[22] Alexandros Kaloxylos, Anastasius Gavras, Daniel Camps, Mir Ghoraishi, and Halid Hrasnica. Ai and ml–enablers for beyond 5g networks. 2021.

[23] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time, 2019.

[24] Kittisak Kerdprasop, Nittaya Kerdprasop, and Pairote Sattayatham. Weighted k-means for density-biased clustering. In *Data Warehousing and Knowledge Discovery: 7th International Conference, DaWaK 2005, Copenhagen, Denmark, August 22-26, 2005. Proceedings 7*, pages 488–497. Springer, 2005.

[25] Asif Ali Laghari, Hui He, Kamran Ali Memon, Rashid Ali Laghari, Imtiaz Ali Halepoto, and Asiya Khan. Quality of experience (qoe) in cloud gaming models: A review. *multiagent and grid systems*, 15(3):289–304, 2019.

[26] Markus Laner, Philipp Svoboda, and Markus Rupp. Parsimonious fitting of long-range dependent network traffic using arma models. *IEEE Communications Letters*, 17(12):2368–2371, 2013.

[27] Aggelos Lazaris and Viktor K. Prasanna. An LSTM Framework For Modeling Network Traffic. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 19–24, April 2019. ISSN: 1573-0077.

[28] Scikit Learn. One Hot Encoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html [Accessed: May 202].

[29] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[30] Rishabh Madan and Partha Sarathi Mangipudi. Predicting Computer Network Traffic: A Time Series Forecasting Approach Using DWT, ARIMA and RNN. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pages 1–5, August 2018. ISSN: 2572-6129.

[31] Open Source MANO. OSM. https://osm.etsi.org/ [Accessed: April 2022].

[32] Lifan Mei, Jinrui Gou, Yujin Cai, Houwei Cao, and Yong Liu. Realtime mobile bandwidth and handoff predictions in 4g/5g networks. *Computer Networks*, 204:108736, 2022.

[33] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifa Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network. In *International conference on passive and active network measurement*, pages 34–47. Springer, 2019.

[34] Florian Metzger, Stefan Geißler, Alexej Grigorjew, Frank Loh, Christian Moldovan, Michael Seufert, and Tobias Hoßfeld. An introduction to online video game qos and qoe influencing factors. *IEEE Communications Surveys & Tutorials*, 24(3):1894–1925, 2022.

[35] Dimitar Minovski, Niclas Ogren, Christer Ahlund, and Karan Mitra. Throughput Prediction using Machine Learning in LTE and 5G Networks. *IEEE Transactions on Mobile Computing*, pages 1–1, 2021. Conference Name: IEEE Transactions on Mobile Computing.

[36] Hyeonjun Na, Yongjoo Shin, Dongwon Lee, and Joohyun Lee. Lstm-based throughput prediction for lte networks. *ICT Express*, 2021.

[37] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

[38] Chien-Nguyen Nhu and Minho Park. Dynamic Network Slice Scaling Assisted by Attention-Based Prediction in 5G Core Network. *IEEE Access*, 10:72955–72972, 2022. Conference Name: IEEE Access.

[39] Laisen Nie, Zhaolong Ning, Mohammad S. Obaidat, Balqies Sadoun, Huizhi Wang, Shengtao Li, Lei Guo, and Guoyin Wang. A Reinforcement Learning-Based Network Traffic Prediction Mechanism in Intelligent Internet of Things. *IEEE Transactions on Industrial Informatics*, 17(3):2169–2180, March 2021. Conference Name: IEEE Transactions on Industrial Informatics.

[40] Ali Yadavar Nikravesh, Samuel A. Ajila, Chung-Horng Lung, and Wayne Ding. Mobile Network Traffic Prediction Using MLP, MLPWD, and SVM. In *2016 IEEE International Congress on Big Data (BigData Congress)*, pages 402–409, June 2016.

[41] Qinghai Ou, Jigao Song, Yanru Wang, Zhiqiang Wang, Yang Yang, Diya Ran, and Lei Feng. A Method of Dynamic Resource Adjustment for 5G Network Slice. In Qi Liu, Xiaodong Liu, Lang Li, Huiyu Zhou, and Hui-Huang Zhao, editors, *Proceedings of the 9th International Conference on Computer Engineering and Networks*, Advances in Intelligent Systems and Computing, pages 929–936, Singapore, 2021. Springer.

[42] Nipun Ramakrishnan and Tarun Soni. Network Traffic Prediction Using Recurrent Neural Networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 187–193, December 2018.

[43] Tim Smith. Autocorrelation: What it is, how it works, tests, 2023. https://www.investopedia.com/terms/a/autocorrelation.asp, Accessed Feb. 10, 2023.

[44] SNDlib. Abilen network topology, 2022. http://sndlib.zib.de/home.action?show=/top.action%3Fframeset [Accessed: April 2022].

[45] Statista. Video Games - North America. https://www.statista.com./outlook/dmo/digital-media/video-games/north-america [Accessed: May 2022].

[46] Anurag Thantharate, Rahul Paropkari, Vijay Walunj, and Cory Beard. DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0762–0767, October 2019.

[47] Marcos Toscano, Federico Grunwald, Matías Richart, Javier Baliosian, Eduardo Grampín, and Alberto Castro. Machine Learning Aided Network Slicing. pages 1–4, July 2019.

[48] Hoang Duy Trinh, Lorenza Giupponi, and Paolo Dini. Mobile traffic prediction from raw data using lstm networks. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1827–1832. IEEE, 2018.

[49] Wen Wu, Kaige Qu, Peng Yang, Ning Zhang, Xuemin Sherman Shen, and Weihua Zhuang. Cost-Effective Two-Stage Network Slicing for Edge-Cloud Orchestrated Vehicular Networks. In *2022 IEEE/CIC International Conference on Communications in China (ICCC)*, pages 968–973, August 2022. ISSN: 2377-8644.

[50] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, 90:166–180, May 2018.

[51] Qiang Ye, Junling Li, Kaige Qu, Weihua Zhuang, Xuemin Sherman Shen, and Xu Li. End-to-end quality of service in 5g networks: Examining the effectiveness of a network slicing framework. *IEEE Vehicular Technology Magazine*, 13(2):65–74, 2018.

[52] Faqir Zarrar Yousaf, Michael Bredel, Sibylle Schaller, and Fabian Schneider. Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal on Selected Areas in Communications*, 35(11):2468–2478, 2017.

[53] Chaoyun Zhang and Paul Patras. Long-Term Mobile Traffic Forecasting Using Deep Spatio-Temporal Neural Networks. December 2017.

[54] Chuanting Zhang, Haixia Zhang, Dongfeng Yuan, and Minggao Zhang. Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks. *IEEE Communications Letters*, 22(8):1656–1659, August 2018. Conference Name: IEEE Communications Letters.

[55] Yin Zhang. University of texas. https://www.cs.utexas.edu/~yzhang/research/AbileneTM/ [Accessed April 2021].

[56] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, volume 17, pages 3602–3608, 2017.