Applied AI for QoE-Aware Video Service and Network Management

Hossein Ebrahimi Dinaki

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the
Ph.D. degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

With the vast growth of multimedia traffic on the internet, although revenue for the contributors considerably increases, video service management (VSM) becomes a complex and highly demanding task. Depending on the application, VSM systems have to deliver satisfactory Quality of Experience (QoE) to customers to retain a competitive market service. Furthermore, lack of sufficient QoE is the foremost reason for customer turnover. In this thesis, we investigate two QoE-aware VSM systems: server selection in the cloud gaming and fault diagnosis system in the video networks, where we apply artificial intelligence (AI) to address the challenges in these systems.

Cloud gaming (CG) is a high-performance and cost-effective cloud-based video game system, a promising paradigm for game users and providers, where all the computational tasks are offloaded to the cloud and players with low-end devices can play high-end games without the need for advanced hardware. However, resource management is a challenging task on this platform. This study aims to present optimal resource management in a CG system by considering both the service provider and player's benefits. Accordingly, we model an optimization problem and propose efficient metaheuristic methods: Boosted-GA and Boosted-PSO, for the GPU-based server selection in CG. The proposed methods simultaneously consider service providers' profits and players' experiences by maximizing GPU utilization.

The second management system investigates video network diagnosis to help service providers perform fault detection and isolation properly. Fault diagnosis is the heart of every VSM system. Network faults degrade QoE and must therefore be detected, isolated, and fixed. However, this is difficult because multiple entities own the end-to-end path's sections, and the video service provider (VSP) usually does not have access to the other entities' networks, such as the internet service provider (ISP) and the client's local network operator. In this study, we first collect a dataset of QoE and network

metrics from an actual video streaming testbed. Multiple videos are streamed from a video server to a client network through a simplified ISP network, while faults are generated in the ISP and/or client networks. Second, we propose a novel approach that shows that it is feasible for the VSP to localize the fault with AI's aid, using only QoE metrics, and without access to the faulty section. The two proposed deep learning methods of multi-layer perceptron (MLP) and long-short-term memory (LSTM) detect and localize the issues precisely.

Furthermore, for the QoE-aware VSM systems, when QoE degradation is detected, the users have already experienced that degradation on their screens. To address this lateness, we propose a hybrid state of the art: the deep learning method, i.e., BiLSTM-CNN, to forecast the QoE metrics in future time-steps before they appear on the client's screen. The proposed approach allows VSM systems to fix a problem before causing a serious issue at the end-user or at least reduce overall QoE degradation. This approach can be used in other VSM systems, such as resource allocation in wireless networks.

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to Professor Shervin Shirmohammadi, my supervisor, for his continuous and invaluable guidance, support, and encouragement.

I would like to extend my thanks to my thesis committee members: Prof. Carsten Griwodz, Prof. Peter X. Liu, Prof. Jiying Zhao, and Prof. Hossein Al Osman for their insightful comments and constructive feedback.

Also, I sincerely appreciate Dr. Mahmood Reza Hashemi for his constructive comments during my first journal publication.

Moreover, I would like to thank all members of the Distributed and Collaborative Virtual Environment (DISCOVER) Laboratory, for their cooperation, support and friendship.

Also, I would like to thank the Canadian federal and provincial governments for funding my research over the past couple of years, through QEII-GSST and Mitacs.

Most of all, I would like to express my deepest gratitude to my spouse for supporting and helping me to keep my feet on the ground. And special thanks to my daughter Mana for understanding why dad could not play all the time with her and wish all the best for my new baby Nami. I would also like to thank my parents and siblings for their continuous love and encouragement.

Table of Contents

Abstract ii

Acknowledgments iv

List of Figures viii

List of Tables xi

List of Abbreviations xii

Chapter 1. Introduction 1

1.1 Motivation 1

1.1.1 Cloud Gaming 1

1.1.2 Packet Video Network 3

1.2 Challenges and research problem 4

1.2.1 Cloud Gaming 4

1.2.2 Video Network 6

1.3 Methodology Approach 7

1.3.1 Server Selection in CG 7

1.3.2 Video Network's Fault Diagnosis 9

1.3.2.1 Fault detection and localization 9

1.3.2.2 QoE Forecasting 10

1.4 Contributions 11

1.5 Research Publications 13

1.5.1 Journals 13

1.5.2 Refereed Conferences 14

1.6 Organization of thesis 14

Chapter 2. Background and Related Works 17

2.1 The Architecture of Cloud Gaming Service 17

2.1.1 Related selection, placement, and provisioning strategies 19

2.2 Packet video network diagnosis 23

2.2.1 Proactive approach 23

2.2.2 Reactive approach 24

2.2.3 Related video network diagnostic mechanisms 24
2.2.4 QoE Forecasting 26
Chapter 3. GPU/QoE-Aware Server Selection for Multiplayer Cloud Gaming 29
3.1 Introduction 29
3.2 Server Selection Framework 29
3.3 Utilization, Delay, and User's Experience Models 31
3.3.1 GPU Utilization Model 31
3.3.2 Response Delay Model 32
3.3.3 User's Experience Model 34
3.4 Problem Statement 38
3.4.1 Problem Formulation 39
3.5 Proposed Metaheuristic Algorithms 41
3.5.1 Proposed PSO Model 41
3.5.2 Proposed Supporter-Algorithm 44
3.5.3 Proposed GA Model 48
3.5.4 Computational Complexity 51
3.6 Experiments, Results and Evaluation 52
3.6.1 Comparison with GA and PSO 52
3.6.2 Comparison with Four Common Bin-Packing Algorithms 54
3.7 Summary 58
Chapter 4. QoE-Aware Fault Diagnosis in Packet Video Networks 60
4.1 Introduction 60
4.2 Testbed and Data Collection Infrastructure 61
4.2.1 Tools 61
4.2.2 QoE Measurement 62
4.2.3 The Dataset 64
4.3 System Design and Training 65
4.3.1 MLP 66
4.3.1.1 Batch Normalization: 67
4.3.2 LSTM 68
4.4 Experiments and Performance Evaluations 71

4.4.1 Experiments 71
4.4.2 Experiments on Catastrophic Congestion 74
4.1.1 Discussion 79
4.4.3 Summary 80
Chapter 5. A BiLSTM-CNN Model for Multivariate Time-series QoE Metrics' Forecasting 82
5.1 Introduction 82
5.2 Proposed Forecasting Model 84
5.2.1 BiLSTM 85
5.2.2 CNN 85
5.2.3 BiLSTM-CNN 86
a. Activation function 87
b. Architecture 89
5.3 Evaluation and Results 91
5.4 Summary 94
Chapter 6. Conclusion and Future Works 95
References 100

List of Figures

Figure 1. A flow diagram of this thesis's contributions 13
Figure 2. Architecture of the RR-GaaS 19
Figure 3. Server selection in MCG. $P1$to PN are the game players, $DC1$to DCM are distributed datacenters, and $RS1$to RSL are the rendering servers in each DC. 30
Figure 4. Regression model curve fitting for logistic sigmoid 34
Figure 5. SSIM values versus Frame rate (f/s) for three games (a) BFV(FPS), (b) AoM (RTS), (c) War (RPG) 36
Figure 6. How a particle finds a new direction and position by comparing the current position (Xxt) with current local best position (lbest), global best position (gbest), and velocity (Vxt) 42
Figure 7. Boosted-PSO for server selection 47
Figure 8. Convergence Comparison between PSO and Boosted-PSO 48
Figure 9. Overall Boosted-GA Method for Server Selection 49
Figure 10. Progress in terms of utilization percentage compared with results achieved in [32] for (a) Boosted-GA and (b) Boosted-PSO Methods 53
Figure 11. (a) Capacity wastage of the Boosted-PSO and Boosted-GA algorithms for different number of players. (b) The number of utilized GPUs by the Boosted-PSO and Boosted-GA algorithms and corresponding QoE values for different number of players 54
Figure 12. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for the different number of players. Using F1 frame set 57
Figure 13. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for the different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F1 frame set 57
Figure 14. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for the different number of players. Using F2 frame set 58

Figure 15. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for the different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F2 frame set..... 58
Figure 16. Data Collection Testbed and Fault Diagnosis Framework..... 60
Figure 17. Steps of the QoE Calculation during the video playback..... 63
Figure 18. Normalized Value of 6 Features at 7 time-steps (0 to 6) during a single video playback..... 64
Figure 19. QoE Hysteresis for the three classes of the data. The classes are 0, 1, and 2 for no issues, ISP issues, and client issues. 65
Figure 20. The MLP model..... 67
Figure 21. A single cell of an LSTM Network 69
Figure 22. The LSTM model 70
Figure 23. Accuracy of training/validation for the six QoE metrics using (a) MLP (b) LSTM..... 72
Figure 24. (a) Confusion Matrix for MLP Classifier. (b) Confusion Matrix for LSTM Classifier 72
Figure 25. Classes' distribution for whole (new) data set 75
Figure 26. The four classes' distribution of the resampled training data sets. (a) an equal number of video samples created by oversampling classes 1, 2 and 3, to be equal to class 0 (majority). (b) An equal number of video samples were created by oversampling classes 1, 2, and 3 and undersampling class 0. (c) Oversampling the class 3 to be equal to class 0. (d) Oversampling the class 3 (minority) to be equal to class 1.. 77
Figure 27. Precision-recall curve. a) Imbalanced MLP b) Imbalanced LSTM..... 79
Figure 28. Precision-Recall Curve after training with Resampled_01 dataset. a) MLP b) LSTM..... 79
Figure 29. A sample use case of the proposed forecasting mechanism; Issue detection/isolation framework 83
Figure 30: BiLSTM; Architecture of one hidden layer 86
Figure 31. Performance comparison between SELU and tanh activation functions for LSTM, BiLSTM and BiLSTM-CNN Forecasting models in terms of RMSE and MAE. (a) LSTM, (b) BiLSTM and (c) BiLSTM-CNN. 89

Figure 32: BiLSTM-CNN Architecture. 90
Figure 33: Comparison of four forecasting ML Models in terms of RMSE for each metrics. 92
Figure 34: Comparison of four forecasting ML Models in terms of MAE for each metrics. 92
Figure 35: Forecasted vs actual QoE for four ML Models. 93

List of Tables

Table 1. R-Squared values of different genres 36
Table 2. Notations Table 37
Table 3. Performance using the six quality metrics 73
Table 4. Performance using calculated QoE only 73
Table 5. Number/percentage of the streamed videos in the new dataset. 75
Table 6. Classifiers' performance on the imbalanced test data 76
Table 7: Number/percentage of the streamed videos in the new training dataset (IMB: Imbalanced and RES: Resampled). 77
Table 8: Evaluation of the MLP classifier for the imbalanced and the resampled data sets. 78
Table 9: Evaluation of the LSTM classifier for the imbalanced and the resampled data . 78
Table 10: Specification of Activation Function's Test Models. 88

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
AoM	Age of Mythology
AS	Autonomous System
BFA	Best Fit Algorithm
BFV	Battlefield V
BP	bin-packing
CDN	Content Delivery Network
CG	Cloud Gaming
CPU	Central Processing Unit
DL	Deep Learning
DT	Decision Tree
FFA	First Fit Algorithm
FPS	First Person Shooter
FTP	File Transfer Protocol
GA	Genetic Algorithm
GaaS	Game as a Service
GPU	Graphics Processing Unit
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
ISP	Internet Service Provider
LR-GaaS	Local Rendering GaaS
MCG	Multiplayer Cloud Gaming
ML	Machine Learning
MLP	Multi Layer Perceptron
MOS	Mean Opinion Score
LSTM	Long Short Term Memory
NFA	Next Fit Algorithm
NPAW	Nice People At Work

NP-hard	Non-deterministic polynomial-time hardness
OTT	Over The Top
PD	Processing delay
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
ReLU	Rectified Linear Units
RNN	Recurrent Neural Networks
ROA	Route Optimization and Assurance
RPG	Role-Playing Game
RR-GaaS	Remote Rendering GaaS
RS	Rendering Server
RTS	Real-Time Strategy
RTT	Round-Trip Time
SSIM	Structural Similarity Index
SVM	Support Vector Machine
VM	virtual machines
VSM	Video Service Management
VSP	Video Service Provider
VQM	Video Quality Monitor
WFA	Worst Fit Algorithm

Chapter 1. Introduction

1.1 Motivation

1.1.1 Cloud Gaming

According to Newzoo's report, there were 2.5 billion active gamers worldwide in 2019, with an expected global games market of $152.1 Billions [1]. Remarkable advancements in cloud computing provide inexpensive and flexible opportunities for game service providers to deploy their games in the cloud, known as Cloud Gaming (CG). In the CG model, a.k.a Game as a Service (GaaS), or gaming on demand computationally intensive tasks such as the game engine, graphics rendering, encoding the game scenes is performed on remote servers in the cloud, and the game video is streamed to the player's end device [2]. The cloud servers have higher processing power and memory compared to the players' device, and the only requirement on the client-side is a broadband internet connection and ability to play video, not the need for high-end hardware. These features are attractive for game providers and have encouraged even big stakeholders such as Google, Amazon, Verizon, Apple, and Electronic Arts to develop their own video game streaming services [3]. CG's global market in 2017 was $802 million and is estimated to reach $6.944 billion by 2026 [4]. Onlive, G-cluster, and Gaikai were few of the pioneer companies of GaaS. Sony acquired Gaikai in 2012 and an essential part of Onlive in 2015 when the later discontinued its services. Currently, Sony's game video streaming service, PlayStation Now, is powered by technology from Gaikai. Another company, Broadmedia GC Corp., has been operating its own version of CG using its G-cluster technology since July 2016 [5]–[8]. Most recently, Google announced its own CG service, Google Stadia [9], with much fanfare.

To summarize, several CG service providers are available in the market e.g. PlayStation Now [10], Parsec [11], VORTEX [12], GeForce Now [13], Playkey.net [14] Google Stadia [9].

In CG, players operating on heterogeneous devices; e.g., PC, laptop, tablet, game consoles, desktops, set-top boxes, and smartphones, send their commands to the cloud game server which runs the game engine and makes the appropriate decision to produce the corresponding video frames rendered using a GPU and encoded by a video codec. Afterwards, the compressed frames are streamed to players through the network, and finally decoded and played on user's devices [15]. The tasks that need to be performed on the server-side require a well-organized resource management system to satisfy both the service provider's and end-user's requirements [16]. This management system would be more challenging for Multiplayer CG (MCG) cases.

The real-time and interactive nature of MCG makes response delay a very sensitive factor in the end-user's quality of experience, and thereby many works have been done to address this challenge[17][18][19]. Response delay consists of network delay, processing delay, and playout delay. Usually, the playout delay is assumed to be negligible and does not have a significant impact on player's game experience [20]. To reduce the network delay, normally a large number of games are supplied by service providers and replicated in multiple instances at different datacenters in the cloud, which are geographically distributed. Processing delay depends on the available processing power in the cloud server. Processing power, in turn, is determined by server resources such as CPU, GPU, memory, and storage, and the workload of the game sessions that run on the cloud server.

Furthermore, the server selection strategies would be more complicated, considering that there is heterogeneous computation power of the resources in cloud servers. Furthermore, CG has higher delay sensitivity compared to the

other platforms, such as online games [16]. Since each game, depending on its genre and pace, has different resource requirements, assigning the cloud servers to game players in this distributed network is one of the remaining challenges of CG platforms [21].

1.1.2 Packet Video Network

The Packet Video Network Diagnosis is another controversial challenge in Video networks management, which is evaluated in this study. It is safe to say that packet video has already taken over the Internet, when considering that video traffic constituted 75% of all IP traffic in 2017, and will constitute 82% of all IP traffic by 2022 [22]. Sandvine [23] reports that the top 4 biggest consumers of global IP traffic in 2019 were packet video systems: HTTP Media Streaming (12.8% of all IP traffic), Netflix (12.6%), YouTube (8.7%), and Operator IPTV (7.2%). The former 3 are also known as Over-The-Top (OTT) video, because they directly offer videos to consumers over the Internet, instead of the traditional distribution approaches of cable, broadcast or satellite.

Since OTT video uses the best-effort Internet instead of dedicated networks, the user's Quality of Experience (QoE) becomes a crucial aspect. QoE is a subjective measure of how satisfied the user with the video viewing experience.

Maintaining a satisfactory customer Quality of Experience (QoE) is of vital importance for video service providers such as video-on-demand providers (Netflix, YouTube, etc.), cloud gaming providers (Google Stadia, PlayStation Now, etc.) and videoconferencing providers (Zoom, Microsoft Teams, etc.). Network faults degrade QoE, and must therefore be detected, isolated, and fixed. To do so, continuous monitoring of the end-to-end path is required. However, in today's networks, each part of the end-to-end path belongs to a different Autonomous System (AS) that is typically owned by a different entity

such as the video streaming provider, the Internet Service Provider (ISP), the client's local network operator, etc. Therefore, although the video service provider is usually blamed by the customer in case of poor QoE, it usually does not have access to many parts of the network in order to localize the issue. Recently, it has been suggested that the big data generated in these networks can provide rich information to diagnose the network.

1.2 Challenges and research problem

In this section, we present the challenges addressed in this study for both CG and video network diagnosis.

1.2.1 Cloud Gaming

In CG, players send the commands from their heterogeneous devices to the cloud, and all the event processing, running the game logic, and rendering the game scene are done in the cloud, which then streams the video of the game scene to the players [15]. Normally a large number of games are supplied by service providers and replicated multiple instances at different datacenters in the cloud which are geographically distributed. In this case, CG service providers face a server selection problem which is still a serious issue in this platform [24], due to different resource requirements of each game, heterogeneous computation power of the resources, and high delay sensitivity of CG compared to the other platform such as online games [16].

Since in CG systems, the game's video updates must be streamed to the player's devices, the spatial and temporal quality of the video is a very important parameter to satisfy player's perceived video quality [2]. The better the spatial quality, the higher the resource requirement, e.g., bandwidth and processing power. Many works have been done to deliver high-quality video to players while minimizing resource usage. For example, authors in [25]

proposed a method considering the visual attention and object priority, and achieved a noticeable bitrate reduction while keeping good enough player's game experience. To improve the accuracy of the predicted attention maps, the effect of game state was considered in [26]. Authors in [27] conducted a test procedure to evaluate game performance for Samsung mobile phones. Regarding their evaluations, video smoothness (i.e., Frame rate and its stability) and graphic quality are essential factors for performance benchmarking apart from phones' battery, temperature, and swiftness. Also, frame rate contribution on the QoE and performance depends on the user type (e.g., passive or active), technology type (e.g., TV, VR …), task type, etc. [28]. The common belief for the maximum perception of the human eye system is 50 or 60 fps, which may be right for flicker's case [29]. However, subjective evaluation in terms of the object in motion shows that frame rates higher than 60 fps improve the motion picture quality, with the picture quality plateauing at 240 fps [30] [31]. Frame rates up to 120 fps are specified in the UHDTV (ultra-high-definition) video standard in ITU-R Rec. 2020 [32]. Although spatial quality is an important parameter for player's QoE in CG, in comparison with the other media, the frame rate has a crucial impact on game player's performance [33]. Therefore, game players are always looking for higher frame rates, and rates of up to 120 fps are now being demanded, depending on the game type. Game developers are trying to address this demand; for example, Battlefield 4 and Counter-Strike run at 120 fps with supporting hardware [34]. Since the game producers are delivering 120 fps, CG needs to support it. This demands more resources such as GPU in CG systems and makes the resource management problem more critical than before.

In terms of hardware resources, GPU's necessity is a fundamental difference between CG and other applications operating on cloud datacenters. GPUs play the leading role for fast video rendering in CG as a real-time service,

therefore using GPU-equipped cloud servers in this platform is an essential requirement. Also, GPU is the most expensive infrastructure that CG providers should pay for [21]. Therefore, its maximum utilization is vitally important from their profitability perspective. However, utilization should be efficient to satisfy more players as well, as they are the primary source of revenue for the CG provider [21]. Consequently, GPU is a critical parameter for server selection in the CG platform.

In the server selection problem, we consider two essential parities, i.e., service provider and player, involving in the cloud gaming system.

1.2.2 Video Network

In video network diagnosis, poor QoE is a key reason for customer turnover: both re-buffering and long initial buffering lead to abandonment, although the former 6 times more than the latter, and a single rebuffering event has 3 times the negative impact of a video bitrate change event [35]. In OTT, poor QoE occurs because of two main reasons: (1) network impairments such as congestion, router failure, router capacity degradation, router disabling, link disabling, etc. (2) the video player's overestimation or underestimation of the optimal video bitrate: since the player dynamically adjusts the video bitrate based on estimated network bandwidth, video buffer level, and computing limitations (screen resolution, CPU, etc.), if the estimates are not accurate enough, the player requests the incorrect video bitrate from the server, leading to rebuffering or quality degradation. Both underestimating and overestimating can lead to an unsatisfied customer [35].

In this study, we tackle the first reason: network impairments. Due to required real-time presentation of video frames, streaming video is particularly sensitive to congestion, latency, jitter, and network failure. But, a typical network of today either fails to identify OTT video traffic from other

traffic or cannot identify the paths that OTT video traverses across the network, or both. Furthermore, it is a difficult operational process to determine the source of poor OTT video delivery, due to the temporary or intermittent nature of many video-impacting network impairments.

Video-specific network fault management is therefore crucial in providing a satisfactory QoE to end users. This includes fault detection, fault diagnosis, and fault mitigation by alarming the operator and possibly providing suggested actions to remove the fault. But network fault management is very challenging due to the existence of a variety of physical or virtual devices and different operators in the end-to-end path: segments on the path could be owned by the video service provider within its datacenter network, content distribution network (CDN) provider, backbone Internet provider, ISP, or the end-user in its local network. These entities typically only have access to their own network sections.

For OTT video streaming, a fault can occur on the client side, video server side, or somewhere in the network between these two sides. The end result is the same from the customer's perspective: stalling/freezing of the video. The VSP, such as Netflix or Amazon Prime, who charges the customer for the video service, is the entity whom the customer expects to fix the issue. But fixing the issue can be challenging due to the said multi-ownership of the end-to-end path.

1.3 Methodology Approach

We address the technical challenges described in section (1.2) as follows:

1.3.1 Server Selection in CG

In CG, to address both parties' priorities, we proposed a method that optimally assigns cloud servers to the game sessions requested by game

players [36]. The method assesses the requested game and its required video quality in terms of frame rate and the load of the frames while considering the capacity of the eligible datacenters to allocate an appropriate cloud server to the player for the requested game session. We modeled the problem of server selection as an optimization problem focusing on maximizing both the GPU utilization and players' QoE. Our objective was the maximization of GPU utilization, which leads to the service provider's economic benefit, while increasing the player's QoE. To determine the priority of the two objectives, an end-to-end lag model weights them adaptively.

Then we proposed two efficient methods based on two metaheuristic algorithms, namely: Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). They worked well in terms of maximizing GPU utilization or minimizing capacity wastage and also improving the QoE.

However, the utilization percentage for small number of players in the proposed methods, i.e., GA, and PSO, is not sufficient enough and needs to be improved. For example, the results show almost 20 and 40 percent capacity wastage, for 20 players' case in PSO and GA-based methods respectively, however for 100 players' case, these wastages are half of these percentages. This problem shows that in some cases, a premature convergence takes place, and the algorithm is stuck in a local minimum. To generalize the algorithm and have almost a consistent efficiency for the different numbers of players, we should address this issue adequately. Towards this goal, in this study, we propose a new Supporter-algorithm which replaces the random regeneration part of the previous algorithms. The idea behind this Supporter-algorithm is that when the algorithm understands that it is stuck in a local minimum, it should boost itself to come out from that local point. This Supporter-algorithm plays a reinforcement role when the main algorithm approaches towards premature convergence. Besides, since it is not always in the main algorithm's

chain, its impact on the total speed of the main algorithms would be negligible. We refer to these two new algorithms as Boosted-GA and Boosted-PSO, hereafter.

There are two placement approaches in cloud literature: static and dynamic [37][38]. The static approaches place the users based on their service demands and the constraints of the problem statement. Dynamic or online approaches, in addition to placement, monitor or predict the demand and update the decision, accordingly. What we present in this study is a static approach, which is a placement at the initial step that works very well in terms of utilization with respect to the constraints.

1.3.2 Video Network's Fault Diagnosis

1.3.2.1 Fault detection and localization

In video networks, fixing the issue can be challenging due to the said multi-ownership of the path. It is therefore beneficial to equip the VSP with a system that can localize the fault even if the fault is not in an AS owned by the VSP, so the VSP can know where the fault is and take action; e.g., ask the owner of that AS to fix the issue or using another network provider if they have contracts with more than one.

In this study, we propose such a tool and study its feasibility. In particular, we design a machine learning (ML) system to both detect and isolate network impairments in packet video networks. Unlike existing ML-based fault detection approaches that use internal network QoS metrics, which are not necessarily available to the video service provider, or use predicted QoE metrics which do not always match with the actual QoE metrics, we use only the QoE metrics measured directly at the end user's video player, for both detection and isolation of the network issue. This is a big advantage because

the video service provider controls the video player and therefore has access to these QoE metrics. To the best of our knowledge, this is the first work that studies the feasibility of using only the video player's actual (not predicted) QoE metrics with ML to detect and isolate faults in packet video networks.

Towards this end, we employed an industry testbed at Ciena's Corp. to collect QoS and QoE metrics, the former used only for verification as ground truth, not as features in our dataset. We streamed a large number of videos over this testbed and created a dataset with congestion in different segments of the path. During video steaming, we collected video QoE metrics with timing data for 3 different cases: without congestion, with congestion on the client network, and with congestion on the ISP network. For simplicity, we refer to the latter two as *client-side* and *ISP-side*, respectively. Since VSPs are the content server owners and have comprehensive control over their own issues, we localize the issue only on the ISP and client side, which are not under their control. The collected data gives a multivariate time series, which we used to train two artificial neural networks (ANN): multi-layer perceptron (MLP) and long-short-term memory (LSTM). We chose MLP and LSTM, as they have been shown to perform well for addressing classification and multivariate Time-series problems [39][40].

1.3.2.2 QoE Forecasting

Today's VSM systems, typically predict QoE from the Quality of Service (QoS) parameters or the client-side's actual QoE metrics measured at the current time-step. But the former does not precisely reflect the users' experience, and the latter has a delay between QoE measurements at the client-side and the user's current experience. Accurate forecasting of QoE for the near future allows the service management system to take a proactive approach and fix delivery issues before they become a noticeable problem at

the end user, or at least reduce overall QoE degradation. In this thesis, we propose a method to prognosticate QoE metrics. We define a multivariate time series forecasting problem and model a hybrid state-of-the-art deep learning method, BiLSTM-CNN, to forecast the QoE metrics in advance. The proposed method can also be utilized in other management applications such as rate adaptation in DASH and resource allocation in wireless networks. Also, having a QoE forecasting mechanism in CG server selection systems, proposed in section 1.3.1, can increase security of the system when face with an incorrect QoE feedback from client-side.

1.4 Contributions

The primary contributions of this thesis are as follows:

1. Proposing two efficient GPU-based server selection methods in CG, i.e., improved versions of two well-known metaheuristic algorithms called particle swarm optimization (PSO) and genetic algorithm (GA), referred to as boosted-PSO and boosted-GA, respectively; our objective is to maximize GPU utilization, which will lead to economic benefits for the service provider and increase the player's QoE. We propose a new supporter-algorithm to replace the random regeneration part of the GA and PSO, the idea behind which is that when the algorithm understands that it is stuck in a local minimum, it boosts itself to come out of that local point. This supporter-algorithm plays a reinforcement role when the main algorithm approaches premature convergence.
2. Designing an ML system to both detect and isolate network impairments in packet video networks. We use only QoE metrics directly measured at the client-side to detect and localize the network issue

3. Designing a hybrid deep learning model for multivariate time series prognosticating of the QoE metrics' vector in packet video networks. The proposed model forecasts the QoE metrics in advance, before appearing on the client's screen. This approach can help VSP speed up the fault detection process and reduce overall diagnostic time and the end user's dissatisfaction. It can be used for other video service management systems such as resource allocation in wireless networks.

Additional contributions are as follows:

4. Proposing an optimization framework for simultaneously maximizing the service providers' profits and players' experiences; the proposed method optimally allocates the eligible servers' resources to a group of players' requested gaming sessions by assessing the video quality requirements in terms of frame rate and load while considering the rendering servers' capacity
5. Introducing a QoE model for different game genres and frame rates using an objective measure of video quality
6. Collecting a multi-class time-series dataset including the QoE and QoS metrics from a packet video network testbed, which would be appropriate for fault detection and isolation

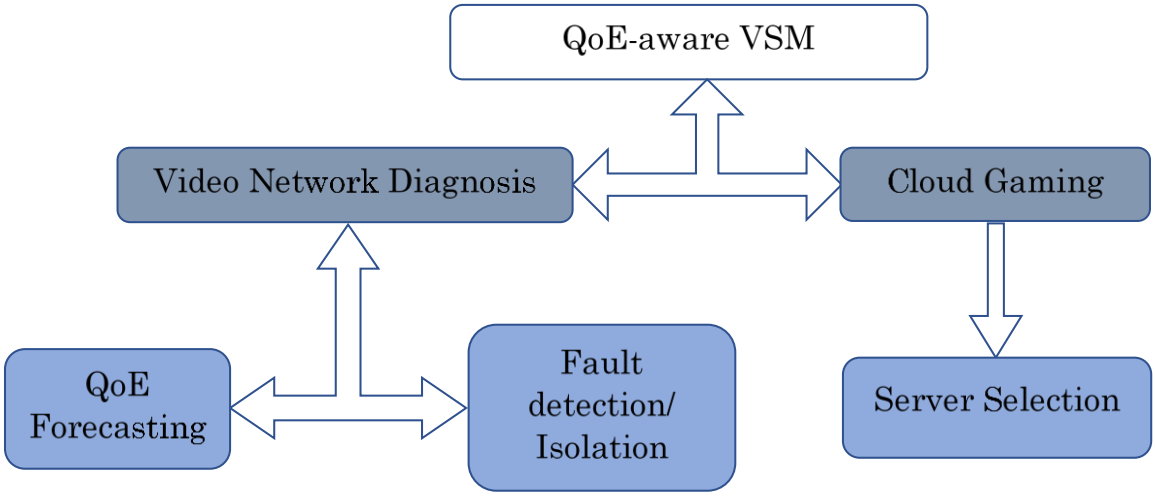


Figure 1. A flow diagram of this thesis's contributions

As presented in Figure 1, in this thesis, we endeavor to evaluate the QoE-aware VSM systems. We chose two systems, first, cloud gaming (chapter 3), as an interactive system that QoE has a significant effect on players' performance and should be considered in the server selection mechanism. Second, video network diagnosis (chapter 4), as the heart of every packet video network management system, where QoE metrics would be crucial for detecting the issues and even localizing them. Having access to the QoE metrics in advance (chapter 5) gives more time to the VSM systems for making the appropriate decision.

1.5 Research Publications

1.5.1 Journals

J1. Hossein Ebrahimi Dinaki, Shervin Shirmohammadi, Mahmoud Reza Hashemi, "Boosted Metaheuristic Algorithms for QoE-Aware Server Selection in Multiplayer Cloud Gaming", *IEEE Access 8. 2020 Mar 24*. (The contents of this paper will reuse in Chapter 3.)

J2. Hossein Ebrahimi Dinaki, S. Shirmohammadi, E. Janulewicz and D. Cote, "Deep Learning Based Fault Localization in Video Networks

Using Only Client-Side QoE," in *IEEE Transactions on Artificial Intelligence*, Dec 1st, 2020, doi: 10.1109/TAI.2020.3041816. (The contents of this paper will reuse in Chapter 4.)

J3. Hossein Ebrahimi Dinaki, S. Shirmohammadi, E. Janulewicz and D. Cote," A BiLSTM-CNN Model for Multivariate Time-series QoE Forecasting", *in IEEE Open Journal of Signal Processing (OJSP), Special Issue on Applied AI and Machine Learning for Video Coding and Streaming 2021 (Submitted).* (The contents of this paper will reuse in Chapter 5.)

1.5.2 Refereed Conferences

C1. DINAKI H. E. AND SHIRMOHAMMADI S., "GPU/QoE-Aware Server Selection Using Metaheuristic Algorithms in Multiplayer Cloud Gaming," 2018 16th Annual Workshop on Network and Systems Support for Games (NetGames), Amsterdam, 2018, pp. 1-6.

1.6 Organization of thesis

The roadmap for the rest of this thesis is outlined below:

Chapter 2 – Background and Related Works presents background knowledge on cloud gaming and packet video network diagnosis; it includes a review of three CG architectures and the current methods used for server selection and placement cost minimization. Furthermore, it presents a discussion on the proactive and reactive approaches to network diagnosis as well as the existing methods that employ these approaches for failure detection and isolation in the packet video network.

Chapter 3 – Proposed QoE-Aware Server Selection provides an optimization framework for rendering server (RS) selection within a cloud gaming system for maximizing GPU utilization, leading to economic benefits

for the service provider while increasing the player's QoE. To solve these NP-hard problems, we propose two efficient group assignment methods that are improved versions of two well-known metaheuristic algorithms called particle swarm optimization (PSO) and genetic algorithm (GA), which we refer to as boosted-PSO and boosted-GA, respectively. The proposed methods simultaneously consider the service providers' profits and players' experiences. This chapter presents a comparison between the proposed methods and the conventional GA and PSO algorithms as well as other existing methods concerning the utilization and number of used GPU.

Chapter 4 – Proposed Packet Video Network Diagnosis with Deep Learning Using QoE presents a data-collection procedure and deep-learning approach for detecting and isolating faults in a video network. We collect a dataset from an actual video streaming testbed, whereby multiple videos are streamed from a video server to a client network through a simplified ISP network. Congestions are generated in both the ISP and client networks. Using only the QoE metrics measured on the client-side, we use machine learning (ML)—specifically two deep learning methods of multi-layer perceptron (MLP) and long-short-term memory (LSTM)—to detect and localize the faults in the client network or the ISP network. Experiments show that our methods can localize the faults, outperforming the well-known ML methods of support vector machines (SVM) and decision trees (DT).

Chapter 5 – Proposed BiLSTM-CNN Model for Multivariate Time-series QoE Forecasting presents an efficient and accurate ML forecasting approach to anticipate the next state of the QoE in the packet video network. This proactive approach assists VSPs' network management systems in reducing fault detection, localization, and mitigation time. For this multivariate time series problem, we proposed a hybrid deep learning model, i.e., BiLSTM-CNN, that precisely forecasts the QoE metrics before appearing

on the end user's screen. This hybrid model is a combination of two state-of-the-art DL algorithms, i.e., bidirectional LSTM (BiLSTM) and convolutional neural network (CNN). The experiment results show that the proposed method outperforms other well-known support-vector regression (SVR), MLP, and BiLSTM methods.

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

Chapter 2. Background and Related Works

2.1 The Architecture of Cloud Gaming Service

CG is a flourishing gaming paradigm that brings noticeable benefits for both game players and providers. Three main architectures can be considered for this cloud-based gaming system [41]: Local Rendering GaaS, Remote Rendering GaaS, and Cognitive Resource Allocation.

In Local Rendering GaaS, the rendering operation is performed at the client-side by using the display instructions set, which represent the gaming graphic and is streamed from the cloud game server. Therefore, client's devices require powerful enough hardware to perform rendering. Since this scenario avoids video streaming over the network, it reduces the burden of the network significantly. However, computational capability and battery-dependency of the client's portable devices are the considerable challenges in this scenario.

The second architecture is called Remote Rendering GaaS (RR-GaaS), in which all of the computational tasks such as gaming logic, video rendering and capturing, video encoding and streaming are performed at the remote servers. In other words, only decoding and displaying the compressed video bitstream are performed at the client-side. Clearly, offloading all computational complexities to the cloud server-side enables players to play high-end games with low-end devices. However, the selected datacenter and server for remote rendering must satisfy the player's tolerable delay and its required processing capacity for the requested game type and quality.

The third architecture of cloud gaming is Cognitive Resource Allocation[42]. In this architecture, depending on the currently available resources in the client and the network, the game may be operated at the

client-side or in the cloud server. In other words, this system manages the game's computational operation based on its cognitive capabilities. As gaming is a real-time service and the load of the requested resources is completely stochastic, providing an adaptive optimization system for this cognitive system is a considerable challenge in this scenario.

Among the above architectures, as mentioned earlier, the RR-GaaS is currently available in the market and was implemented by several CG service providers including OnLive, StreamMyGame, Gaikai, and G-Cluster and is used by the new owners of these companies. That is why we consider RR-GaaS in our model in this study.

Since in this architecture, all complex computations are transferred to the cloud server, and only the video is streamed to the client-side, the quality of the transmitted video and the end-to-end delay are of vital importance for player's experience. To attain a tolerable response delay, both network delay and processing delay should be taken into account. The distance of the assigned datacenter and the network's traffic conditions are the two important parameters to provide an appropriate network delay (e.g., 80 msec) [43]. Selecting a desired rendering server (RS) in cloud datacenters, based on the game's load and computational capacity of the server, has a direct effect on processing delay. Also, there is a close connection among game genre, its required quality, and the processing complexity, and also the hardware rented by the service provider in the cloud server [33][44]. Consequently, this architecture involves both player's QoE and provider's benefits and profit and in any resource management system such as server selection, considering both parties' concerns is imperative. Furthermore, all these challenges will be intensified for the MCG case. Figure 2 shows the architecture of RR-GaaS. In the next section we will look at the related works in the server selection and placement.

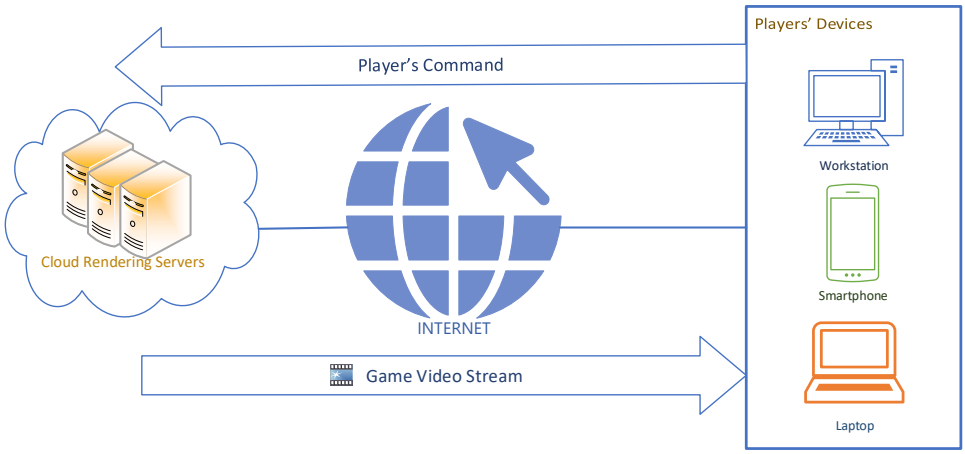


Figure 2. Architecture of the RR-GaaS

2.1.1 Related selection, placement, and provisioning strategies

There are different approaches to the problem of datacenter or server assignment to players in the literature. Some try to reduce the expense of the service provider while others prefer to increase the player's quality of experience. In this way, they employ infrastructure, network, and video parameters in their optimization problem.

The heuristic game hosting server selection method, presented in [43], tries to place the game according to the end-user's voting-based game-placement strategy. In this method, players vote for a game to be hosted on a server in order to serve more players for on-demand gaming. This gives a significant improvement in the number of served end-users compared to a random game placement. The server selection method introduced in [45] is looking for the optimal servers to serve a group of players' requests where the players are located in the same geographical region. The objective is to assign the servers in such a manner as to minimize latency in data transfer. The

authors in [46] formulate a multi-objective problem for virtual machines' provisioning in cloud gaming. The objective of the problem is to minimize inter-player delay among interactive gamers and electricity expenses while providing a good-enough response delay to the gamers. They present a gray wolf-based algorithm that efficiently solves the problem. The network parameters such as response delay and inter-player delay are considered as QoE. In all of the above papers, the servers have been selected only based on Round-trip delay time (RTT) or inter-player delay, and the video quality and computational resources' capacity have not been taken into account.

The work in [21] focuses on server acquisition between available datacenters based on the minimum price to support MCG. They consider four methods, three of them based on price: 1) Lowest Server Price Assignment, 2) Lowest Bandwidth Price Assignment, 3) Lowest Combined Price Assignment, and one based on wastage: Lowest Capacity Wastage Assignment. The results show that compared to Random Assignment and Nearest Assignment, considering wastage is helpful to achieve cost-effectiveness. Authors of [47] formulate the MCG Problem by considering the latency between the client, game server, rendering server, and the datacenter and then present several heuristics to address the server allocation problem for MCG. To solve these NP-Hard problems, they introduce a combination of price and wastage-based assignment algorithms with a greedy hill-climbing approach, called Lowest-Amortized-Cost, to take into account three parameters: server cost, bandwidth cost, and capacity wastage. Their experimental results show that for the same session size (number of players playing the same game), by increasing the latency threshold, the normalized cost of their proposed algorithm remains stable. Another work tries to optimize the inter-players delay between players who are interacting with each other by using an optimization algorithm that places all players' Virtual Machines (VM) and then adjusting the placement

[18]. It presents an optimization algorithm to reduce inter-player delay while keeping normal absolute response delay perceived by players, leading to a reduction of up to 30% in inter-player delay. Their focus is only on delay and bandwidth as quality of service (QoS) metrics, but not QoE in their models and evaluations.

A QoE-aware resource allocation framework is presented in [48] for a mobile cloud gaming system. The proposed system, referred to as EdgeGame, offloads computationally intensive tasks into the edge instead of the remote rendering server. The EdgeGame optimizes users' QoE in the edge using a reinforcement learning method. This system reduces delay and bandwidth costs. However, as they mentioned, it forces a noticeable investment to the service providers at the Edges.

An optimization problem for VM placement in Massively Multi-player Online Gaming was presented in [37]. The goal is to minimize the cost of the VM placement under delay constraints. Toward this goal, they map the problem to a multi-knapsack problem and propose a heuristic algorithm called Cost-aware VMs placement algorithm as a solution. The results show that their proposed method has higher effectiveness in terms of cost-saving, compared to random, Greedy Allocation Cost and Minimum Allocation Cost placement algorithms. In the cloud gaming system, the game's software should be installed on the distributed servers to avoid slowing down the loading and disrupting players' interaction. Authors in [49] formulate an optimization problem for server provisioning to minimize the software storage cost and the server running cost. Among the multiple heuristic algorithms which are employed to solve the problem, their ordered method and Genetic Algorithm have the best performance in terms of the minimum service cost and robustness for dynamic changes. However, both papers did not consider the impact of their placement on the client's quality of experience.

Authors in [50] evaluate QoS parameters and compare them in two different CG systems, namely OnLive and StreamMyGame (SMG). Based on their measurements, OnLive's downlink bandwidth is between 3 and 5 Mbps while SMG's is between 9 and 18 Mbps, showing that the server traffic is system dependent. Based on their results, game delays are also game-dependent. Then, they show that Processing delays (PD) of the two systems is related to the coding technique and hardware acceleration used in the video encoder. Their evaluation for the effect of computational power on PD shows that from a lowest-end to a highest-end CPU, there is merely a 20% improvement, and they concluded that SMG's longer PD must be due to the nature of its architecture/implementation.

The work in [51] proposes two VM placement algorithms to maximize the provider's total profit while providing just good enough QoE to players. They propose their algorithms for public and closed CGs (e.g., Hotels, Internet Café). The public algorithm sorts the servers according to the network latency for that player and then iterates among the sorted servers to support players based on their required resources. For closed CG, since maximizing the QoE is more important, their second algorithm sorts the servers based on quality degradation levels and then iterates through servers and creates a VM on the first server that can support the player. The results show that their heuristic algorithm attains very good results in terms of profits; however, as specified in their VM evaluation, considering VMs as utilization units have a noticeable overhead to the physical machine. Finally, [52] introduces a cost-effective way of dynamic resource provisioning for CG in Geo-distributed datacenters from the service provider's perspective. It reduces the cost of a CG service by using adaptive ways to select datacenters. It categorizes the users based on the game genres to meet the QoE requirements of different game types and cut down at least 25% of service costs. Their model is cost-effective from the service

provider's perspective, and they assign a separate and dedicated server for each player. However, in our model, depending on the load of the games, each server can be used for multiple players simultaneously, which is a more realistic condition.

Furthermore, to satisfy both parties, we align the service provider's benefits and the player's experiences and try to optimize these objectives. By efficiently utilizing the GPUs as the most expensive hardware in CG, we are trying to maximize the service provider's economic benefit, while the end user's experience is taken into account in our quality model.

2.2 Packet video network diagnosis

In a computer network, fault detection refers to discovering whether one or more network malfunctions have occurred, while fault isolation refers to identifying which part of the network is responsible for the detected fault. To detect and isolate faults, there are two main approaches: proactive and reactive.

2.2.1 Proactive approach

In the proactive approach [53], the system uses QoS parameters measured directly from network monitoring, and inputs them into an approximation function or a trained ML method to predict the end users' QoE. If this QoE starts to degrade, it tries to detect, isolate, and remedy the fault. This strategy is also used in other management systems such as resource allocation and traffic routing [54][55]. This approach has 3 shortcomings: (1) it requires full access over the end-to-end path, which is not possible for paths that go through multiple Autonomous Systems (AS) (2) it does not reflect the QoE of the end-users directly, and only uses QoE prediction, and (3) to collect

QoS parameters, traffic analysis tools such as Deep Packet Inspection (DPI) are typically used to extract the packets' contents and gather QoS metrics. However, DPI's effectiveness today is compromised by the dramatic and ever-increasing rise in packet encryption, leaving DPI solutions unable to determine much about the packet. Furthermore, DPI may cause performance degradation. Finally, DPI is unable to collect crucial video-specific information such as video player buffer size, buffering events, resolution downshifts, etc.

2.2.2 Reactive approach

In the reactive approach [56], the system monitors the QoE of the end-users. If QoE degradation is detected, it checks the network path to find the reason. Although this approach does consider the user's actual QoE, it has 2 shortcomings: (1) Similar to the proactive approaches, it needs to check the entire path, which is not possible in multi-AS paths, and (2) because it is a reactive method, by the time the system reacts, the user is already having unsatisfactory experience with the video.

2.2.3 Related video network diagnostic mechanisms

To overcome the above limitations, more intelligent network analytics can be used. Since networks transfer a massive volume of traffic, mostly video, they produce valuable data that could be used to manage the system. If this data is sufficiently large and relevant, using ML could be a good solution for video network analytics. However, the use of ML in this domain is understudied. In fact, we are aware of only 4 works, [57][53][56][58], that have studied the application of ML to video network analytics, described next.

The authors in [57] conduct a comprehensive measurement of the QoS and QoE metrics and develop a Decision Tree (DT) algorithm to find the root cause of mobile video QoE degradation. They use multiple vantage points

between the mobile device and the server to collect data. All the features collected and used for problem isolation are hardware parameters and QoS metrics. Although the approach is innovative and interesting, it does not use QoE metrics in its decision making. The only use case of QoE metrics is as ground-truth, not as features in the classification. In [53], a user-centric approach for anomaly detection in mobile networks is presented. It employs a parametric model that uses network metrics to predict QoE; hence, it does not use actual video QoE. The author in [56] presents a framework, consisting of Q-WATCH and Q-RANK, that identifies anomalous systems in a CDN, utilizing QoE as ground-truth and to detect the failures. When the anomaly is detected, isolation is done using end-to-end traceroute data. Q-WATCH uses correlation among multiple anomalous and normal end-to-end paths to localize the faulty routers or devices. Q-RANK then clusters the faulty devices for each system in the path e.g., cloud server, transit network, or user devices, and the system with the highest failure would be ranked as the anomalous system. While giving good results, this method requires the entire path's detail topology, which might not be available in multi-AS paths, and additionally must analyze a sufficient number of paths with overlap. Finally, a study on fault isolation in IPTV using ML is performed in [58]. This method tries to find the faulty equipment by integrating the objective QoE and QoS metrics and uses DT for fault prediction. Although this method takes QoE into account and its accuracy outperforms existing methods, it still requires QoS metrics that may not be available in multi-AS paths.

As we can see, all of the existing work suffer from one or more of the following shortcomings: (1) requiring QoS parameters, which might not be attainable in practice (2) predicting the QoE from QoS parameters, instead of using actual QoE (3) requiring network topology, network metrics, and comprehensive traceroute data. In contrast, our method during operation uses

only the QoE metrics measured at the client-side, and requires neither QoS metrics nor network topology. The latter two are used only as ground-truth for verification, not for operation.

2.2.4 QoE Forecasting

QoE plays a fundamental role in today's video service management system. In both proactive and reactive video network diagnosis approaches, the system usually estimates QoE at *current* time. Using the estimated QoE, the system detects the fault and isolates the issue by evaluating the network and the end-to-end traceroute data.

The authors in [59] presented a proactive mechanism for network fault forecasting using QoE to find the root cause in an IPTV network. Authors collected data from the Telenor IPTV delivery network, where the Set-top box (STB), WiFi, and Links traffic (i.e., two aggregate switches) were the vantage points. In each time-step, they presented the overall QoE for each user as "Unavailable", "Major", "Minor", or "OK". They then summarized the QoE to a binary class (1 for 'OK' and 0 otherwise). They used the QoE collected from the STB as ground truth and the network features from WiFi and switches as input features fed into their ML models. Using this dataset, they estimated whether the network will be faulty or not in the next time step. Another example is the mobile network anomaly detection approach presented in [53], which deploys an ML method to predict QoE metrics using network metrics. The author in [56] also presents a reactive framework, consisting of Q-WATCH and Q-RANK, that identifies anomalous systems in a CDN, utilizing QoE as ground-truth and detecting the failures. It monitors freezing time, and the bitrate drops at the client-side, and calculates QoE using an approximation function.

The author in [60], proposed a proactive service quality assessment process called Q-Score to estimate a single QoE metric using the network

performance metrics. An ML-based QoE prediction approach is introduced in [61] that can be applied in a proactive approach. It predicts rebuffering events from QoS parameters using a Bayesian Network; hence, it needs to monitor network parameters. Another QoE prediction method is presented in [62] and uses client-side metrics such as bitrate, frame rate, and user information. To increase accuracy, it adds a preprocessing block to extract appropriate features and feeds them into the prediction model. Yet another example is [63] which uses measured video metrics and users' survey data to estimate QoE. Data is collected experimentally including objective metrics such as total stall duration and the number of stalls for each video session. At the end of the video session, the survey data filled by the users were added to the dataset. Three ML models were then trained to classify QoE into five levels. Finally, a continuous QoE prediction method using a neural network is presented in [64]. The prediction model's inputs include rebuffering information, bandwidth-induced fluctuations, and QoE memory of the previous events. An extension of this approach is presented in [65] by the same authors, using a high-performance ensemble model and multiple datasets.

We can see that predicting the video QoE at the current time is a common approach in most proactive and almost all reactive methods. However, as aforementioned, an administrator does not usually have access to the whole end-to-end path. So, the reactive approach would be more common, although as explained before the reactive approach suffers from lateness.

Our goal is to introduce a semi-proactive approach by forecasting the QoE metrics vector for the next time-step using the past QoE sample sequences. This prognosticating approach gives more insight into QoE's next state and more time to the management mechanisms to take appropriate action. For example, the VSP admin has enough time to notify the related AS's admin to address the potential issue, or the wireless network scheduler can reassign the

resources to avoid QoE degradation. The data shape and the multivariate time series forecasting problem are evaluated in the chapter 5.

Chapter 3. GPU/QoE-Aware Server Selection for Multiplayer Cloud Gaming

3.1 Introduction

The combination of cloud computing and GPU advancements has made many real-time services possible, including Cloud Gaming (CG). As such, performing all the process-intensive tasks in the cloud frees players from upgrading their heterogeneous devices and installing new software, allowing them to play when and where they wish to. However, higher quality is always demanded by players. For instance, as frame rate has a significant impact on the player's gaming performance, the demand for a higher frame rate is increasing. On the other hand, service providers aim to offer cost-effective services. Therefore, management of the graphic-intensive CG service demand and maximizing the service providers' benefits is an issue that must be appropriately addressed. As remote GPU plays the central role of rendering and is the most expensive infrastructure rented by the service provider, its appropriate management is vital to address the above issue. For this, we efficiently formulate the problem and propose two methods to maximize GPU utilization and the users' quality of experience (QoE) at the same time, subject to the constraints of the servers. The server selection problem, two efficient metaheuristic solutions: Boosted-PSO and Boosted-GA, and the experiments' results will be discussed in this chapter.

3.2 Server Selection Framework

Figure 3 shows how, in a CG system, players can be assigned to the servers. As shown in the depicted framework, the game server (GS) is the point

that manages the connections between players and the cloud gaming system. Processing the state of the game, evaluating the resource requirement of the game, and allocating the resources to players are the main functions of the GS to arrange a game session between a group of players. For each player, the GS evaluates the state of the game and the required resources of the game scenes and then assigns the game player to a server in an eligible datacenter. It is assumed that the eligible datacenter has the minimum required delay, bandwidth, and processing power for the assigned game sessions. Therefore, the player can play the game in the datacenter where the game is rendered in a GPU equipped RS, and then the game's high-quality video for the new scenes are streamed to the player.

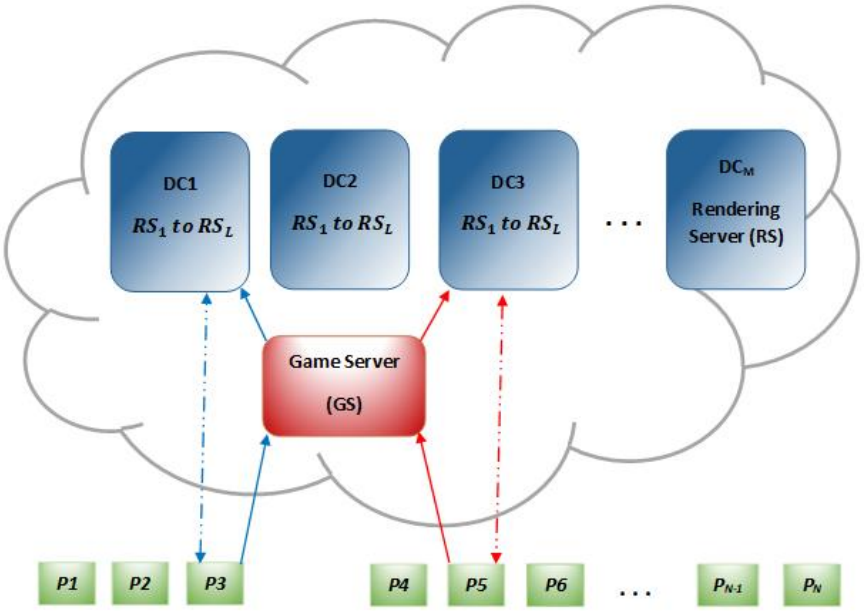


Figure 3. Server selection in MCG. P_1 to P_N are the game players, DC_1 to DC_M are distributed datacenters, and RS_1 to RS_L are the rendering servers in each DC.

3.3 Utilization, Delay, and User's Experience Models

3.3.1 GPU Utilization Model

GPUs play the leading role in speeding up the rendering of the game's video and reducing the processing delay of the CG system and consequently have a considerable effect on end user's experience. Besides, they are the most expensive hardware rented by providers [21]. Furthermore, not all of the servers in datacenters are equipped with a GPU. For these reasons, management of this resource is highly significant. Therefore, we consider its utilization as one dimension of the proposed server selection method. To this end, we formulate the GPU utilization model, and the formulation for the resource which is utilized by one player is as follows:

$$U_{ijl} = \frac{T_{ik} F_{ik}}{f_{jl}} \quad \forall\, i \in P\ ,\ \forall\, j\ \in D, \quad \forall\, l\ \in G \tag{1}$$

$$T_{ik} \leq \frac{1}{F_{ik}} \tag{2}$$

where player i is connected to GPU l in datacenter j. *P, D* and *G* are the set of players, datacenters and the GPUs, respectively. The proposed GPU utilization model is extracted from [66]. f_{jl} is the GPU frequency located in the server l in datacenter j. T_{ik} is the processing time of one frame in genre k and must be equal or less than $\frac{1}{F_{ik}}$ where F_{ik} is the frame rate of the game in genre k selected by player i. And, $T_{ik} F_{ik}$ is the processing time for doing the task of the requested game by player i in one second. In other words, this is the workload of each player corresponding to the selected game and frame rate. The U_{ijl} will be dimensionless, where cycle per second is the dimension of the f_{jl} and frame per second used for F_{ik}, and the T_{ik}, shows cycle per frame, or how many cycles take to each frame being processed. In our problem statement in section 3.3, we assume that each datacenter has L RSs, each of which can serve multiple players. All notations are summarized in Table 2.

3.3.2 Response Delay Model

The response delay is the time between a player command issued on the players' device (e.g., move, run, jump, fire, press a button, etc.) and watching the corresponding result on the display. High delay sensitivity of the RR-GaaS is a serious challenge in this architecture [16][17]. Intuitively, in server selection, the server with the shortest distance to the game player should be selected to do the rendering operation; however, many other challenges may be involved. For instance, the server may be already assigned to other players and not have enough resources for a new game, or the player is playing with another player, so the location of the other player should be taken into account to minimize the inter-player delay, and hence the closest server selection is not the only criterion anymore.

For the MCG described in Figure 3, we assume that all players are in the same cloud region; however, they can be assigned to different datacenters. Due to the reliability of the game servers, we assume that the system can assign gamers to the datacenters that pass the minimum required network delay; e.g., 80 msec. In this case, authors in [67] investigate the impact of the frame rate produced at the server on the response delay in the system, and show the synergy between the server-side tick rate and the client-side frame rate. They show that frames' process queuing in server-side and client-side is a considerable component shaping the end-to-end lag. The tick rate is the frequency of the game state update in the server, and the frame rate is the number of frames per second displayed to the user. To model the delay, we utilize their end-to-end lag data from the output of the video game simulation toolkit in [68], which is written in R. We utilized this data calculated for six different frame rates. Then we formulate the response delay based on the frame rate. To fit the data to an appropriate equation, we evaluate the regression model for several functions, such as $\sqrt{x}$, $ax^2 + bx + c$, log(x) where the logistic sigmoidal function, $Y(F) = 1/\left(1 - e^{(-\alpha - \beta.F)}\right)$ is the best fit function,

by its R-square value of 0.97, and α and β are the model's parameters. Equation (3) represents the response delay, RD, as a function of frame rate F.

$$RD(F) = \mathrm{Y}(F) = 1/\left(1 - e^{(-\alpha - \beta.F)}\right) \quad (3)$$

The R-square values indicate how close the sigmoidal function follows the input data. The response delay data and the corresponding fit curve of the function have been depicted in Figure 4. The curve has a sharper slope for lower frame rates however, if we compare the response delays for 30fps and 120fps, it is around 70 msec. Moreover, if we compare this value with the 80 msec allowable network delay [45], we realize that selection between the frame rate is still valuable and has a noticeable effect on the response delay.

In equation (4), we denote the λ coefficient, acquired from the derivative of the above sigmoidal function $Y(F)$:

$$\lambda_i = \left[\frac{dY(F)}{dF}\right]_{F=F_{ik}} = |\beta e^{(-\alpha-\beta.F)}/\left(1 - e^{(-\alpha-\beta.F)}\right)^2|_{F=F_{ik}} \quad (4)$$

As we described above, the frame rate has a noticeable effect on the response delay [67]. Hence, we use frame rate as the main variable in our problem statement (described in section 3.3), and we use λ as a priority factor, representing how quickly response delay changes with respect to the increase or decrease of frame rate. More about this will be shown in section 3.3.

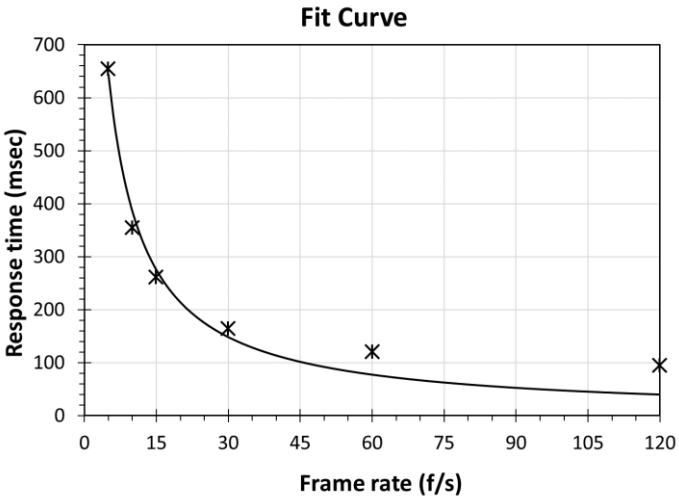


Figure 4. Regression model curve fitting for logistic sigmoid

3.3.3 User's Experience Model

As the user's satisfaction has a direct connection with the provider's profit, in any resource management system, end-users' QoE should be taken into account. In RR-GaaS, the game's video streams to the player's device. Compared with regular video streaming, video games require a higher frame rate. Due to higher interactivity and reactivity in video games, this feature would be an essential factor that has a noticeable impact on the performance of players, especially for fast-paced games. The study of Claypool et al. in [33] for first-person-shooter games shows that frame rate and resolution both have an impact on the perceived quality, but frame rate has a more significant influence on a player's performance than resolution. The tests conducted by [69] also shows the effects of frame rate on quality of experience. To consider the effect of frame rate, we modeled the QoE as a function of frame rate variation. Towards this end, inspired by the measurement study in [69] that represents some objective QoE metrics as a good approximation for MOS value, we deploy Structural SIMilarity (SSIM) as an objective quality of experience metric and formulate our QoE model. In this way, we chose a dataset of nine games from [44]. They are in three genres: Real-time strategy (RTS), First

Person Shooter (FPS), and Role-Playing Game (RPG). The videos were prepared in 6 different frame rates: 15, 30, 45, 60, 90, and 120 frames per second using the ffmpeg open-source software suite [70]. Then we employed a regression model for several functions to find the curve fit for the calculated quality.

The "four parameters logistic" function in (5) is the best fit function for the above measurements.

$$Q(F_{ik}) = d + \frac{a-d}{1+(\frac{F_{ik}}{c})^b} \quad (5)$$

where a, b, c, and d, are model parameters. A sample of the measurement's results for three video games, Battlefield V (BFV), Age of Mythology (AoM), and War, which are representatives of three genres, are plotted in Figure 5. The quality models obtained from this study apply only to the measured games. The SSIM values for different genres are different, but the trend of the curve by altering the frame rates is the same, and the same model-driven method can be applied to other genres. We report the R-squared values between the fitted curve and the collected data for the three genres in Table 1.

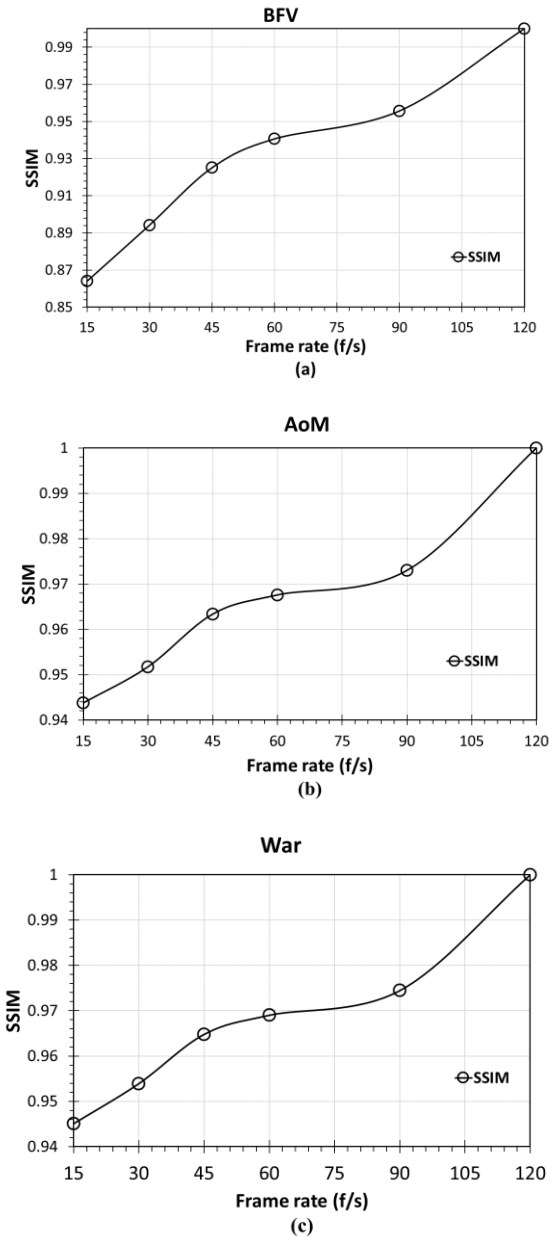


Figure 5. SSIM values versus Frame rate (f/s) for three games (a) BFV(FPS), (b) AoM (RTS), (c) War (RPG)

Table 1. R-Squared values of different genres

GENRE	RTS	FPS	RPG
R2	0.9372	0.8998	0.8778

Table 2. Notations Table			
Notations	Definitions	Notations	Definitions
N	Number of Players	T_{ik}	Processing time of one frame of the game in genre k, requested by player i
M	Number of Datacenters	$T_{ik}\,F_{ik}$	The time for processing the task of the player i in genre k in one second
L	Number of GPU equipped servers in each Datacenter	$RD(F_{ik})$	Response delay as a function of frame rate allocated for player i
K	Game genre′s number	α, β	Delay Model parameters
F_{ik}	fps allocated to game or player i with genre k	U_{ijl}	Resources utilized by player i in server l in datacenter j
x_{ijl}	Whether Player i is assigned to GPU_l (RS_l) in DC_j or not	$Q(F_{ik})$	QoE for player i, (P_i), playing game with fps F_{ik}
F_{mink}	Minimum required fps for genre k	a, b, c, d	QoE Model's parameters
F_{maxk}	Maximum fps for genre k	f_{jl}	frequency of GPU l located in datacenter j
f_{max}	Maximum GPU frequency	λ_i	Weighting coefficient
$v_1(x)$, $v_1(x)$	Violation functions	$f(x)$	Fitness function
X_i^t	Particle i's position at iteration t	V_i^t	Particle i's velocity at iteration t
lbest	Local best particle	*gbest*	Global best particle
ω	Inertial weight	*c1, c2*	PSO velocity model parameters
φ_1, φ_2	Random values between 0 and 1		

3.4 Problem Statement

An MCG system assigns multiple servers, between the eligible datacenters, to multiple players. The eligible datacenters are the datacenters that pass the minimum RTT requirement concerning the game type. As an eligible datacenter is one that has enough capacity for processing a new game and due to the necessity of needing enough bandwidth for an appropriate QoE[71], we assumed that each server in the datacenter has not only the rendering capacity but also sufficient bandwidth. Each server can serve multiple players, including sharing a single physical GPU among multiple users [72], depending on their game's load. The cloud server that has the maximum GPU utilization and still has enough capacity, would be assigned to the new players. The server selection problem is a problem from the bin-packing (BP) problem family, which is NP-hard [73] with no known solution to be run in polynomial time. Most of the BP based algorithms are heuristics, and the four popular ones are First Fit Algorithm (FFA), Best Fit Algorithm (BFA), Next Fit Algorithm (NFA), and Worst Fit Algorithm[74]. The goal in all of these BP algorithms is how to efficiently utilize the resources such that to maximize the utilization and, consequently, the service provider's profits. For a server selection problem, physical machines are bins, and the requests for resources, e.g., CPU, GPU, or storage, should be managed to attain a relevant packing result.

To simultaneously optimize both the player's QoE and the game provider's profits, we propose an optimization problem to maximize both utilization and the player's experience. Figure 3 illustrates the system architecture of MCG. In our proposed model, which is a group assignment, the players request the game video frame rates as a range instead of just one value; i.e., depending on the genre, the client's machine will ask for a minimum and a maximum (desired) frame rate. The proposed algorithms, which are run in

the GS, assign the players to an eligible cloud server by considering the requested loads and processing power of the servers. Since in our formulation the profit of the service providers, in terms of the maximum utilization of the RS and the users' QoE, are in the same line, our proposed algorithms try to maximize GPU utilization and allocate maximum frame rate to the players subject to satisfying the constraints of the servers. In the following section, we explain our problem formulation and our objective function in detail.

3.4.1 Problem Formulation

We assume that there are N players = $\{1,2,\dots,N\}$, M eligible datacenters D= {1,2,…, M}, L GPU included RSs in each datacenter G= {1,2,…,L}, and F = $\{F_{mink}, \dots F_{ik}, \dots F_{maxk}\}$ is a group of frame rates offered for each genre k. The processing time of each frame is T_{ik}, and $f_{jl} = f_{max}$ is the maximum processing capacity of the GPU. The frame rate set contains minimum frame rate F_{mink} required for the player's minimum QoE, the maximum frame rate F_{maxk} beyond which the player can no longer perceive the higher quality, and other frame rates between these two bounds. The objective function is then formulated based on the minimum requirements of the game, in terms of the frame rate, its corresponding processing time,T_{ik}, the maximum processing capacity of the GPU, f_{max}, and the maximum quality that can be perceived by the player.

Normally, in resource allocation systems, the decision-maker prefers to utilize smaller loads, in our case frame rate, to achieve maximum utilization. Nevertheless, as we discussed in sections 3.2.2 and 3.2.3, the smaller frame rate leads to a perceived higher response delay and also lower user experience. On the other hand, assigning the highest frame rates would lead to lower servers' utilization and more capacity wastage. To avoid these, we employ a weighted sum approach, where these weights would adaptively control both objectives of the problem with respect to random variations of input variables. In this way, since our quality objective and the response delay model are aligned, we used λ from eq. (4) in eq. (6) as a priority factor which reflects the

rate of changes of the response delay curve, regarding the frame rate. This ratio shows how quickly response delay changes concerning the increase or decrease of frame rate. By using this variable weight, the decision-maker can dynamically determine the preference of the corresponding objective. Regarding the above definitions, our objective function is as follows:

O(x_{ijl}):

$$Max(\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{L}((1-\lambda_i).\frac{(U_{ijl})^r}{\hat{m}.\hat{l}}+\lambda_i.Q(F_{ik}))).x_{ijl} \qquad (6)$$

Subject to:

a) $F_{mink} \leq F_{ik} \leq F_{maxk}$ $\quad \forall\, i \in \{1,2,\dots,N\},\ \forall\, k \in \{1,\dots,K\}$

b) $\sum_{i=1}^{N}\sum_{j=1}^{M} T_{ik}\, F_{ik}.x_{ijl} \leq f_{jl}$ $\quad \forall\, l \in \{1,2,\dots,L\}$

c) $T_{ik} \leq \frac{1}{F_{ik}}$ $\quad \forall\, i \in \{1,2,\dots,N\},\ \forall\, k \in \{1,\dots,K\}$

d) $\sum_{j=1}^{M}\sum_{l=1}^{L} x_{ijl} = 1,$ $\quad \forall\, i \in \{1,2,\dots,N\}$

e) $x_{ijl} \in \{0,1\}$ $\quad \forall\, i \in \{1,2,\dots,N\}, \forall\, j \in \{1,2,\dots,M\},\ \forall\, l \in \{1,2,\dots,L\}$

f) $\hat{m}.\hat{l} = \sum_{l=1}^{L} \min\,(\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ijl}\ \ , 1)$

The above objective function has two parts. The first part is utilization, where U_{ijl} is derived from section 3.2.1 and represents the utilization of the GPU l in the datacenter j when player i, playing in genre k, is assigned to server l. Also, $\hat{m}.\hat{l}$ is the number of the eligible used servers, and r > 1 is a constant value [75]. The second part is related to the player's QoE concerning the assigned frame rate. The function of the QoE model is derived from section 3.2.3. The $0 < \lambda_i < 1$ is a weighted average coefficient which increases and decreases with the decrease and increase of the frame rate, respectively. Constraint (a) bounds the acceptable range of the frame rate for the genre k, which is played by player i. Constraint (b) determines the maximum allowable workload for each GPU. The workload of the RS_l must be equal or less than the

maximum processing capacity f_{max} . Here we assume that all servers are working at their maximum frequency. The processing time of a single frame in genre k must be equal or less than $\frac{1}{F_{ik}}$ which is mentioned in constraint (c). Moreover, constraint (d) recognizes that only one server l in only one datacenter j can be assigned to each player i. The x_{ijl}, in constraint (e), is a binary value where one means that the server l in the datacenter j is assigned to the player i and zero represents no allocation. Finally, (f) calculates the number of the used GPUs.

3.5 Proposed Metaheuristic Algorithms

In this section, we present the two proposed metaheuristic algorithms based on Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) and also our proposed Supporter-algorithm for boosting of these two methods.

3.5.1 Proposed PSO Model

PSO, a well-known member of evolutionary algorithms' family, is first presented by Kennedy, Eberhart in 1995 [76]. Social behaviour of a bird in the flock, a fish in a school, or an insect in a swarm, constructs the foundation of the PSO algorithm. In this algorithm, each single candidate solution in the search space is called a particle. They provide individuals of the population in the feasible search-space. The individuals communicate with each other to move toward the same regions of problem space. To find the best direction, they adjust their velocity and position iteratively by using their own knowledge, and the information comes from the neighbour swarm members. Figure. 5 shows how an individual updates its position and velocity in a PSO algorithm. Each particle tries to update its current position and velocity regarding the distance between its current position and its best-found position in the feasible search-space. The best-found is updated as the better position found by the particles, and the distance between its current position and global-best achieved among all particles in the swarm. PSO has been successfully deployed in many areas,

such as fuzzy system control, function optimization, artificial neural network training, etc. [77][78]. Figure 6 represents the process in PSO to update the location of the particles.

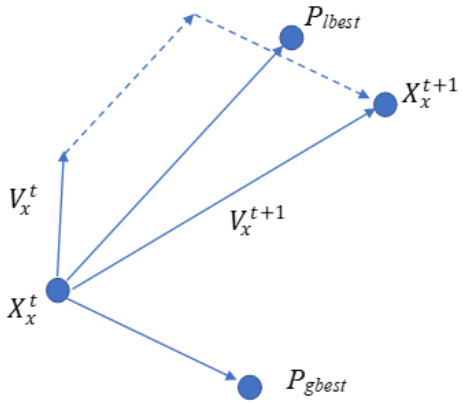


Figure 6. How a particle finds a new direction and position by comparing the current position (Xxt) with current local best position (lbest), global best position (gbest), and velocity (Vxt)

Now, we propose the steps of our PSO based model for the problem presented at section 3.3.1:

Step 1: Initial Population: Initialize population of the individual solutions, contains their position, fitness value, velocity. In our proposed model, each particle represents a cloud server allocation outline, $X = \{x_{111}, x_{112},\ldots, x_{1ML},x_{211},x_{212},\ldots, x_{ijl} ,\ldots,x_{NML}\}$ where N is the number of players, M is the number of eligible datacenters, and L is the number of GPU-equipped servers in each datacenter. In this particle, $M \times L$ binary components assign to each player an identifier code, and each binary number in this code represents the cloud server that could be assigned to this player. Since normally we know the number of servers in each datacenter, this particle codes the datacenters as well. We deploy the decimal version of the above binary codes inside the computation of the algorithm, and then we convert the decimal number to the binary to determine the server selection results in the last iteration. So, the

definition of an individual particle is: particle = $\{P_1, P_2 \ldots P_i \ldots P_N\}$. We set the number of particles for our experiments to 25.

Step 2: Evaluate particles' fitness: Evaluate the fitness of individual particles with the fitness function.

Fitness Function: In this study, our fitness value comes from the objective function (6) and its constraints. This fitness value demonstrates the utilization of the GPUs, the players' QoE, and some penalty values to handle the constraints.

To reflect all of the goals of the problem in the fitness function, we define some violations' functions from the above constraints and add them to the objective function as a penalty function. The defined violation function $v_1(x)$ and $v_2(x)$ come from constraints 6(b) and 6(d) respectively:

$$\mathrm{v_1(x)} = \max\left(\sum_{i=1}^{N}\sum_{\mathrm{j=1}}^{\mathrm{M}} \frac{\mathrm{T_{ik}\, F_{ik} . x_{ijl}}}{\mathrm{f}} - 1\,, 0\right) \tag{7}$$

$$\mathrm{v_2(x)} = \max\left(\sum_{\mathrm{j=1}}^{\mathrm{M}}\sum_{l=1}^{\mathrm{L}} \mathrm{x_{ijl}} - 1{,}0\right) \tag{8}$$

Therefore, we define the following fitness function with respect to the objective function (6) and the punishment functions (7) and (8):

$$\mathrm{f(x)} = \mathrm{O(x)} - \mathrm{C_1 v_1(x)} - \mathrm{C_2 v_2(x)} \tag{9}$$

Where C_1 and C_2 are penalty coefficients and $\mathrm{x} = \mathrm{x_{ijl}}$.

Step 3: Update Best particles, lbest, Global Particle, gbest:

$$lbest_i^t = \begin{cases} X_i^t \quad if \quad f(X_i^t) > f(lbest_i^t) \\ lbest_i^t \qquad\qquad\qquad else \end{cases} \tag{10}$$

The gbest is replaced by the best lbest among the particles.

Step 4: Update velocity and move each particle to the new position:

$$V_i^{t+1} = \omega V_i^t + c1 * \varphi_1 * (lbest_i^t - X_i^t) + c2 * \varphi_2 * (gbest_i^t - X_i^t) \quad (11)$$

$$X_i^{t+1}. = X_i^t + V_i^{t+1} \quad (12)$$

Where ω is an inertial weight that creates a balance between global exploration and local exploitation for each particle, and it will be damped in each iteration. c1 and c2 are the weights, that manage the particle's tendency towards its local best location, lbest, or swarm's best location, gbest. To determine the weights during the iterations, we employed the Time-Varying Acceleration Coefficients (TVAC) presented in [79], which is an efficient technique to speed up the algorithm. V_i^t is the velocity of the particles, and X_i^t is the particles' current position. Also, $\varphi_1$and φ_2 are random numbers between 0 and 1.

Step 5: Stop, if convergence or Termination condition is satisfied. Otherwise, go to step 6

Termination condition: The algorithm should stop when it reaches a plateau. The terminating condition of our algorithm is: after 100 iterations, there is no progress for the best fitness value in the last 50 iterations, or the algorithm stops at the 400th iteration.

Step 6: If the Supporter-algorithm's condition is satisfied, run the Supporter-algorithm (described in the next section), otherwise go to step 2.

3.5.2 Proposed Supporter-Algorithm

In our previous work [36], the proposed algorithms were successful in terms of the utilization percentage for a higher number of players; however, they suffered for a small number of players. The results show almost 20 percent capacity wastage, for example, for 20 players' case. To address this issue and have more consistent results, we proposed a heuristic algorithm that can boost the main metaheuristic algorithm and achieve a steady result for various conditions. Since the main reason for this behaviour is a premature

convergence of the proposed algorithms, we developed the regeneration part of the algorithms to avoid trapping in local minimum or maximum. The use of this re-initialization section can ensure us that no premature convergence takes place; however, to avoid increasing the complexity of the total algorithm we consider some conditions to bring this Supporter-algorithm in the main process loop only for some exceptional cases.

In two cases, this Supporter-algorithm may come to the calculation chain. Firstly, for the case that the algorithm is repeating a single positive fitness value in a large number of iterations. In this case, the algorithm may trap in a local minimum, and there is no progress in the results. Therefore, to avoid premature convergence, the Supporter-algorithm randomly transfers some loads from the server with high capacity wastage to the servers with low capacity wastage. It would be similar to mutation, but the action is more wisely. Secondly, the main algorithm is repeating a negative fitness value for a large number of initial iterations. This is the case that one or some servers are overloaded, and in this case, this Supporter-algorithm transfers part of the load of the overloaded server to the next servers that have enough capacity to handle this game's load. By employing this Supporter-algorithm, most likely, the main algorithm comes out from the premature convergence conditions. It is worth noting that this Supporter-algorithm is just boosting the main algorithm and is not looking for an optimum solution. The complexity of the Supporter-algorithm explained in the section 3.4.4 shows that it does not increase the total complexity of the main algorithm. Figure 7 describes our Boosted-PSO server selection method. The pseudo-code of the proposed Algorithm is as follows:

Algorithm 1 Pseudo code for Supporter-Algorithm in Boosted-PSO

N = Number of Individuals (or Particles)

Iter = Current iteration number

Thrsh = Threshold for entering the Supporter-algorithm

if Iter > Thrsh && fitness.value (Iter) == fitness.value (Iter - Thrsh) OR Iter > Thrsh /2 && fitness.value <0 && fitness.value (Iter) == fitness.value (Iter – Thrsh /2)

for *i* = 1 to N

- if fitness.value (individual *i*) <= 0
  - for j = 1 to the number of players in individual *i*
    - if assigned server to the player *j* is overloaded
      - Transfer the player *j* to the next server with enough capacity to process the game session of this player.
      - Update capacity of the source and destination servers after moving.
    - end if
  - end for
- else if fitness.value (individual *i*) > 0
  - for *j* = 1 to the number of players in individual *i*
    - if the player *j* assigned to the server with maximum capacity wastage.
      - Transfer the player j to the next servers with lower capacity wastage and enough capacity to process the game sessions of this player.
      - Update capacity of the source and destination servers after moving.
    - end if
  - end for
- end if

Update position of individual *i*

Update fitness of individual *i*

Update velocity of individual *i*

end for

end if

Figure 8 compares the convergence of the PSO based algorithms with and without Supporter-algorithm for the same dataset. For the sake of page limitation, we are not showing the iterations of the algorithm. In this figure,

for 20 players, the PSO starts with a good initial fitness value, and the Boosted-PSO converges to the higher value at the end. To attain this result, the Supporter-algorithm was called nine times into the main chain of the algorithm in iterations, where five of them created progress (jumps in the curve): iterations 30, 50, 60, 80, and 90. However, the previous PSO version does not experience any changes in fitness value after iteration 105 (horizontal straight line), and it converges to the lower value. This comparison shows the behavior of the Supporter-algorithm for boosting the PSO and avoiding getting stuck in a local minimum.

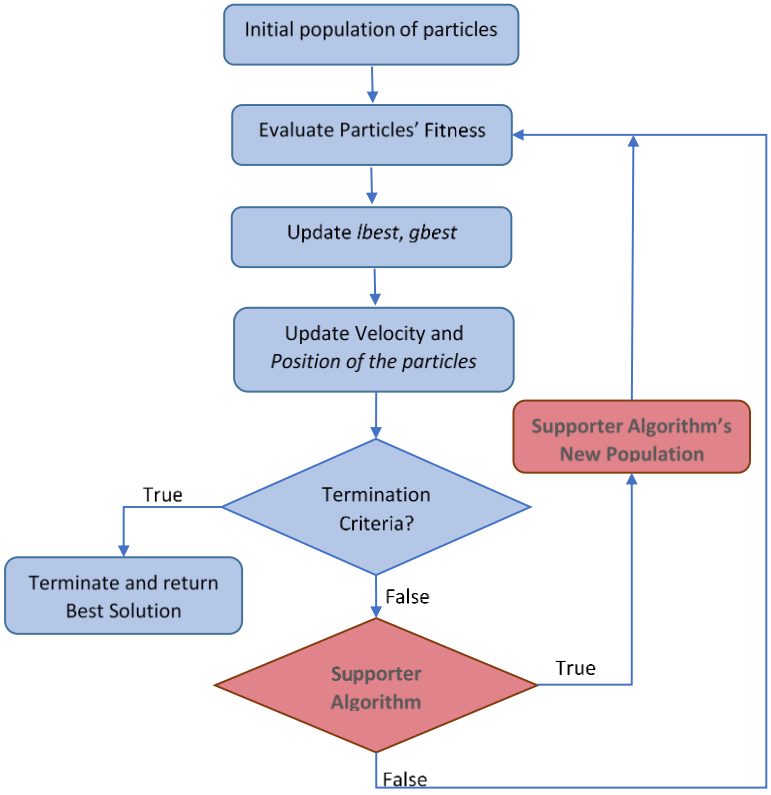


Figure 7. Boosted-PSO for server selection

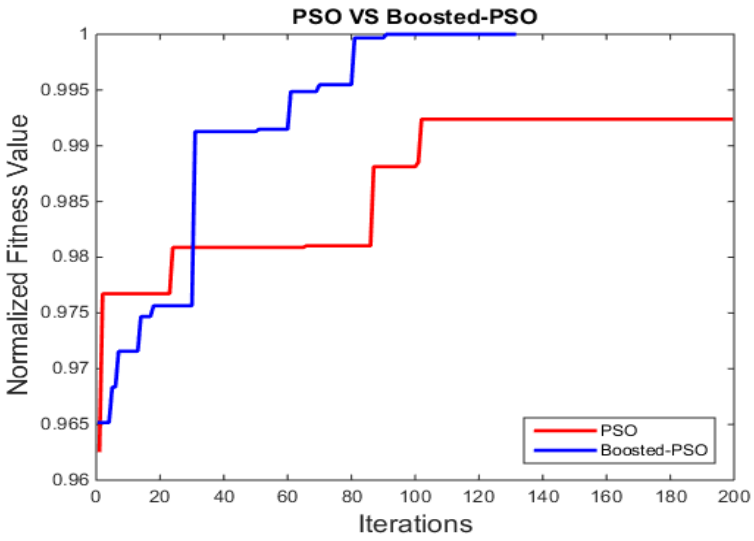


Figure 8. Convergence Comparison between PSO and Boosted-PSO

3.5.3 Proposed GA Model

Genetic Algorithm is an evolutionary algorithm that becomes popular from the 1990s [80]. In this algorithm, the first individual chromosomes are created randomly as the initial population or generation. A fitness function that reflects the objectives and constraints of the problem will be used to score the chromosomes. The values of the fitness function of the current chromosomes determine which chromosomes are more powerful to be used for producing the next generation. The weak chromosomes will be eliminated, and the most powerful ones have a higher chance of staying alive. The flow chart depicted in Figure. 9 describes our proposed Boosted-GA server selection process, which has the following steps:

Step 1. Initial Population: As with all GA models, we set the algorithm's parameters, such as choosing population size, mutation rate, and stopping condition. To produce an initial population for the GA model, we followed the same procedure as PSO, in section 3.4.1, for generating the initial particles. The definition of a chromosome is Chromosome = $\{P_1, P_2 \ldots P_i \ldots P_N\}$. Each P_i symbolizes a gene, which is the basic unit of a genetic operator. We set the

initial population to 50 and generate the first generation of chromosomes randomly.

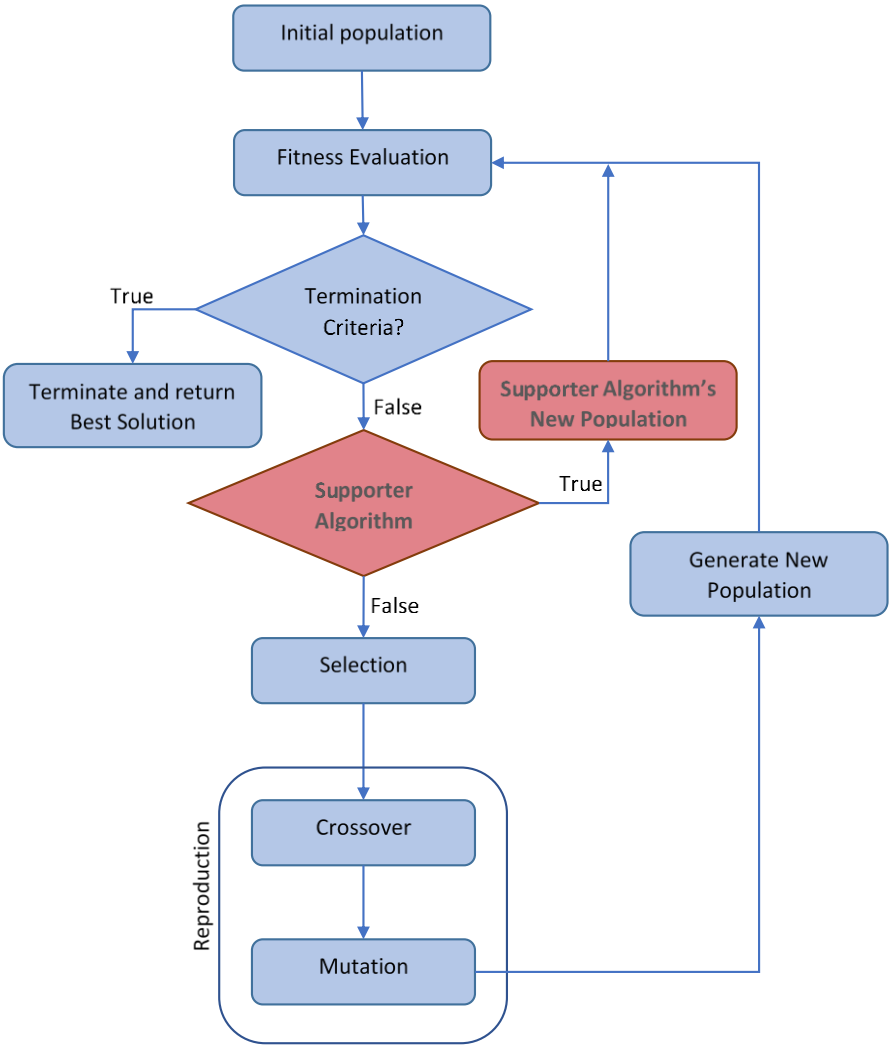


Figure 9. Overall Boosted-GA Method for Server Selection

Step 2. Fitness Evaluation: The fitness value should be allocated to each chromosome and considered in each decision. For the GA algorithm, we use the same fitness function presented in equation (9).

Step 3. Selection: After the generation of the initial population, the stronger chromosomes should be selected. These strong chromosomes will produce the next generation. We choose these individuals using a roulette

selection operator, developed by Holland [81]. On this selection wheel, the size of the segment for each parent is proportional to its fitness. Obviously, the larger the fitness, i.e., the larger the segment size, the higher the selection's chance. During each successive generation, the roulette selection operator keeps both the utilization and the player's QoE for the next generation by preferentially selecting the best solutions with randomization. The probability P, for each individual chromosome i, is defined by:

$$P(Individual\ i) = \frac{f_i}{\sum_{j=1}^{PopSize} f_i} \tag{13}$$

Where f_i equals the fitness of individual i.

Step 4. Reproduction: This step is to produce a second-generation population of solutions from those selected through genetic operators: crossover and mutation.

a) *Crossover*: We adopt the one-point crossover method where a gene is a basic unit to form the next generation, and two parents produce two offsprings (i.e., new solutions) that inherit information from both. Our algorithm randomly picks the crossover point in the parents' chromosomes. Then, it swaps the ends of the parents, from the crossover point to the end of the chromosome, to generate two new solutions. The percentage of the crossover in our algorithm is set to 50%.

b) *Mutation*: This operator is used to alter the value of a gene in the selected chromosomes randomly. For our case, changing the value of a gene turns the combination of the server assignment in the chromosome and produces a new solution. This new offspring will be added to the new population. The mutation ratio is the parameter that specifies the number of mutation chromosomes. The percentage of mutation in our proposed method is set to 50%.

Step 5. Termination: The algorithm should stop when it reaches a plateau. The terminating condition of our algorithm is: after 100 iterations, there is no progress for the best fitness value in the last 50 iterations, or the algorithm stops at the 400th iteration. If the termination conditions have not been met, go to step 6.

Step 6. Check Supporter-algorithm criteria: If Supporter-algorithm's condition is satisfied, run the Supporter-algorithm, otherwise go to step 2.

Since the chromosome in GA and the particle in PSO have the same concept and structure in our problem, the same Supporter-algorithm described in section 3.4.2 is deployed in our proposed GA method. The performance of both proposed methods will be presented in the next section.

3.5.4 Computational Complexity

For both algorithms, the complexity of computing the initial population and the fitness for one individual is O(QMS) and O(MS), respectively. Where Q is the number of populations, M is the number of players, and S is the total number of eligible servers. The maximum value of S is N×L. The Supporter-algorithm's complexity is O(QMS). The computation complexity to produce the particles in the Boosted-PSO is O(QMS+QMS); therefore, the total complexity is O(QMS). Since we limit the number of iterations to 'I', the maximum complexity of the Boosted-PSO will be O(IQMS). For GA, the computational complexity of selecting Q pairs of parents is O(QlogQ). The complexity of producing a new generation is O(QMS). Since the complexity of the Supporter-algorithm is the same, the maximum complexity of Boosted-GA would be O(I(QMS + QlogQ)). For an equal number of DCs, RSs, and players, since our population's size of the Boosted-PSO is half of the Boosted-GA, its maximum complexity would be around half, too. Consequently, our Supporter-algorithm does not increase the maximum complexity of the GA and PSO. However, as the results in the next section show, it causes a noticeable improvement in the efficiency of the algorithms.

3.6 Experiments, Results and Evaluation

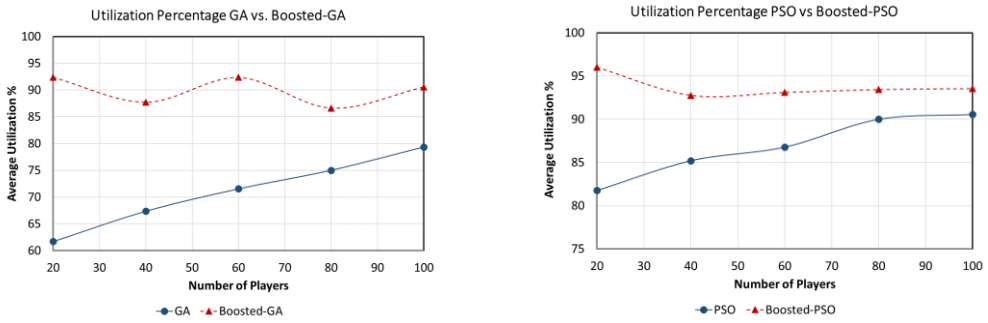
We evaluate the proposed methods at two levels. Firstly, we compare them with each other and with the server selection method introduced in [36], using the same setting and frame rate set as mentioned there, i.e., F1 = {15, 30, 45, 60}. Secondly, we compare them with four greedy algorithms called First Fit Algorithm (FFA), Best Fit Algorithm (BFA), Next-Fit Algorithms (NFA), and Worst-Fit Algorithm (WFA). These are common and efficient solutions to the bin-packing NP-Hard problems used in Cloud Gaming and the other similar contexts. For example, WFA was used by Onlive for game placement [82]. Also, the concept of the BFA, FFA, and NFA employed in other cloud gaming studies [49][21][47]. In this part of the experiments, we employ a new frame rate set, i.e., F2= {30,45,60,90,120}, which includes some higher and realistic frame rates for today's games. The simulations run on a Quade core Intel Core i7 3.2 GHz workstation with 12 GB of RAM.

3.6.1 Comparison with GA and PSO

To compare the proposed algorithms with each other and with [36], we created four groups of datasets with a different number of players and processing loads. We assumed that there are 10 eligible datacenters that can satisfy the player's maximum tolerable latency, and only 20 servers in each datacenter are GPU-equipped to be eligible for using as an RS. To allocate a range of quality to the players regarding the problem's constraints presented in 3.3.1, we consider four popular used frame rates [83] in this experiment, F = {15, 30, 45, 60}. Besides, a uniform distribution of the processing load is randomly assigned to the frame rates to satisfy equation (2) in the utilization model. Also, we assumed that all GPUs are working at their maximum frequency, which is considered 500,000 cycles per second. The simulation runs for different number of players, P= {20, 40, 60, 80, 100}. As our problem formulation presented in section 3.3 has two components, i.e., utilization and end user's experience, we consider both parameters to evaluate the efficiency

of the algorithms. We examine the capacity wastage and utilization percentage, as comparison criteria in terms of GPU utilization, and the QoE value as a metric of the end user's experience. The evaluation's results are presented in figure 10 to 11.

Figure 10 (a) and (b) show the progress of both Boosted-GA and Boosted-PSO algorithms respectively, compared to the methods presented in [36], in terms of utilization percentage. From Figure. 10(a), we can see that compared to our previous work, a considerable improvement happened for the GA-based method; e.g., around 30% improvement in utilization for 20 players, and more than 10% for 100 players. Similarly, the progress for the PSO-based method depicted in Figure. 10(b) is noticeable, especially for the small number of players. Another significant improvement is the consistency and steadiness of the utilization curve in the proposed methods versus the number of players, which makes these algorithms more reliable than our previous approach in [36].

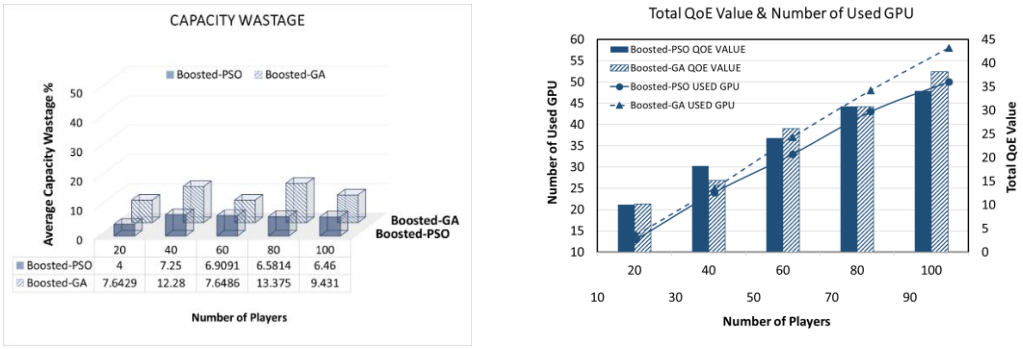


(a) **(b)**

Figure 10. Progress in terms of utilization percentage compared with results achieved in [32] for (a) Boosted-GA and (b) Boosted-PSO Methods

To compare the proposed methods with each other, we consider two parameters such as capacity wastage and total QoE in figures 11 and show how these two parameters vary for the different number of players. In Figure. 11a, we see that the Boosted-PSO has lower capacity wastage than the Boosted-GA. This figure shows the higher efficiency of Boosted-PSO with a

6.85% average capacity wastage compared to 11.09% for the Boosted-GA method.



	20	40	60	80	100
Boosted-PSO	4	7.25	6.9091	6.5814	6.46
Boosted-GA	7.6429	12.28	7.6486	13.375	9.431

(a) **(b)**

Figure 11. (a) Capacity wastage of the Boosted-PSO and Boosted-GA algorithms for different number of players. (b) The number of utilized GPUs by the Boosted-PSO and Boosted-GA algorithms and corresponding QoE values for different number of players

Another evaluation has been done based on QoE and the number of used GPUs that service providers should pay for. Fig. 11b depicts the achieved QoE values and the number of used GPUs after server assignment for the different number of players. The curves in this figure show that the Boosted-PSO always allocates a smaller number of servers. Moreover, the close value of its QoE to the Boosted-GA method is shown in Fig. 11b. Also, in some cases, e.g., 60 and 100 players, the QoE of the Boosted-GA is a little bit greater than PSO but at the expense of using several extra GPUs. For example, for 100 players, the Boosted-GA's QoE is higher by 4; however, it utilizes 8 more GPUs. For other cases, i.e., 20, 40, and 80, Boosted-PSO has the same or higher QoE while using a lower number of GPUs.

3.6.2 Comparison with Four Common Bin-Packing Algorithms

To evaluate the efficiency of these algorithms in terms of utilization, we compare them with four greedy algorithms that are well-known for solving bin-

packing problems, in cloud gaming and other contexts: FFA, BFA, NFA, and WFA. In the bin-packing problem, the goal is to pack multiple players' requested resources into the smallest number of servers. The requested resources are objects of varying size, and servers have a defined capacity. The main idea of the above four greedy algorithms are as follows:

The FFA places a newly requested resource in the first server with space to accommodate it. The NFA is similar to the FFA but sets a new request in the next server, from where it left off at the previous searching. The BFA places a new player into the server with the least capacity remaining (The most filled) and still has enough room to fit the player's request. The WFA allocates a new request in the least-filled existing server. Although these algorithms showed their high ability in resource allocation, they cannot switch between the qualities. Therefore, we compare our proposed algorithms with them in terms of utilization and number of used GPU, which are proven capabilities of these bin-packing algorithms.

To this end, we used the server selection's data of our proposed algorithms as inputs of the above greedy processes and compared their results with our proposed metaheuristic algorithms. Through these experiments, we can notice if there is a better assignment which is not considered by our algorithms and perceive how they are weaker or stronger than these four popular methods.

Figure. 12 to Figure. 15 compare the results of the Boosted-GA and Boosted-PSO algorithms with all of these greedy algorithms. We do these evaluations using the frame rates employed in [36]. The frame rate bounds in [36] are 15 to 60 fps. The 15 fps is used because it is a comparison baseline in some other studies, such as [28][83]. Today, however, 30 fps to 120 fps is more realistic. Therefore, the frame rate set used in our experiments in this section is F2 = {30,45,60,90,120}. To pass the constraint (b) of the problem statement for F2, the maximum frequency of the GPUs is set to 1.2GHz. The setting of this experiment includes the number of eligible datacenters and servers is the same as section 3.5.1.

Figure 12, and Figure 13 are the experimental results for F1, and Figure 14 and Figure 15 represent the evaluations for F2. Figure 13 shows that the Boosted-GA method's efficiency is comparable to BFA and FFA. However, in most of the cases, the Boosted-GA outperforms the BFA and FFA. Regarding Figure 14, the Boosted-GA shows better efficiency for the new frame rate set compared to the greedy algorithms.

The results in Figure 13 show that BFA is more efficient than FFA, NFA, and WFA. However, none of them can meet the Boosted-PSO's results. Similar behavior presented for F2 results in Figure 15. For all different number of players, records of the Boosted-PSO are stronger than the four other algorithms. In summary, the Boosted-PSO method always has a higher utilization percentage and a lower number of utilized GPUs. The NFA and WFA represent the lowest efficiency in these experiments. By increasing the number of players, the WFA faces noticeable efficiency degradation, which is compatible with the experiments' results conducted by Claypool et al. in [82]. This algorithm suffers from the over-provisioning of the resources.

The Boosted-PSO algorithm in MATLAB and on the test computer, which is mentioned in section 3.5, takes around 55 seconds to converge for 20 players with 200 servers. By increasing the number of players, this time will increase linearly. This processing time is based on the said processing power, which is remarkably less than a professional cloud server. Furthermore, the programing language and implementation techniques matter. A study on the speed of multiple programming languages in [84] shows that MATLAB is between 9 to 11 times slower than C++ for computing. Therefore, to have a practical sense of the convergence time, these points should be taken into account.

Overall, with these comparisons, the Boosted-PSO method shows its superiority to our previous methods and all Boosted-GA, BFA, FFA, NFA, and WFA in terms of Utilization and the number of utilized GPUs. Also, it has a

minimum fluctuation and shows a higher level of stability to variations of the number of players.

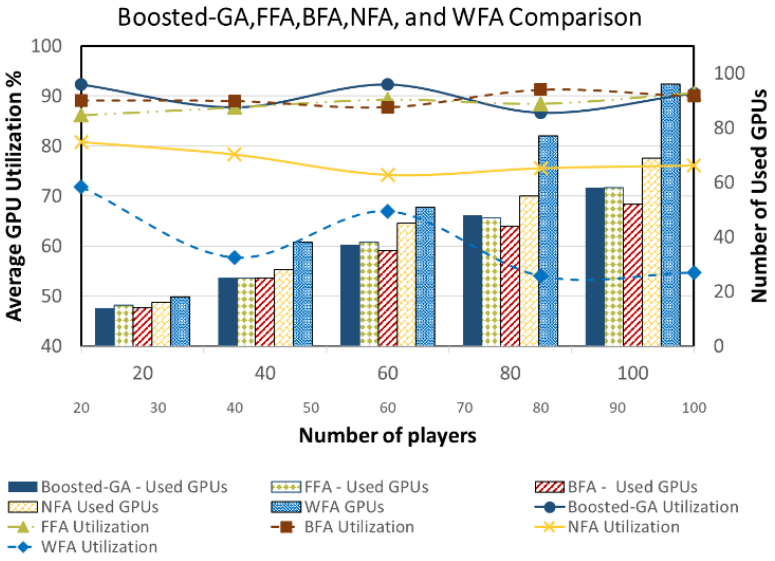


Figure 12. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for the different number of players. Using F1 frame set

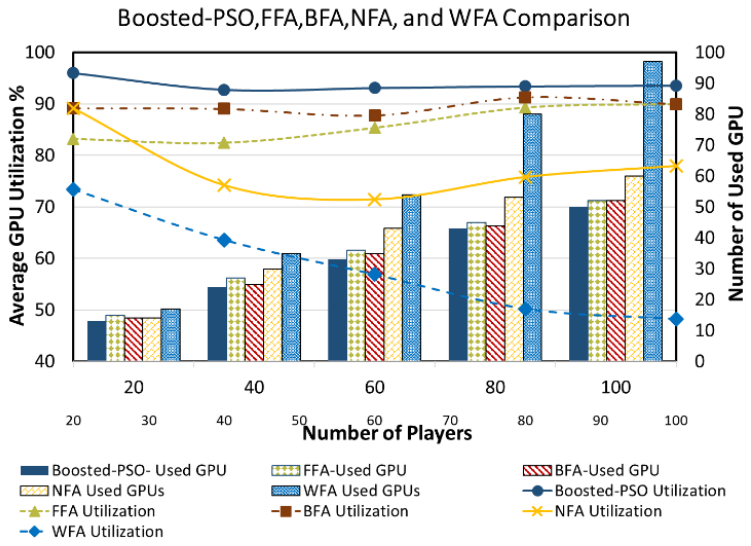


Figure 13. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for the different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F1 frame set

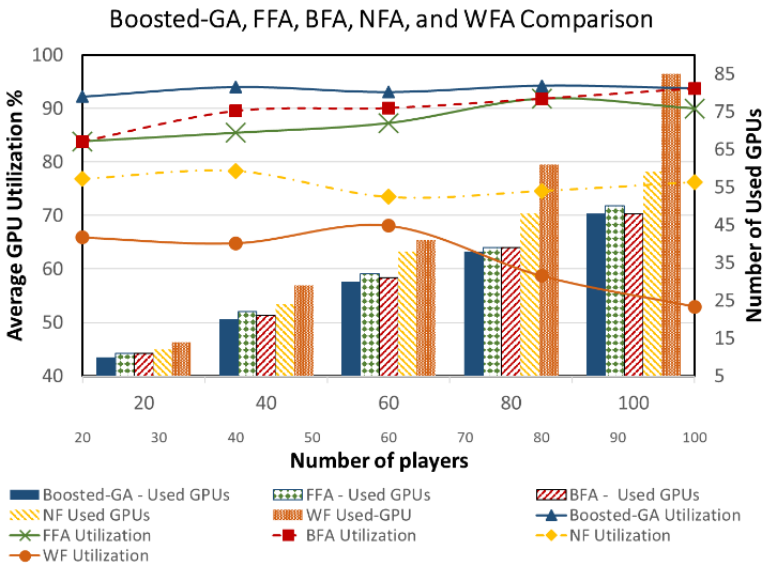


Figure 14. Comparison among Boosted-GA, FFA, BFA, NFA and WFA methods in terms of the Average GPU Utilization and the number of utilized GPUs, for the different number of players. Using F2 frame set

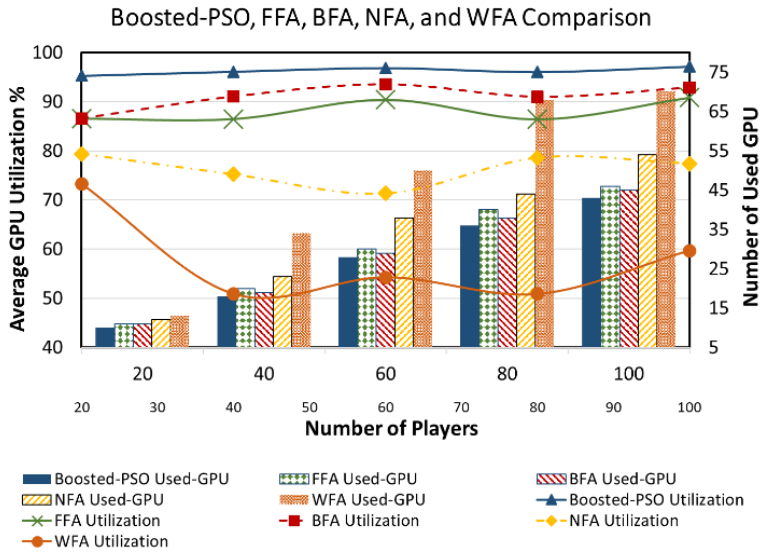


Figure 15. Comparison among Boosted-PSO, FFA, BFA, NFA and WFA methods for the different number of players in terms of the Average GPU Utilization and the number of utilized GPUs. Using F2 frame set

3.7 Summary

In CG, finding an appropriate resource allocation method to satisfy the service provider's profit and the end user's quality of experience simultaneously, is still a challenge. In this work, we address this challenge by

presenting a server selection method that considers the provider's and the client's concerns together. The presented objective maximizes GPU utilization while increasing the player's experience. To solve the problem, we proposed two metaheuristic algorithms, Boosted-PSO and Boosted-GA. The proposed methods considerably compensate for the lack of efficiency in our previous work, especially for the low number of players. We evaluated the boosted algorithms in two aspects, utilization and player's experience. Also, we compared the proposed algorithms to the GA and PSO methods and four popular bin-packing algorithms. The simulation results showed that our Boosted-PSO method achieves the highest efficiency among the other methods, including Boosted-GA, FFA, BFA, NFA, and WFA in terms of GPU utilization, capacity wastage, and player's QoE. Also, it has remarkable stability for the different number of players.

Chapter 4. QoE-Aware Fault Diagnosis in Packet Video Networks

4.1 Introduction

In order to maintain an acceptable QoE for a video streaming service, fault detection and isolation is necessary. Continuous network monitoring requires such a system; however, no single entity can access the whole end-to-end path in a multi-AS network, making fault management a challenging task. In this chapter, we developed two DL models for fault detection/isolation problems and showed that it is possible to use only client-side QoE metrics to detect and isolate faults with very good accuracy. The data collection procedure, its testbed, and the data-driven methodologies will be presented in the following sections.

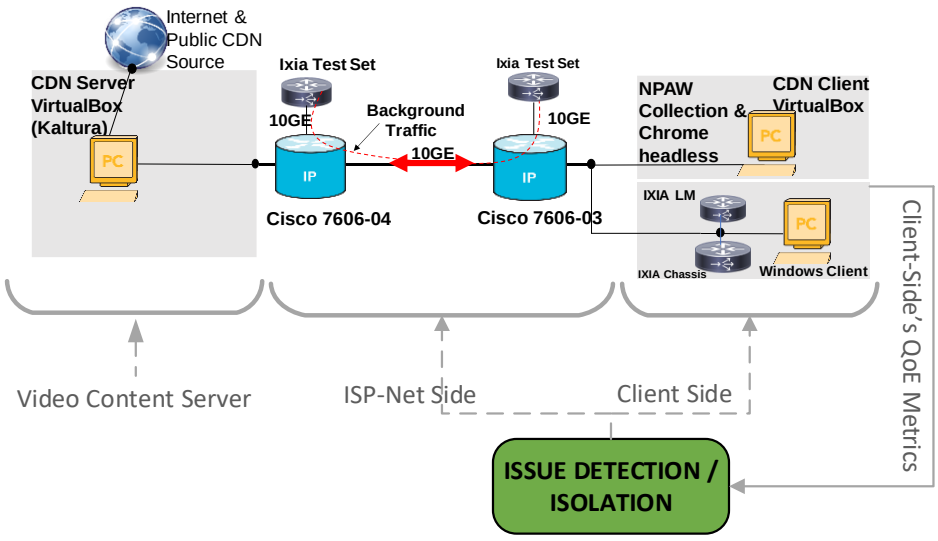


Figure 16. Data Collection Testbed and Fault Diagnosis Framework

4.2 Testbed and Data Collection Infrastructure

To build an appropriate dataset of video metrics that assists us for network diagnosis, we employed the testbed presented in Figure15. The path includes a simplified ISP with two Cisco7606 routers and Ixia background traffic generators, one video server, and a client network. We used Kaltura [85] as a video server with 20 SD and HD videos. We streamed multiple videos from Kaltura to the client and collected network QoS and client QoE metrics.

4.2.1 Tools

The tools we used to collect data include iXiA for generating background traffic, Nice People At Work (NPAW) for collecting QoE, Route Optimization and Assurance (ROA) for collecting QoS, and Chrome-headless for playing video in headless mode on the client. Each are described next.

1) iXiA [86] is a professional tool for generating background traffic. We used it to create congestion in the path. In our experiments, packet loss due to the said congestion varied between 0% to 20%.

2) NPAW [87] is a professional real-time media evaluation suite used for collecting real-time QoE metrics. It measures different types of events some of which, such as JOINTIME and BUFFER, contain QoE metrics. We will explain more about these metrics in section 4.1.2.

3) ROA is a professional network management suite for traffic monitoring, anomaly detection, and reporting. We employed ROA to monitor the network path, status of the generated anomaly, and collect network parameters such as latency, httprtt, jitter, and packet loss. Again, these QoS metrics are used only as ground-truth for validation, not as features in our system.

4) Chrome headless [88]. Displaying multiple videos at the same time on the screen is very distractive, due to the pop-up window of multiple players. To avoid this, we wrote a python script to automatically run multiple videos on

the client in a headless mode; i.e., without popping the player up. In this script, we connect with the browser using a browser automation framework known as Selenium.

To automatically collect data, we wrote a data collection program in python. When the videos play on the client, this program can extract and simultaneously record both the QoE and QoS metrics coming from NPAW and ROA, respectively. In every time window Δt, it saves these metrics in a CSV formatted file.

4.2.2 QoE Measurement

QoE's evaluation can be complicated due to its dependency on the end user's subjective perception and expectation. In a real-time networked system such as video streaming, subjective QoE assessment methods such as ITU-R B-500 [89] or ITU-T P.917 [90], which measure the Mean Opinion Score (MOS) offline, are not practical. Therefore, approaches such as the Video Quality Metric (VQM) [91] and others have been proposed to predict the MOS objectively. In this study, since we had access to the industry-grade NPAW suite, we used that to measure real-time QoE. A video service provider can embed in its video player a similar tool or any other tool that measures the QoE locally. We implemented a real-time process to measure the QoE in the time windows Δt from the beginning of a video's playback, as shown in Figure 17, employing real-time parameters that come from the NPAW events JOINTTIME, BUFFER and STOP. These events are in JSON format. At each Δt and stop time, we parse these events to extract playback duration, join time buffering, buffering length, buffering frequency, and average bitrate that are the metrics used for our QoE calculation with the following equations [92][3]:

$$Q_1 = \exp\left(-\sum_{i=1}^{k} \frac{N_i * L_i * W_i}{T_i}\right) \quad (1)$$

Where:

Ni is the number of buffering event in time-window i;

Li is the average buffering lengths in time-window i;

Wi is a weight factor

Ti is the time period in seconds of each time-window i;

k is the number of time windows of a video

Q_1 gives us a value between 0 and 1.

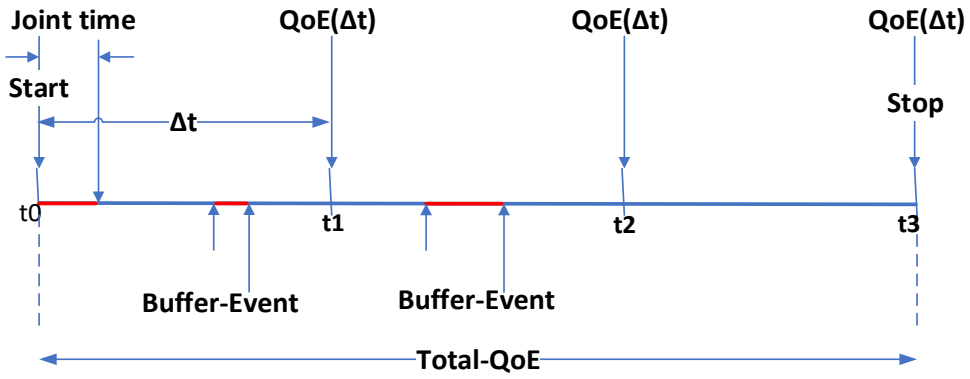


Figure 17. Steps of the QoE Calculation during the video playback

To calculate the maximum possible video quality for a given average bitrate, we used the following equation [93]:

$$Q_2 = v_1 - \frac{v_1}{\left(\frac{B_{rv}}{v_2}\right)^{v_3}} \qquad (2)$$

Where Q_2 presents the maximum video quality, between 0 and 4, at a given video bitrate (BrV), and coefficients v1, v2, and v3 are dependent on the codec type, video format, key frame interval, and video display size. The final QoE value is then calculated as follow:

$$Q = 1 + C_1 Q_2 Q_1 \qquad (3)$$

Where C_1 is a scaling coefficient. But, sometimes in real-time data, the value of the bitrate is not available. For these cases, we measured the QoE using equation 4:

$$Q_{total} = \begin{cases} 1 + C_1 Q_2 Q_1 & If\ bitrate\ is\ available \\ C_2 Q_1 & If\ bitrate\ is\ not\ available \end{cases} \quad (4)$$

Where C_2 is another scaling coefficient. Q_{total} is the QoE value of each sample in a given time window.

4.2.3 The Dataset

Using our infrastructure shown in Figure15 and the procedures described above, we streamed from 20 to 40 videos simultaneously over the 10 Gbps link for normal operation, and hundreds of videos simultaneously to create congestion. In total, we recorded the QoE statistics of 2355 video streams leading to more than 17000 samples. From each real-time event on the client-side, we extracted six metrics: playback duration, buffering length, buffering frequency, average bitrate, happiness score (QoE given by NPAW at the stop time), and the calculated Q_{total}. The dataset has three labels: no issues, ISP issue, and client issue. Figure 18 shows a sample of the six features which are collected in 7 Δt time-steps during a sample video playback. Each video has a unique token.

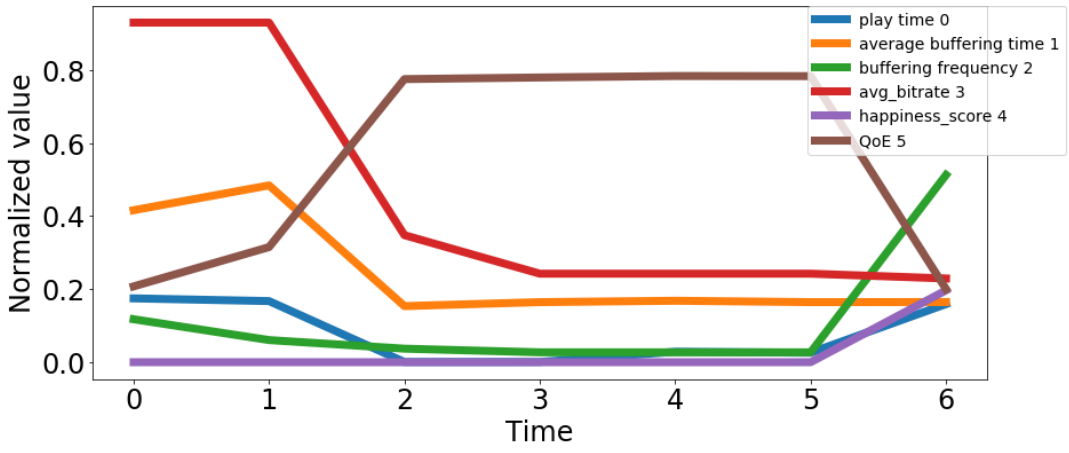


Figure 18. Normalized Value of 6 Features at 7 time-steps (0 to 6) during a single video playback

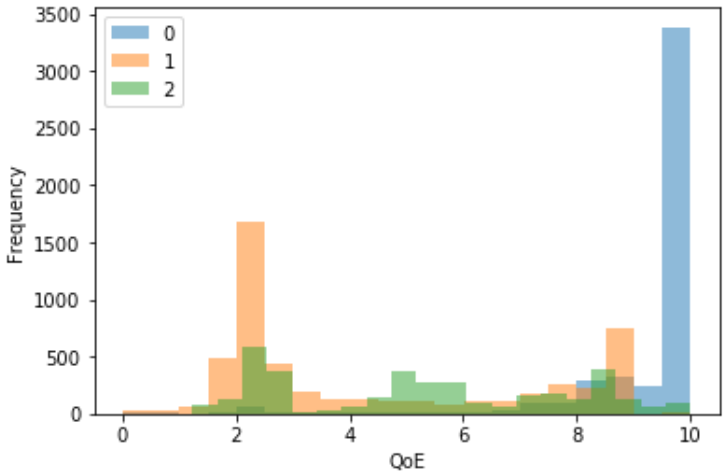


Figure 19. QoE Hysteresis for the three classes of the data. The classes are 0, 1, and 2 for no issues, ISP issues, and client issues.

Hysteresis of the QoE for the three classes is presented in Figure 19, which shows the overlap of the three classes with each other. Classes 0, 1, and 2 are Congestion_Free, Net_Congestion, and Client_Issue, respectively, which refer to video samples with no issues, ISP issue, and client issue, respectively. As expected, classes 1 and 2 have noticeable overlaps in QoE, because the user doesn't care about the location of the fault. What the user faces in both class 1 and 2 is the stalling of the video.

4.3 System Design and Training

Deep learning methods such as MLP and Recurrent Neural Networks (RNN) have been shown to yield excellent results on challenging activity recognition tasks with light or no feature engineering[94][39][40]. MLP is a classical ANN method that is efficient in classification and prediction problems, where a class should be assigned to the input data. LSTM is currently the most successful RNN method used mostly in Natural Language Processing, where we typically face sequences of time series data. According to the data evaluation in section 4.1.3, our problem is essentially a sequence classification problem. Therefore, in this study, we chose MLP and LSTM to train our system. We chose both because we wanted to compare their performance. In our approach, problem detection and problem localization are

performed simultaneously. For instance, if the classifier assigns a single test sample to class 0, it implies that no issues are detected. If it allocates the test sample to classes 1 or 2, it means an issue detected; furthermore, class 1 means the issue is located at the ISP-side, while class 2 means it's located at the client-side. Next, we describe the MLP and LSTM models we built.

4.3.1 MLP

A neural network has a chain-based structure that includes multiple layers. Each layer includes a group of units or neurons, and is a function of the preceding layer [95]. For the input layer, $h^{(1)}$, that function, f(), is:

$$h^{(1)} = f^{(1)}(W^{(1)T}x + b^{(1)}) \quad (5)$$

Where x is the input data, W is the weight matrix, and b is the bias. The second layer is:

$$h^{(2)} = f^{(2)}(W^{(2)T}h^{(1)} + b^{(2)}) \quad (6)$$

and so on. In DL, an essential consideration is how deep and with what width the structure should be. This is inferred from practical experiments and monitoring the validation set's error. In our MLP model, the best outcome was achieved with one input layer followed by three hidden layers and one output layer, as shown in Figure 20. Also, we employed batch normalization (4.3.1.1) and kept the drop-out value, as a regularization technique, equal to 0.5. We set the batch size for training to 128, and the adopted optimizer was Adam [95]. The activation function at the output layer was a Softmax function, and at the layers before was a Rectified Linear Units (ReLU) function. Equations (7) and (8) are the mathematical expressions of these activation functions, respectively.

$$P_i = Softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad (7)$$

$$ReLU(x) = \max(0, x) \quad (8)$$

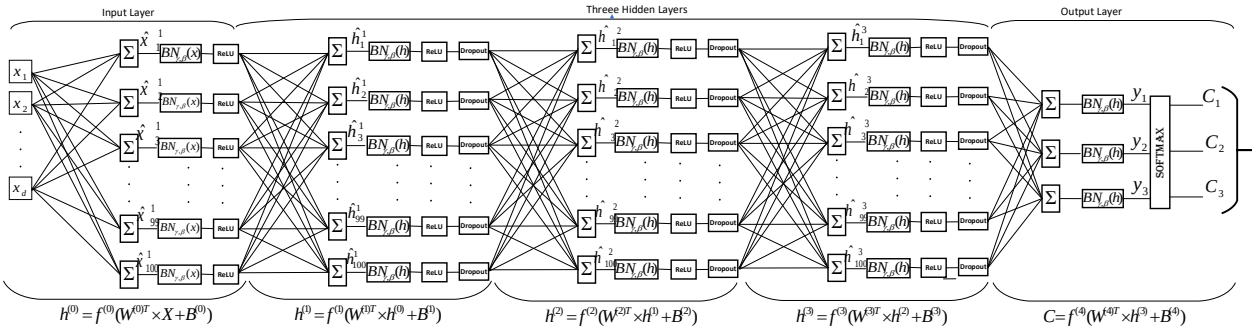


Figure 20. The MLP model

4.3.1.1 Batch Normalization:

During the training of a deep neural network, the distribution of each layer's input changes due to the changes of the previous layers' parameters. This phenomenon, called internal covariate shift, causes a slowdown of training by requiring a lower learning rate.

Batch normalization (BN) is a regularization technique that can be used in any deep learning layer. It is used to reduce internal covariate shift, improve training efficiency, and enhance the network's generalization by normalizing each layer input. The operation performing over a BN layer is as follows:

$$\mu_B = \frac{1}{m}\sum_{k=1}^{m} x_k \tag{9}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{k=1}^{m}(x_k - \mu_B)^2 \tag{10}$$

$$\hat{x}_k = \frac{x_k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{11}$$

$$y_k = \gamma \hat{x}_k + \beta \tag{12}$$

Where x_k is k-th input of the BN layer over a mini-batch: B = $\{x_{1\ldots m}\}$, γ *and* β are scale and shift parameters that should be learned and the y_k represents the output (transformed) of this layer.

4.3.2 LSTM

An LSTM network can learn and memorize during a long sequence of input data. The model can support, in parallel, multiple channels of data such as client QoE metrics. A single cell of a typical LSTM network is depicted in Figure 21. It has three gates: input gate, forget gate, and output gate, labeled in the figure as i_t it, f_t, and O_t, respectively. They determine the share of the current, previous, and output representations in the current timestamp. Cells are connected recurrently to each other. LSTM can add or remove information using this gate structure. The gates employ sigmoid activation σ, which keeps the gating value between 0 and 1. Zero means all information should be thrown away, and 1 means all information can be transferred. Each cell should generate a new cell sate s_t and a new output h_t, based on the current input vector x_t, previous hidden state h_{t-1}, and previous cell state s_{t-1} . To this end, the forget gate f_t, decides which part of the information from the previous cell state s_{t-1} should be kept. The following six equations present the mathematical operations of LSTM.

$$f_t = \sigma\left(b_f + U_f x_t + W_f h_{t-1}\right) \tag{13}$$

Equations (14) and (15) apply the same strategy to update the input information and hidden state, respectively, while equation (16) gives the actual value of the current cell state.

$$i_t = \sigma(b_i + U_i x_t + W_i h_{t-1}) \tag{14}$$

$$g_t = tanh(b_g + U_g x_t + W_g h_{t-1}) \tag{15}$$

$$s_t = f_t s_{t-1} + g_t i_t \tag{16}$$

The output gate o_t, as formulated in equation (17), controls the hidden state, and the current output is determined using equation (18).

$$o_t = \sigma(b_o + U_O x_t + W_O h_{t-1}) \quad (17)$$

$$h_t = tanh(s_t)\, o_t \quad (18)$$

Where b, U, and W are biases, input weights, and recurrent weights, respectively. Each gate has its own b, U and W.

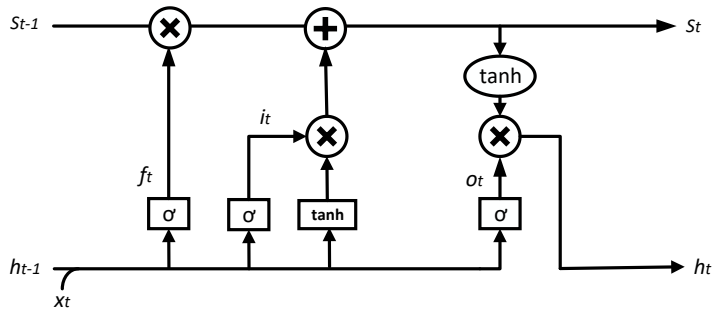


Figure 21. A single cell of an LSTM Network

In this study, we implemented an LSTM model with one input layer, four hidden layers, and a fully connected layer as an output. We kept the drop-out value equal to 0.5. We set the batch size for training to 128, and the adopted optimizer was Adam. We applied tanh for the activation function of the LSTM layer and Softmax for the fully connected layer. The model is shown in Figure 22.

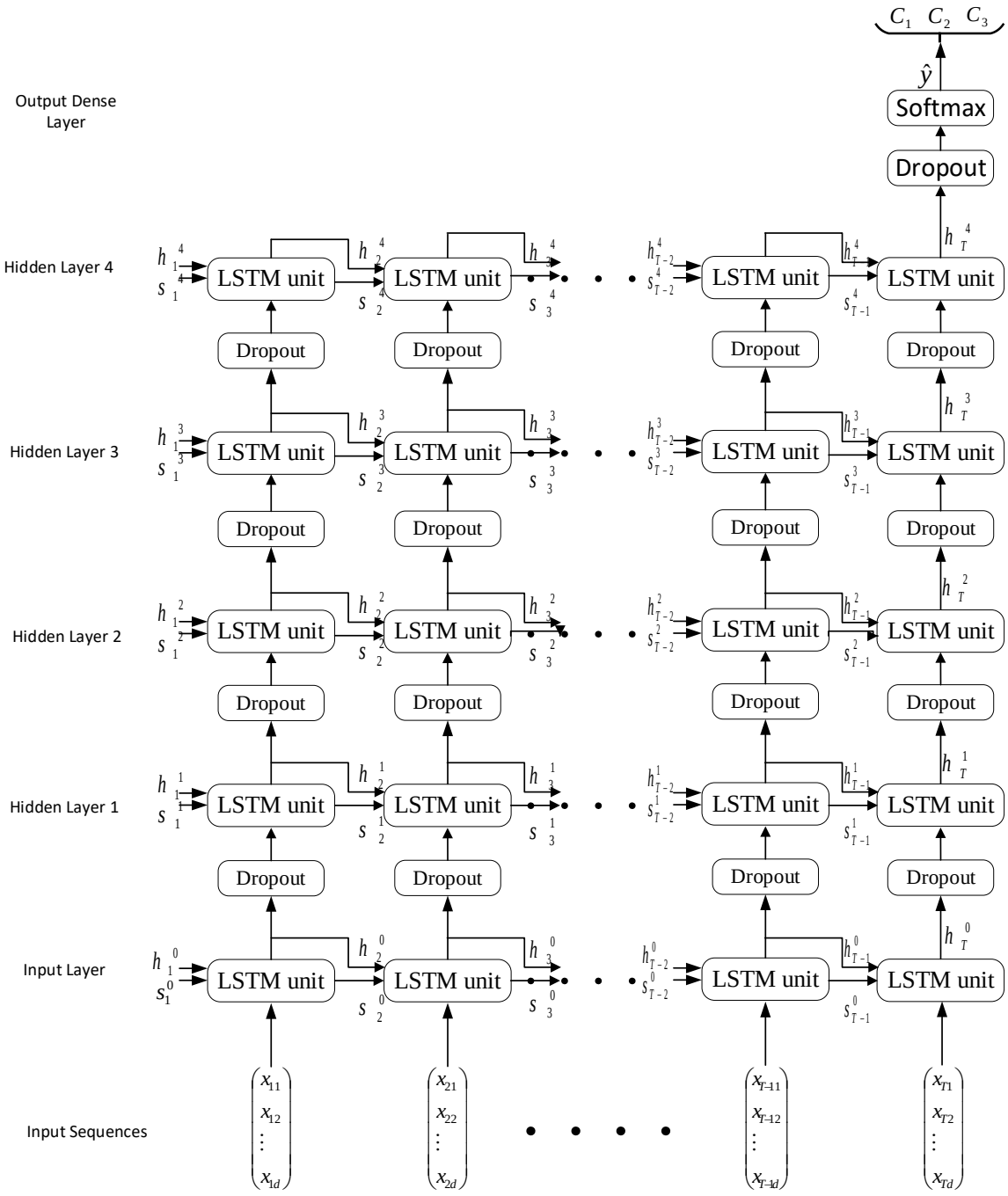


Figure 22. The LSTM model

We trained the MLP and LSTM models with the dataset described in 3.3, divided into two parts: 90% for training/validation and 10% for testing. Figure 23 shows the training/validation data's accuracy, indicating that both DL models are trained well, without considerable overfit.

4.4 Experiments and Performance Evaluations

The experiments have two parts. In the first part, we evaluated the proposed models' performance for classification among congestion-free, ISP-issue, and client-side issues. In the second part, we assess the catastrophic situation of congestion on both ISP and client sides. We conclude with a discussion.

4.4.1 Experiments

We evaluated the performance of the models and compared them with two other well-known algorithms: SVM and DT, with regards to accuracy, precision, recall, and F1-score. We used SVM as a classic classifier and DT as a method that used successfully in other studies such as [57]. Similar to the [57], we use version 3.8.3 of WEKA [96] machine learning suite for evaluating the DT. J48 is the implementations of the popular C4.5 of DT algorithms. We also implemented the SVM classifier using Scikit-Learn[97] library.

Accuracy is the ratio of the number of correct predictions and divides true positives (TP) plus true negatives (TN) by the total number of predictions, formulated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (19)$$

Where FP and FN are false positives and false negatives, respectively. Precision is the number of TP instances divided by all positive instances, formulated as:

$$Precision = \frac{TP}{TP+FP} \quad (20)$$

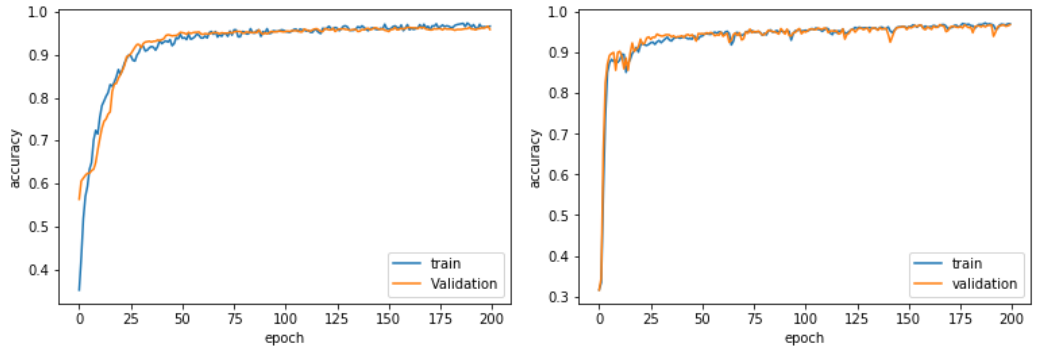
Recall is formulated as:

$$Recall = \frac{TP}{TP+FN} \quad (21)$$

which represents the number of true positives divided by all positive instances. The F1 score is the harmonic average of precision and recall, which is formulated as:

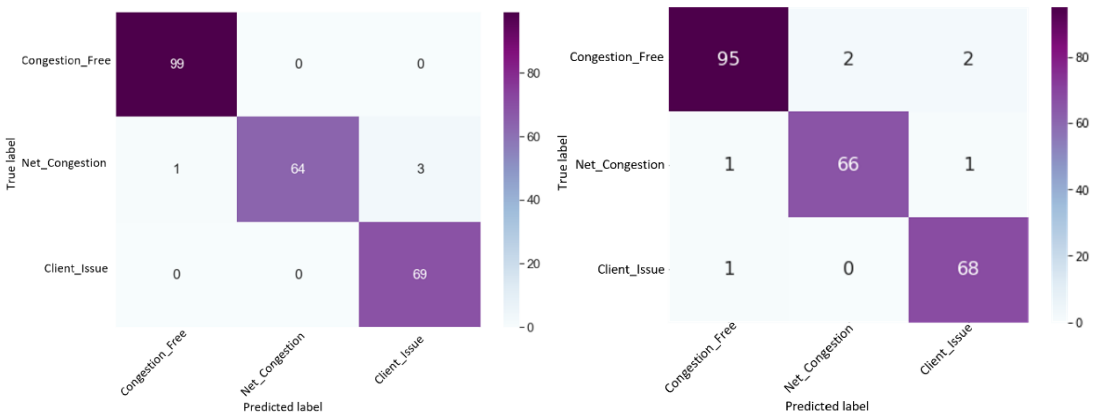
$$F1 - Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (22)$$

We employed 10-fold cross-validation. To evaluate how effective the used vector features were, i.e., the six metrics, we did the experiment twice: once with the six metrics, and once with Q_{total}, and then we compared the performance of the two.



(a) **(b)**

Figure 23. Accuracy of training/validation for the six QoE metrics using (a) MLP (b) LSTM



(a) **(b)**

Figure 24. (a) Confusion Matrix for MLP Classifier. (b) Confusion Matrix for LSTM Classifier

Table 3 presents the comparison results for six metrics, showing that MLP, followed by LSTM, outperform DT and SVM. We repeated the same experiment but this time using only Q_{total} from (4). Table 4 presents the results, showing again that MLP, followed by LSTM, had the best performance. However, compared to the results in Table 3, the algorithms encountered a noticeable performance degradation of between 3.75% and 11%. This showed that using all six features had a positive effect on performance. Figure 24 (a) and (b) show the confusion matrices of the proposed MLP and LSTM models, respectively, for the test dataset, which included 236 tokens. All samples from classes 0 and 2 were classified correctly by MLP, while only 6% from class 1 were classified mistakenly. Similar results were attained by LSTM, with a couple more misclassified samples from classes 0 and 2.

Table 3. Performance using the six quality metrics

Algorithms /Criteria	Accuracy	Precision	Recall	F1-Score
MLP	97.1%	97.0%	97.2%	97.1%
LSTM	96.6%	96.7%	96.8%	96.7%
DT	90.9%	91.0%	90.9%	90.8%
SVM	89.7%	89.9%	88.9%	88.8%

Table 4. Performance using calculated QoE only

Algorithm/ Criteria	Accuracy	Precision	Recall	F1-score
MLP	93.6%	93.3%	93.8%	93.4%
LSTM	92.7%	92.6%	92.3%	92.4%
DT	84.6%	85.3%	84.6%	84.3%
SVM	78.6%	78.9%	76.8%	76.3%

4.4.2 Experiments on Catastrophic Congestion

Catastrophic failure happens when both ISP and client-side networks are experiencing congestion. This is a severe issue that causes abortion of video playback. Because we collected our data from the client side, the resulting data for catastrophic failure was very close to class 1 and 2 (i.e. client_issue and ISP_issue), and we can still detect them one by one. Since our solution is a classification, it gives a percentage of matching to a class, so, if there is catastrophic failure, then our system will first pick the highest percentage match (which could be class 1 or 2). After the operator fixes that, then our system will give the second-highest percentage match, if by that time the problem still exists. For example, assume that P = [0.5, 0.41, 0.09] is the matching probability of the three classes during a catastrophic failure, and the first two elements in the list are the probability of congestion at the ISP and client-net, respectively. In this case, the video management system first contacts the ISP admin as the highest critical entity, and then the client network as the second highest. This process is one approach that could be efficient. A second approach could be a new training model that covers catastrophic failure as a new class. This approach requires a dataset that includes this new class. To evaluate this approach, we tried to import the catastrophic class to our dataset, retrain the new classes' models, and evaluate the decision process. Our collected data had 89 video samples showing the catastrophic situation that were not used in our study until now. These videos were aborted due to severe congestion in the end-to-end path during playback. To make them more realistic, we added a 10% variance to the playtime, buffering length, buffering frequency, and happiness score in each time step of this data. Since, in our dataset, the reason for the issue on the client-side is client-network's bandwidth restrictions, we kept the average bitrate's distribution the same as client-issue's class. Then, we recalculated the total QoE using the new altered metrics and varied the value in each time slot of the data. We considered this dataset the fourth class, i.e., client+ISP_issue, and

we added it to the previous dataset. The number of available streaming videos in each class and the corresponding percentages are presented in Table 5. Fig. 25 shows the distribution of the four classes in the new dataset.

Table 5. Number/percentage of the streamed videos in the new dataset.

	Class 0	Class 1	Class 2	Class 3	Total
Number	990	678	687	89	2444
Percentage	40.507%	27.741%	28.110%	3.642%	100%

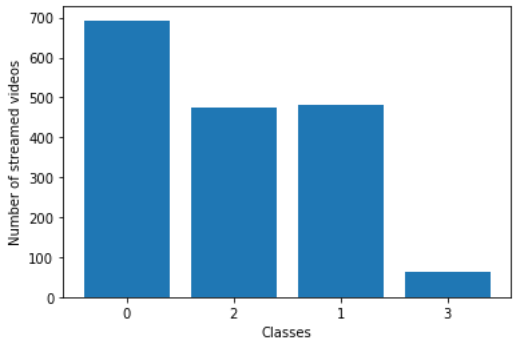


Figure 25. Classes' distribution for whole (new) data set

We changed the training and testing data ratio due to the limited samples in the fourth class. We divided the dataset into 70 and 30 percent for training/validation and testing, respectively. By trying various hyperparameters, we realized that changing the learning rate from 0.001 to 0.01 could give the best result for training the new dataset. Therefore, the only change in the models was the learning rate. We trained our MLP and LSTM classification models for the new dataset. Table 6 shows the performance of the classifiers for the test data.

As the results show, the classifiers were working quite well in terms of accuracy. However, sensitivity had degraded, especially in the MLP network. The imbalance in data could be one reason for the degraded sensitivity of a classifier [98]. Indeed, as shown in Fig. 10, the new dataset had a slight class

imbalance. We tried to evaluate this problem.

Table 6. Classifiers' performance on the imbalanced test data.

	Imbalanced Test data			
	Accuracy	Precision	Recall	F1-Score
MLP	93.72%	91.06%	84.50%	86.85%
LSTM	94.96%	92.31%	90.27%	91.22%

Depending on the skew ratio in the dataset, the classification problem could be severe or slightly imbalanced. Different strategies can be employed to address the issue, depending on the problem's data and complexity. One approach can be at the data level, e.g. collecting more data for the minor class, changing performance metrics, resampling the dataset, generating the synthetic samples, using, for example, the synthetic minority oversampling technique (SMOTE) [99], Borderline SMOTE [100], or adaptive synthetic sampling (ADASYN) [101]. Another approach could be at the algorithm level, e.g. one-class learning, cost-sensitive learning, ensemble or hybrid methods [102].

Because our data was not severely imbalanced, we tried a data level approach, i.e. resampling. As the classification's accuracy was not enough to evaluate the imbalanced data [98], we used three other performance metrics: precision, recall, and F1-score, which revealed the performance degradation for the models. We did a combination of random oversampling and undersampling on the minority and majority classes. Oversampling would be a reasonable approach in our application. This can happen in real video streaming, where similar videos pass through the same congested link at the same time. In our actual dataset, we have samples that have analogous issue patterns with high correlation. We resampled the training dataset in four different patterns: (1) an equal number of video samples created by oversampling classes 1, 2, and 3 to be equal to class 0, (2) an equal number of video samples created by

oversampling classes 1, 2, and 3 and undersampling class 0, (3) oversampling class 3 to be equal to class 0, and (4) oversampling class 3 (minority) to be equal to class 1. Figure 26 shows the distribution of the classes in all four datasets. For the four datasets, Table 7 presents the number of used streamed videos in each class and the share of the class in the related dataset.

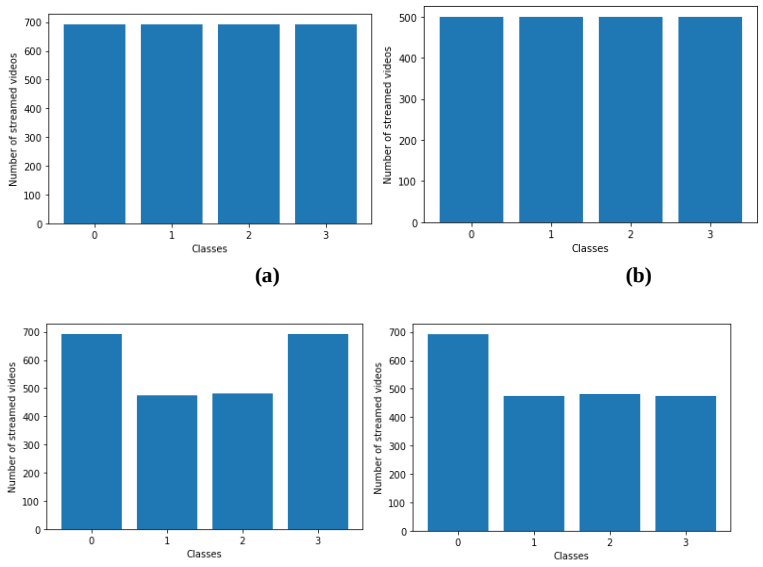


Figure 26. The four classes' distribution of the resampled training data sets. (a) an equal number of video samples created by oversampling classes 1, 2 and 3, to be equal to class 0 (majority). (b) An equal number of video samples were created by oversampling classes 1, 2, and 3 and undersampling class 0. (c) Oversampling the class 3 to be equal to class 0. (d) Oversampling the class 3 (minority) to be equal to class 1.

Table 7: Number/percentage of the streamed videos in the new training dataset (IMB: Imbalanced and RES: Resampled).

IMB/ RES	Num./ Perc.	Class 0	Class 1	Class 2	Class 3	Total
IMB	Num.	693	474	481	62	1710
	Perc.	40.526%	27.719%	28.129%	3.626%	100%
RES 01	Num.	693	693	693	693	2772
	Perc.	25.000%	25.000%	25.000%	25.000%	100%
RES 02	Num.	500	500	500	500	2000
	Perc.	25.000%	25.000%	25.000%	25.000%	100%
RES 03	Num.	693	474	481	693	2341
	Perc.	29.603%	20.248%	20.547%	29.603%	100%
RES 04	Num.	693	474	481	474	2122
	Perc.	32.658%	22.337%	22.667%	22.337%	100%

We divided the datasets into 70% and 30% for training/validation and testing, respectively, and trained the four MLP and LSTM models using the four new datasets. Tables 8 and 9 show the results for the MLP and LSTM models, respectively. As shown, resampling had noticeable improvement on the performance of both algorithms. Even the LSTM network, which had better results with imbalanced data, showed a significant improvement, especially for the resampled_01 dataset. The same dataset followed by the fourth dataset also gave the best results in the MLP classifier. As we expected, the proposed classifiers were impacted by the imbalanced dataset, but it was a slight imbalance problem, and by applying different resampling patterns to the training data, we addressed the problem properly.

Table 8: Evaluation of the MLP classifier for the imbalanced and the resampled data sets.

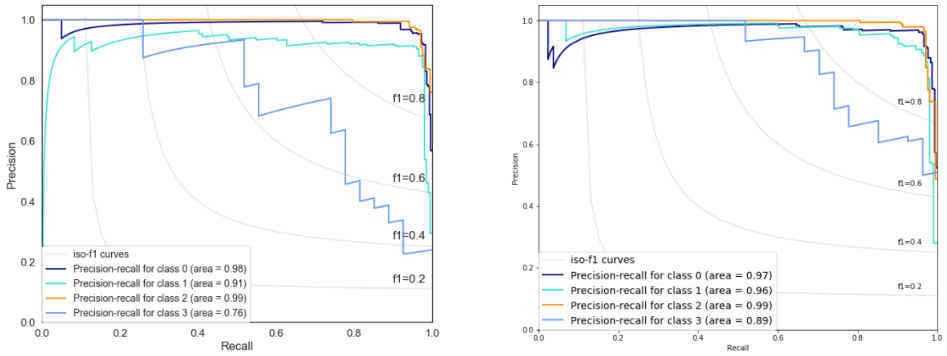
MLP Classifier	Test			
Imbalanced/ Resampled	Accuracy	Precision	Recall	F1-Score
Imbalanced	93.72%	91.06%	84.50%	86.85%
Resampled 01	96.32%	95.50%	96.28%	95.88%
Resampled 02	94.82%	90.29%	94.20%	91.97%
Resampled 03	95.63%	93.10%	95.59%	94.23%
Resampled 04	96.45%	95.01%	96.32%	95.64%

Table 9: Evaluation of the LSTM classifier for the imbalanced and the resampled data

LSTM Classifier	Test			
Imbalanced/ Resampled	Accuracy	Precision	Recall	F1-Score
Imbalanced	94.96%	92.31%	90.27%	91.22%
Resampled 01	95.50%	95.77%	96.51%	96.13%
Resampled 02	95.78%	94.26%	95.22%	94.72%
Resampled 03	95.64%	94.90%	95.78%	95.33%
Resampled 04	95.23%	94.72%	94.46%	94.59%

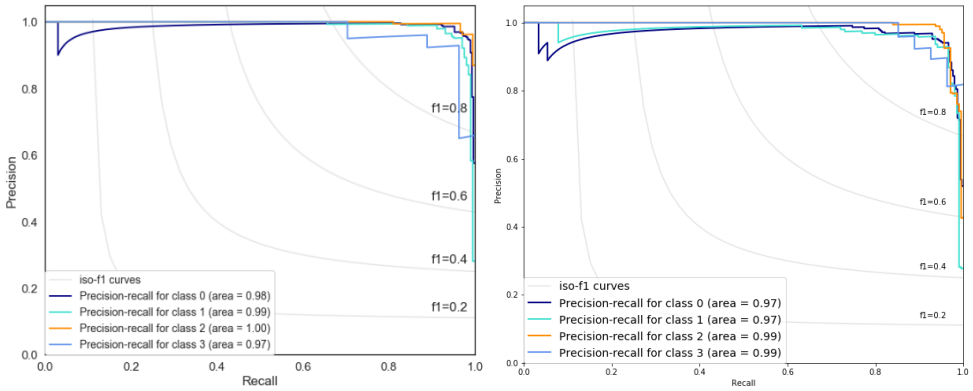
Figure 27 shows the Precision-Recall curves for the MLP and LSTM classifiers trained using the imbalanced dataset. The smaller area under the

curve for the fourth class shows the classifier's performance degradation for this class. Figure 28 represents the same curves for the MLP and LSTM classifiers, which were trained using the Resampled_01 dataset. The curve shows a noticeable improvement in the classifiers' performance when they are trained with the resampled data.



(a) (b)

Figure 27. Precision-recall curve. a) Imbalanced MLP b) Imbalanced LSTM



(a) (b)

Figure 28. Precision-Recall Curve after training with Resampled_01 dataset. a) MLP b) LSTM

4.1.1 Discussion

Collecting the time series data with the specifications described in section 4.1.3 requires the cooperation of multiple ASs and, to the best of our knowledge, no such dataset exists in the public domain, which is why we had to create our own simplified network and collect our own dataset. While the

network is simplified, we have to consider that even in complex networks that span multiple ASs, congestion in the path and its relevant delay cause buffering at the client's device, leading to QoE degradation. This has been captured in essence by our model. We can in fact assume that this congestion in our model represents that of a complex network, and the rest of the clients are receiving normal video content, indicated by the no congestion class. Of course, to add a new AS to the path, a new class of the data should be collected using the approach outlined in section 4.1.3, and the algorithms should be trained for the new dataset. Since we are using only the client-side QoE metrics, the ultimately collected data of the new class would be another highly overlapped version of the current classes, i.e. classes 1 and 2. Therefore, we will confront a multivariate Time-series class-overlapping problem again, similar to our simplified model. In short, despite its simplified model, our study shows the feasibility that with the help of AI, a VSP can indeed use only the QoE to localize the fault.

Our trained model does not have a remarkable overhead in computation and can work in real time. However, the reactive approach itself has an overhead, because an unsatisfactory experience with the video comes from an issue that has already occurred in the path. Forecasting the QoE can smoothen this issue and is a good direction for future work.

For the catastrophic condition evaluation, it should be noted that we did not need to employ synthetic data generation approaches such as SMOTE, as our resampling approach gave great results. Also, since in our data there is dependency not only between the features but also between time steps, the regular SMOTE family cannot handle these spatio-temporal dependencies to generate a meaningful sample.

4.4.3 Summary

To provide satisfactory QoE for video streaming, fault detection and isolation are necessary. However, in a multi-AS network, no single entity has

access to the whole end-to-end path, making fault management challenging. In this chapter, we developed two DL models for fault detection/isolation and showed that it is possible to use only client-side QoE metrics to detect and isolate faults with excellent accuracy. Although we only considered two fault localizations of ISP and client-side, our methodology can be expanded to add other ASs. Using the aborted video samples, we generated the fourth issue class, which is the catastrophic issue resulting from the ISP and client-side problem. We tried to handle the imbalanced class issue in the classification process, which was created due to a lack of samples in the fourth class.

Chapter 5. A BiLSTM-CNN Model for Multivariate Time-series QoE Metrics' Forecasting

5.1 Introduction

This section mostly endeavors to address the second shortcoming of the reactive network diagnosis approach using AI. Although our example and corresponding collected data related to the fault detection/localization problem, which is presented in Chapter 4, the proposed QoE forecasting can also be utilized by other video service management systems.

Historically, QoS such as packet loss, delay, and jitter were considered service satisfaction metrics. However, since QoS does not reflect a user's quality perception, QoE is now used instead. QoE describes the degree of enjoyment or irritation of the end-user while using a video service [103]. A significant amount of research has been conducted to understand, measure, and model QoE in different video services. This knowledge can help service and network providers deliver high-quality and cost-effective video services while efficiently managing network operations [104].

At its simplest, a video streaming chain includes the client side, server side, and the network between these two sides. To manage QoE in such a chain, there are two approaches: proactive and reactive as explained in chapter 2. In multi-AS networks, the end-to-end path is not always accessible for the VSPs. So, the reactive approach would be a common for these applications. However, as aforementioned, this approach suffers from lateness.

Smoothing this delay provides a noticeable improvement in different QoE-oriented video service control systems. Accurate forecasting of actual QoE metrics for the next time-step is a valuable component to smooth this shortcoming and improve management systems' performance. It assists service

management in proactively monitoring the system, having feedback from the client-side and reducing overall QoE degradation. It can be utilized in various management applications such as video network fault mitigation, rate adaptation in DASH, and resource allocation in wireless networks. This study, inspired by big data and deep learning capability, proposed an efficient method that prognosticates QoE metrics. Toward this goal, we used the collected data set presented in chapter 4. and defined a multivariate time series forecasting problem. We modelled a hybrid state of the art deep learning method, i.e., BiLSTM-CNN, to forecast the QoE metrics in advance, before appearing on the client's screen. We evaluated the proposed method's performance and compared it to three other well-known ML models, i.e., Support Vector Regression (SVR), Multi-Layer Perceptron (MLP), and Bidirectional LSTM (BiLSTM). The proposed model outperforms other models. Figure 29. also represents a sample use case of the proposed forecasting mechanism for a fault diagnosis system in simple multi-AS networks.

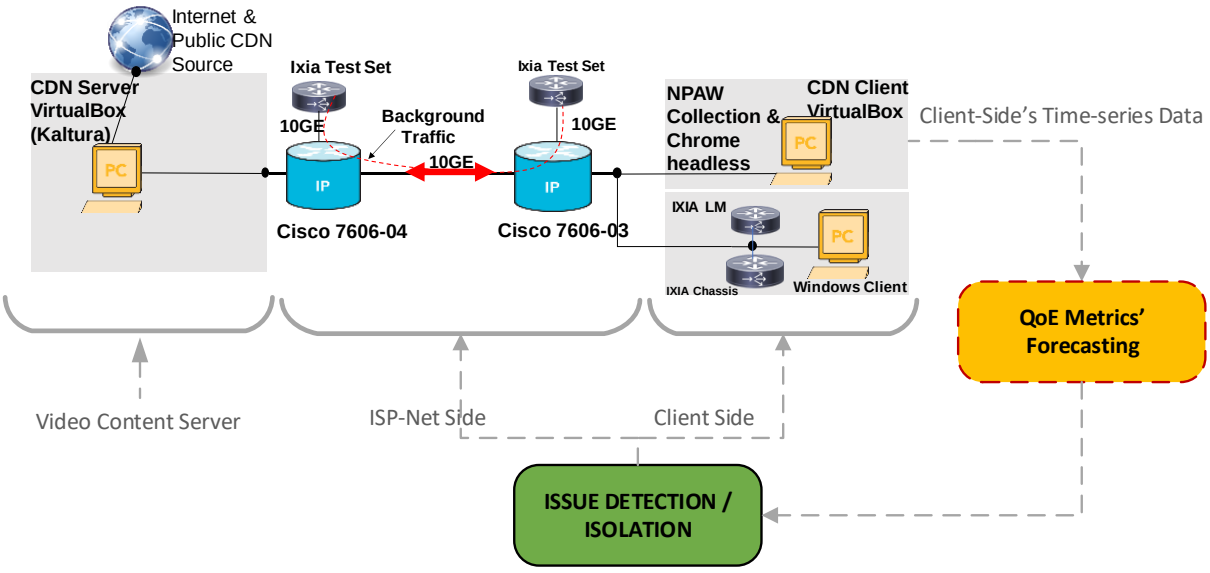


Figure 29. A sample use case of the proposed forecasting mechanism; Issue detection/isolation framework

5.2 Proposed Forecasting Model

Although in general QoE depends on more recent experiences, the *recency* is not enough if it is affected by the *primacy* effect, for example, if multiple buffering events occurred in the past [105]. The primacy effect is the human tendency to recall the events at the beginning of an events' series [106]. In other words, the early samples of the QoE have beneficial information for forecasting the QoE in the next time step. Therefore, our proposed algorithm should extract appropriate temporal features from the input sequences. Also, the QoE metrics have a nonlinear relation with each other because, for example, the user's experience always suffers from a buffering event, but the strength of this negative impact depends on the video bitrate level [107]. In modelling the forecasting process, these valuable dependencies should be taken into account. Therefore, due to the nature of our multivariate time series, we proposed a hybrid deep learning method called BiLSTM-CNN.

Over the past years, CNN has shown its strong capability to represent complex data, especially in image feature extraction [95]. LSTM has also become a proven solution for short and long-term dependencies identification with excellent results in Natural Language Processing, image captioning [108] and question answering [109]. The combination of CNN and LSTM has shown to help a system extract spatial and temporal features of the data efficiently [110][108]. BiLSTM has further improved LSTM in forecasting accuracy for time series data [111]. Therefore, we employed BiLSTM in our hybrid model to capture the temporal dependencies of the QoE metrics. To increase the model's performance and extract local features that summarize the QoE metrics' nonlinear dependencies, we employed a CNN layer after the BiLSTM layers in the proposed model. The BiLSTM layer accepts QoE metrics as input and extracts temporal features in order to feed into the CNN Layer for spatial feature extraction.

The input data includes the happiness score and Q_{total} that directly show the quality level perceived by the user, and the average rebuffering time, number of rebuffering, and average bitrate that impact the QoE. The video playback duration has been utilized as another input information since it is a useful metadata that shapes the final QoE score. We used the first six time-steps as input and the seventh time-step would be the expected result from the forecasting model.

5.2.1 BiLSTM

The LSTM model presented in chapter 4, have a causal structure where the hidden activation propagates information only in the forward direction through time. This approach may lose some useful information. However, BiLSTM provides learning in both forward and backward directions and allows the model to compute a representation that depends on both past and future [95]. By feeding the input sequence into the LSTM units, BiLSTM can precisely capture the underlying context and achieve higher performance than unidirectional LSTM [111]. Figure 30 describes a single hidden BiLSTM's layer, where h is the hidden activation for forward propagation, and g is the hidden activation for backward propagation. At each point t, in order to map input sequence x to output sequence y, the model combines the forward (from left to right) and backward (from right to left) propagation outputs. The output representation depends on the past and the future and can capture the complex temporal dependencies of our time series QoE metrics.

5.2.2 CNN

CNN uses the input data's spatial structure and connects the input patches to the single units in the subsequent hidden layer [95]. These connections are formed by merely sliding the patch window across the input data. This patchy operation is a convolution that extracts local features of the data by applying a set of weights for the filter. By changing the filter's weights,

we can change what the filter is activating and looking for. Different filters can activate different feature maps.

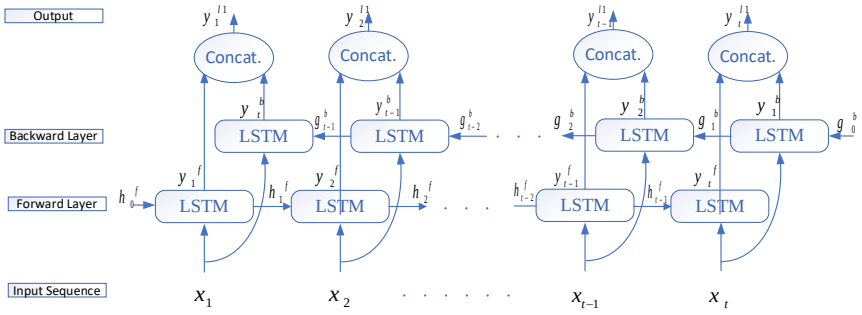


Figure 30: BiLSTM; Architecture of one hidden layer

A pooling layer is used to downsample the features extracted from each convolution layer and reduce the feature map size. It allows the network to deal with multiple scales of the input data. We employed the Max-pooling technique, which selects the most active feature of the feature patch chosen by the sliding window. Max-pooling helps make the system robust and the representation approximately invariant to the inputs' small deviations.

The next section describes the structure of the proposed hybrid neural network model.

5.2.3 BiLSTM-CNN

In this section, we introduce the proposed model designed to leverage advantages of BiLSTM for temporal feature extraction and CNN for capturing the complex dependencies among QoE metrics.

Let x^t be a vector of QoE metrics at time step t, and y^{t+1} and $\hat{y}^{t+1}$ be the actual and forecasted QoE feature vector at time instant t+1, respectively. In order to forecast the next QoE vector at time instant t+1, we have to take into account the following function:

$$\hat{y}^{t+1} = f(x^1, \dots, x^{t-2}, x^{t-1}, x^t) \quad (1)$$

Where $x^1, \dots, x^{t-2}, x^{t-1}, x^t$ is the QoE feature vector at each time step from the beginning of the video playback until time instant t. The nonlinear function $f(\cdot)$ is approximated by BiLSTM-CNN. The proposed model feeds the normalized input features into the BiLSTM layers to reduce temporal variations in input QoE metrics. Then, the last BiLSTM layer's output is fed into a CNN layer to reduce spatial variation. The CNN layer's output passes through the Max-pooling layer, then fed into a fully connected layer, transforming the features into a divisible space that makes the output more straightforward for final decision/forecast. Figure 32 shows the architecture of our proposed model.

a. Activation function

Applying nonlinear activation function allows the network to deal with nonlinear data and introduce complexity into the learning process [95]. Therefore, the network can solve more complex tasks. The default activation function for LSTM models is *tanh*. We used the scaled exponential linear units (*SELU*) activation function, which induces self-normalizing properties [110] and converges to zero mean and unit variance when propagated through multiple layers. This convergence property makes training very robust by avoiding vanishing and exploding gradient problems. It outperforms other regularization methods such as weight normalization and batch normalization. The following formulation shows the behavior of the SELU activation function [110]:

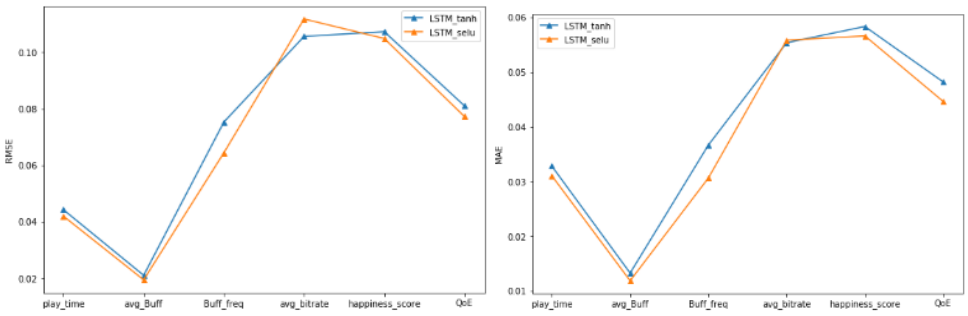
$$SELU(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha \exp(x) - \alpha & if\ x \leq 0 \end{cases} \quad (2)$$

Where $\alpha = 1.67733$ and $\lambda = 1.0507$, the same values as proposed in [110].

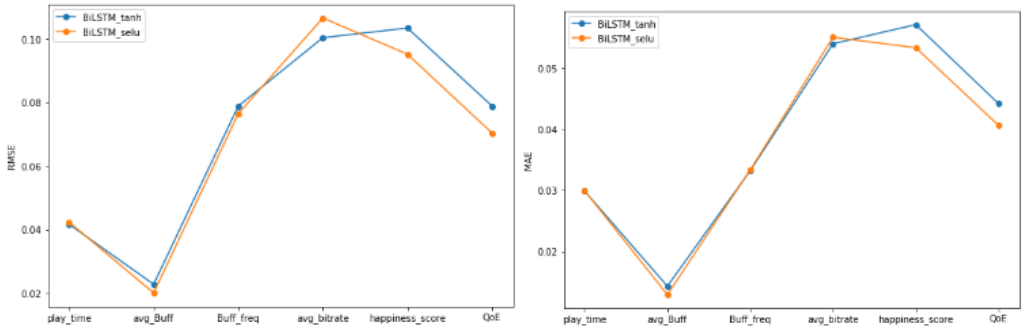
To evaluate the performance of SELU compared to tanh, we trained the LSTM, BiLSTM, and BiLSTM-CNN forecasting models with both activation functions. Table 10 represents the specifications of the test models.

Table 10: Specification of Activation Function's Test Models.

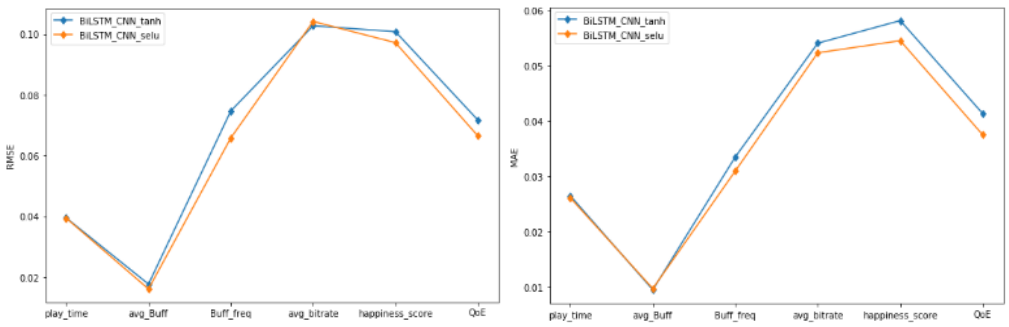
	LSTM	BiLSTM	BiLSTM-CNN
Batch Size	128	128	128
Training Epoch	400	400	400
Learning rate	0.001	0.001	0.001
Hidden Units	5	5	4 BiLSTM + 1 CNN



(a)



(b)



(c)

Figure 31. Performance comparison between SELU and tanh activation functions for LSTM, BiLSTM and BiLSTM-CNN Forecasting models in terms of RMSE and MAE. (a) LSTM, (b) BiLSTM and (c) BiLSTM-CNN.

The performance comparison has been made for all six QoE features. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are the performance criteria in this evaluation. The comparison results for all six forecasted features are presented in Figure 31. As depicted in this figure, for most QoE metrics, *the SELU* activation function converges to the higher performance.

b. Architecture

For For our proposed model, by testing the different number of layers, we experimentally determined that more than four BiLSTM layers and one hidden CNN layer do not show noticeable progress in the performance. We utilized drop out after the fully connected layer, which led to a very positive effect on minimizing the overfitting between the training and validation data. The final BiLSTM-CNN model was implemented with four BiLSTM layers, one 1D-Convolutional layer, a Maxpooling layer, a fully connected layer and an output dens layer. We kept the drop-out value to 0.5 after the fully connected layer. We set the batch size for training to 128, and the adopted optimizer was Nesterov-*Adam or Nadam* [112]. We applied *SELU* for the model's activation function, which is very effective by inducing self-normalizing properties.

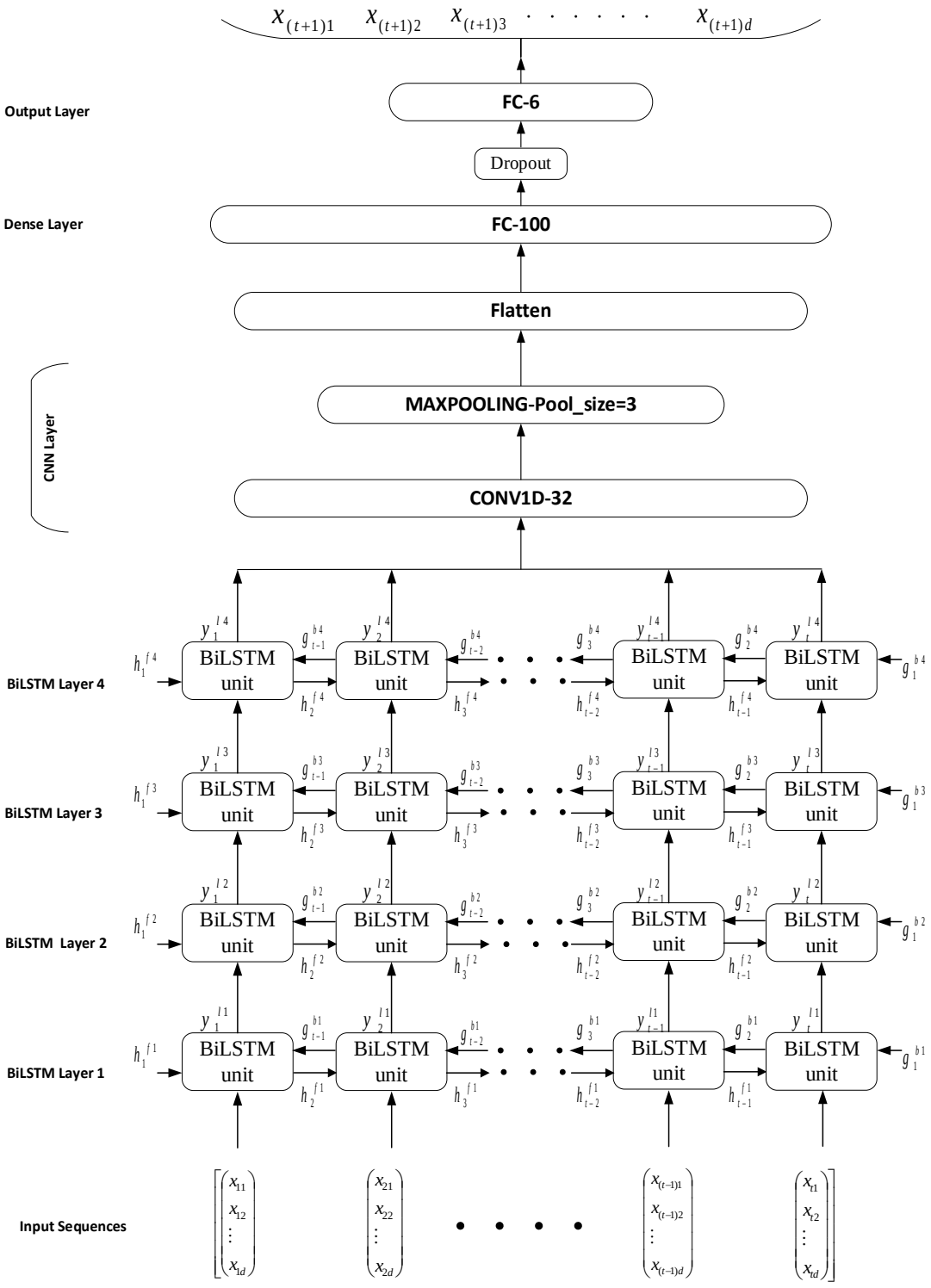


Figure 32: BiLSTM-CNN Architecture.

5.3 Evaluation and Results

To evaluate the proposed model's accuracy, we trained the model with the dataset described in chapter 4, divided into two parts: 80% for training, 10% for validation and 10% for testing. The hybrid BiLSTM-CNN model is compared with three anchor models: BiLSTM as a state-of-the-art time-series' forecasting model, MLP which is the most basic neural network model, and the SVR as the most classic time series forecasting model. The implementations of BiLSTM-CNN, MLP, LSTM and BiLSTM are based on the Keras library with the TensorFlow backend. Besides, we use Scikit-Learn library for implementing the SVR.

To evaluate the proposed methods' performance, we utilized MAE, a linear measure for error scoring, and also RMSE, a quadratic scoring criterion that shows how the residuals (forecasted error) spread out. MAE gives equal weight to the errors; however, RMSE penalizes the variance. In other words, errors with larger absolute value will be given higher weight by RMSE. These measures' formulation is as follows:

$$RMSE = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}} (\hat{y}_i - y_i)^2} \quad (3)$$

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |\hat{y}_i - y_i| \quad (4)$$

Where, y_i is the i^{th} instance or observation and $\hat{y}_i$ is its associated forecasted value. The smaller the RMSE and the MAE, the smaller the error.

For all four models, we compared the forecasting of all six input quality metrics in terms of MAE and RMSE. Figures 33 and 34 show the proposed BiLSTM-CNN model's evaluation results compared to the three anchor models. As

shown for all six features, the best performance belongs to BiLSTM-CNN. The average bitrate and the happiness score have the worst scores as expected due to unavailability in some samples. To visualize the forecasted features' accuracy, we plotted the actual and the predicted QoE for all four models in Figure 35. As we expected, from (*a)* to (*d)*, we can see the least to most correlated forecast, respectively, with our method performing the best.

Figure 33: Comparison of four forecasting ML Models in terms of RMSE for each metrics.

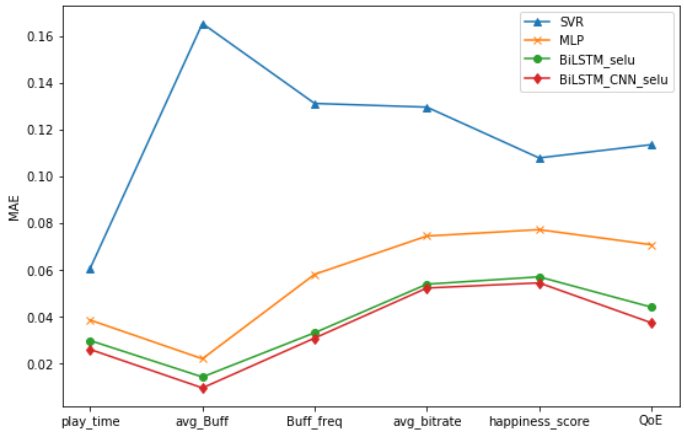
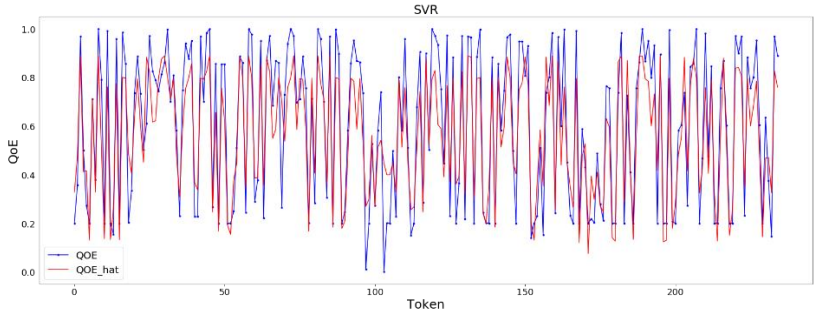
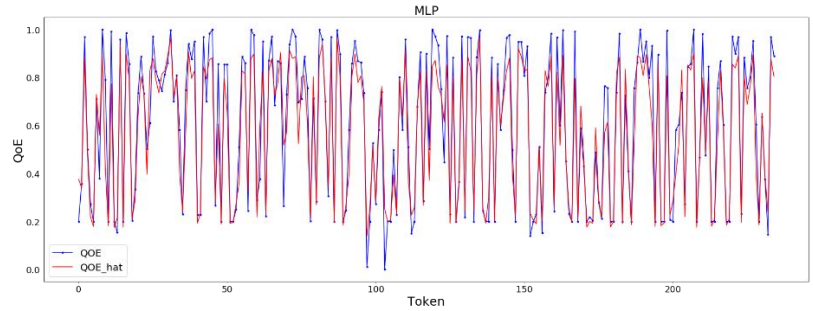


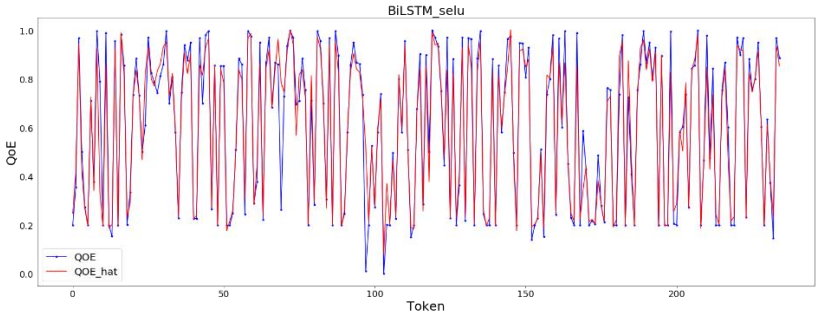
Figure 34: Comparison of four forecasting ML Models in terms of MAE for each metrics.



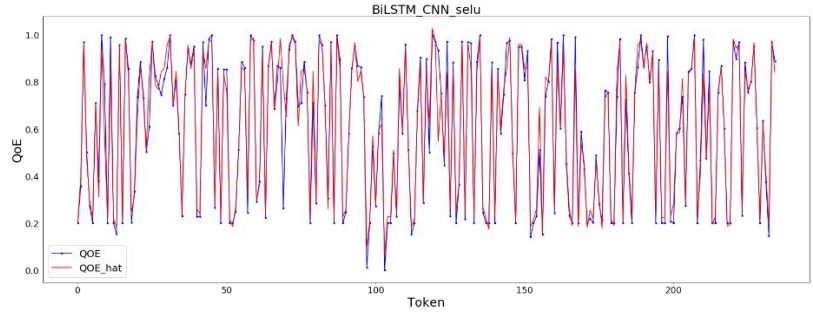
(a)



(b)



(c)



(d)

Figure 35: Forecasted vs actual QoE for four ML Models.

5.4 Summary

QoE plays crucial role in video network management systems. We proposed a ML based multivariate time series method to forecast video QoE before its effect shows at the client's screen. This approach gives more time to the management system to avoid QoE degradation and reduce customers' dissatisfaction. Evaluation results showed that our proposed BiLSTM-CNN method attains higher performance than other prominent ML methods.

Chapter 6. Conclusion and Future Works

In this chapter, we conclude the works conducted in this thesis, then explain future research.

6.1 Conclusion

With the advancement in video technology, the number and variety of multimedia applications had an extensive increase in the past years and expected more growth in the future. Despite this remarkable revenue, due to the higher quality demanded, the real-time nature of the quality, and increased complexity in the end-to-end networks, video service management has been challenging—users are accustomed to more resource-demanding service and higher QoE. The end user's experience is the fundamental criterion for a stable video service delivery. However, achieving high QoE is crucial due to its dependency on various elements such as types of devices, content dependency, network conditions, and spatial and temporal variation of the influence factors. In recent years, AI and ML enhancement have provided state-of-the-art solutions to address many challenging tasks.

This thesis's overall focus was on applying AI and end user's QoE in video service management. Two chosen challenging systems were (1) server selection in cloud gaming, where most of the computationally intensive tasks are uploaded into cloud rendering servers, and (2) multi-entities video network diagnosis, where a single entity does not have access to the end-to-end path.

The study of the CG systems and QoE of the game players in Chapter 2 gave us enough insights into CG management systems' challenges and the crucial parameters that should be considered. Using this study, in Chapter 3, we proposed an efficient resource allocation process that satisfies both service providers and players. CG is an interactive system, and game players are not regular video service users; they have specific quality expectations. We

evaluated the cloud gaming systems and influence factors of the player's QoE and formulated a new optimization framework for server selection in cloud gaming. In addition, we modeled delay and QoE as a function of the game's frame rate, which is a crucial parameter in the gamers' performance. The proposed framework maximizes GPU utilization and game player's QoE simultaneously. The optimization model considers the load of the requested games by a group of players and assigns them to the servers to maximize utilization while increasing players' experiences. We solved the problem with two metaheuristic algorithms, i.e., GA and PSO, and improved them by adding a supportive block into the chain of the main algorithms. This modification keeps the algorithms away from premature convergence. The proposed Boosted-GA and Boosted-PSO outperformed GA, PSO, and four other well-known greedy algorithms, some of which were used in cloud gaming resource provisioning. The Boosted-PSO showed the highest efficiency in terms of utilization, capacity wastage, players' QoE, and stability to variations of the number of players.

Another challenge investigated in this thesis is the QoE-aware fault diagnosis mechanism, which is the heart of every service management system in packet video networks. Chapter 2 also explored the packet video network diagnosis problem and two major fault diagnosis approaches: proactive and reactive. In proactive methods, the system estimates QoE using QoS parameters, then detects and isolates the network issue based on the level of the estimated QoE; the reactive approach uses actual QoE influence factors on the client-side and localizes the fault using end-to-end traceroute data. Despite the former, the latter uses actual QoE parameters that reflect the user's experience precisely. However, both approaches require access to the end-to-end path, which is not practical in multi-entity and multi-owner networks. In Chapter 4, we proposed a novel fault detection/localization approach using only client-side data and the capability of deep learning to address this challenge. To this end, we conducted a multivariate time series data collection in an

industry video streaming testbed. From the start time of the videos' playback until the stop time, we collected six QoE metrics at the client-side in each time-step. The collected dataset has three different patterns: no issues, ISP issue, and client issue. We modeled two state-of-the-art deep learning models: MLP and LSTM, for a sequence classification problem. The problem's objective was to detect the issue and localize it between the client-side and ISP side if the issue was detected. The deep learning models were trained using the collected dataset. They detect and localize the issue precisely. We compared them with two other well-known ML methods: SVM and DT. The 10-fold cross-validation results showed that the MLP followed by LSTM has the highest efficiency in terms of accuracy, precision, recall, and F1-score.

Furthermore, we evaluated the catastrophic condition in a path, when congestion happens in both client-side and ISP side and aborts the video playback. Although our proposed mechanism can detect this condition one by one, we added it as a new class, i.e., client+ISP_issue, to our model and trained the models again. Since our new class does not have enough samples, we faced an imbalanced class problem that resulted in a degradation in the algorithm performance in terms of sensitivity. We tried to address the problem at the data level by employing four combinations of oversampling and undersampling for the majority and minority classes. Then, we trained the deep learning algorithms using these new datasets. The test results showed a noticeable improvement when the models were trained with the resampled datasets. The resampled data that oversampled the classes to be equal to the majority class showed the most improvement among the four resampling patterns.

Almost all proactive and reactive approaches estimate QoE at the current time; however, since the reactive system uses client-side metrics, it suffers from an inherent shortcoming, i.e., lateness. In other words, the issue appearing on the screen has already happened at the delivery network. To smooth this shortcoming, we proposed a deep learning sequence model, in Chapter 5, to forecast QoE metrics at the next time step and before occurring

on the client's screen. This prognosticating approach gives the provider enough time to localize and fix the fault before the user notices any degradation or reduces the latter's duration. To this end, we proposed a multivariate time-series process to forecast QoE metrics at the future time-step. We designed a state-of-the-art hybrid deep learning model, i.e., BiLSTM-CNN, and trained it using the time-series data collected from our industry testbed. The test results showed our hybrid DL method's supremacy over the other prominent machine learning methods with minimum error. The proposed forecasting process can be used in other video streaming applications, such as anomaly detection in CDN and wireless networks and resource scheduling in wireless networks for adaptive DASH delivery.

6.2 Future Work

The extension topics for the proposed approaches in this thesis are evaluated in this section.

For the CG resource allocation system presented in Chapter 3, the QoE model's improvement can be explored. For example, by considering other quality metrics, such as the player's device type, e.g., smartphone, tablet, etc., their screen size, viewing angle, and distance can be evaluated. Besides, adding a forecasting mechanism for player's required QoE, similar to the one presented in chapter 5, can protect the CG resource allocation system from the potential hackers and their incorrect requested quality on the client-side. Moreover, modeling other state-of-the-art machine learning methods such as Reinforcement Learning (RL) can be investigated for this type of problems. For such an algorithm, creating an appropriate CG environment and determining the rewards and corresponding actions would be challenging. Furthermore, the proposed objective function can be extended to a cost-effective algorithm by considering an expense model of the cloud infrastructures used by players during the game sessions.

There are several opportunities to improve the proposed approaches in chapters 4 and 5. The proposed model in Chapter 4 is a supervised model that needed to be retrained for the new end-to-end path configuration. As such, investigating an accurate unsupervised model that can be updated for the system's new configuration would be considered a remarkable improvement for this solution and make it more practical and incentive for video service providers. Furthermore, collecting a new dataset from the more extensive network with real entities and evaluating this method would be an open challenge for this approach. Collecting a time series data set similar to the one presented in chapter 4 requires cooperation among multiple entities in the end-to-end path, which is a challenging task. But, in today's real networks, some VSPs and ISPs do have agreements to let the VSP reroute traffic within the ISP. For example, Netflix uses Open Connect Appliances (OCA) [113] inside some of its ISP partners to offload Netflix content traffic from peering or transport circuits. Therefore, a similar cooperation mechanism can be applied for collecting an appropriate dataset for training the fault detection/isolation model presented in chapter 4. The algorithms only need to use this dataset for its first training, and it can update itself for the rest of its life. More investigation on forecasting systems, proposed in Chapter 5, such as multi-time-steps forecasting, can be an open and exciting research topic. The proposed approaches in chapters 4 and 5 can also be investigated in a comprehensive management system where the system can fix the issue, for example, in a software-defined networking (SDN) framework. These virtualized networks allow the management system to take some level of action on the network resources based on the diagnostics algorithm's warnings.

References

[1] "The Global Games Market Will Generate $152.1 Billion in 2019 as the U.S. Overtakes China as the Biggest Market | Newzoo." [Online]. Available: https://newzoo.com/insights/articles/the-global-games-market-will-generate-152-1-billion-in-2019-as-the-u-s-overtakes-china-as-the-biggest-market/. [Accessed: 20-Feb-2021].

[2] M. . .Semsarzadeh, M. Hemmati, A. Javadtalab, A. Yassine, and S. Shirmohammadi, "A video encoding speed-up architecture for cloud gaming," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014, pp. 1–6.

[3] "Sony says cloud gaming could be future threat to PlayStation business." [Online]. Available: https://www.businessinsider.com/sony-says-cloud-gaming-future-threat-to-playstation-business-2019-2. [Accessed: 20-Feb-2021].

[4] "Global Cloud Gaming Market Will Reach USD 6,944 Million By 2026: Zion Market Research." [Online]. Available: https://www.globenewswire.com/news-release/2018/12/31/1679151/0/en/Global-Cloud-Gaming-Market-Will-Reach-USD-6-944-Million-By-2026-Zion-Market-Research.html. [Accessed: 20-Feb-2021].

[5] "Sony Computer Entertainment to Acquire Gaikai Inc., A Leading Interactive Cloud Gaming Company SCE To Build A Cloud Service Bringing Gaikai's Cloud Based-Streaming Technologies into Its Network Business." [Online]. Available: https://www.sie.com/en/corporate/release/2012/120702.html. [Accessed: 20-Feb-2021].

[6] "OnLive." [Online]. Available: http://onlive.com/. [Accessed: 20-Feb-2021].

[7] "PlayStation Now game-streaming service coming summer 2014 (update) - Polygon." [Online]. Available: https://www.polygon.com/2014/1/7/5284504/sony-playstation-now-gaikai-based-streaming-service-ps-ps2-ps3. [Accessed: 20-Feb-2021].

[8] "Company Profile | Broadmedia GC Corporation." [Online]. Available: https://www.broadmediagc.co.jp/en/corporation/. [Accessed: 20-Feb-2021].

[9] "Stadia Founder's Edition - One place for all the ways we play - Google Store." [Online]. Available: https://store.google.com/ca/product/stadia_learn. [Accessed: 20-Feb-2021].

[10] "PlayStation Now – Online Streaming Services on PS4 or PC - PlayStation." [Online]. Available: https://www.playstation.com/en-ca/explore/playstation-now/. [Accessed: 20-Feb-2021].

[11] “Game Streaming - Play Games With Friends | Parsec.” [Online]. Available: https://parsecgaming.com/. [Accessed: 20-Feb-2021].

[12] “Vortex - Cloud Gaming for Android, PC and macOS.” [Online]. Available: https://vortex.gg/. [Accessed: 20-Feb-2021].

[13] “Game anywhere on your Mac, Windows PC, or SHIELD device with NVIDIA’s cloud gaming service.” [Online]. Available: https://www.nvidia.com/en-us/geforce/products/geforce-now/. [Accessed: 20-Feb-2021].

[14] “PLAYKEY | games online.” [Online]. Available: https://welcome.playkey.net/en/lp/eu-quiz-before/. [Accessed: 20-Feb-2021].

[15] S. Shirmohammadi, M. Abdalla, D. T. Ahmed, K.-T. Chen a.k.a. ShengWei Chen, Y. Lu, and A. Snyatkov, “Introduction to the Special Section on Visual Computing in the Cloud: Cloud Gaming and Virtualization,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1955–1959, Dec. 2015.

[16] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu, “Cloud gaming onward: research opportunities and outlook,” in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014, pp. 1–4.

[17] M. Amiri, H. Al Osman, S. Shirmohammadi, H. Al Osman, and M. Abdallah, “Toward delay-efficient game-aware data centers for cloud gaming,” *ACM Trans. Multimedia Comput. Commun. Appl*, vol. 12, no. 71, 2016.

[18] Y. Chen, J. Liu, and Y. Cui, “Inter-player Delay Optimization in Multiplayer Cloud Gaming,” in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, 2016, pp. 702–709.

[19] M. Amiri, A. Sobhani, H. Al Osman, and S. Shirmohammadi, “SDN-Enabled Game-Aware Routing for Cloud Gaming Datacenter Network,” *IEEE Access*, vol. 5, pp. 18633–18645, Sep. 2017.

[20] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei, “Measuring the latency of cloud gaming systems,” in *Proceedings of the 19th ACM international conference on Multimedia - MM ’11*, 2011, p. 1269.

[21] Y. Deng, Y. Li, X. Tang, and W. Cai, “Server Allocation for Multiplayer Cloud Gaming,” in *Proceedings of the 24th ACM international conference on Multimedia. ACM*, 2016, pp.

918–927.

[22] "Cisco Visual Networking Index: Forecast and Trends,2017–2022," *Cisco February 27, 2019*. [Online]. Available: www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. [Accessed: 24-Feb-2021].

[23] "Global Internet Phenomena Report," *Sandvine,*. [Online]. Available: www.sandvine.com/press-releases/sandvine-releases-2019-global-internet-phenomena-report. [Accessed: 24-Feb-2021].

[24] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "The brewing storm in cloud gaming: A measurement study on cloud to end-user latency," in *11th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2012, pp. 1–6.

[25] H. Ahmadi, S. Khoshnood, M. R. Hashemi, and S. Shirmohammadi, "Efficient bitrate reduction using a Game Attention Model in cloud gaming," in *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)*, 2013, pp. 103–108.

[26] E. Babaei, M. R. Hashemi, and S. Shirmohammadi, "A State-based Game Attention Model for Cloud Gaming," in *Proceedings of the 15th Annual Workshop on Network and Systems Support for Games, IEEE Press*, 2017, pp. 34–36.

[27] H. Dar, J. Kwan, Y. Liu, O. Pantazis, and R. Sharp, "The Game Performance Index for Mobile Phones," 30-Oct-2019. [Online]. Available: http://arxiv.org/abs/1910.13872.

[28] R. Ellerweg, "Make Frame Rate Studies Useful for System Designers," in *Proceedings - ICGI 2018: International Conference on Graphics and Interaction*, 2018, p. Lisbon, 2018, pp. 1–8.

[29] A. MacKin, F. Zhang, and D. R. Bull, "A Study of High Frame Rate Video Formats," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1499–1512, Jun. 2019.

[30] Y. Kuroki, T. Nishi, S. Kobayashi, H. Oyaizu, and S. Yoshimura, "3.4: Improvement of Motion Image Quality by High Frame Rate," *SID Symposium Digest of Technical Papers*, vol. 37, no. 1, p. 14, Jun. 2006.

[31] M. Emoto, Y. Kusakabe, and M. Sugawara, "High-frame-rate motion picture quality and its independence of viewing distance," *IEEE/OSA Journal of Display Technology*,

vol. 10, no. 8, pp. 635–641, 2014.

[32] "BT.2020 : Parameter values for ultra-high definition television systems for production and international programme exchange." [Online]. Available: https://www.itu.int/rec/R-REC-BT.2020-2-201510-I/en. [Accessed: 17-Jan-2021].

[33] M. Claypool, K. Claypool, and F. Damaa, "The effects of frame rate and resolution on users playing first person shooter games," in *Multimedia Computing and Networking*, 2006, vol. 6071, pp. 607101, International Society for Optics and Photo.

[34] J. A. Berton and K.-L. Chuang, "Effects of Very High Frame Rate Display in Narrative CGI Animation," in *2016 20th International Conference Information Visualisation (IV)*, 2016, pp. 395–398.

[35] H. Nam, K. H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *Proceedings - IEEE INFOCOM*, 2016, vol. 2016-July, p. San Francisco, CA, USA, 1-9.

[36] H. E. Dinaki and S. Shirmohammadi, "GPU/QoE-Aware Server Selection Using Metaheuristic Algorithms in Multiplayer Cloud Gaming," in *2018 16th Annual Workshop on Network and Systems Support for Games (NetGames)*, 2018, pp. 1–6.

[37] E. Dhib, K. Boussetta, N. Zangar, and N. Tabbane, "Cost-aware virtual machines placement problem under constraints over a distributed cloud infrastructure," in *2017 Sixth International Conference on Communications and Networking (ComNet)*, 2017, pp. 1–5.

[38] A. Laghrissi and T. Taleb, "A Survey on the Placement of Virtual Resources and Virtual Network Functions," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2. pp. 1409–1434, 01-Apr-2019.

[39] H. Shengli, "Detecting Concealed Information in Text and Speech," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, p. Felorence, Italy, 402-412.

[40] I. R. Widiasari, L. E. Nugroho, and Widyawan, "Deep learning multilayer perceptron (MLP) for flood prediction model using wireless sensor network based hydrology time series data mining," in *Proceedings - ICITech Conf.*, 2017, vol. 2018-Janua, p. Salatiga, 1-5.

[41] W. Cai, M. Chen, and V. C. M. Leung, "Toward Gaming as a Service," *IEEE Internet Computing*, vol. 18, no. 3, pp. 12–18, May 2014.

[42] X. Nan *et al.*, "Delay-Rate-Distortion Optimization for Cloud Gaming with Hybrid Streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 12, pp. 2687–2701, Dec. 2017.

[43] S. Choy, B. Wong, G. Simon, and C. Rosenberg, "A hybrid edge-cloud architecture for reducing on-demand gaming latency," *Multimedia Systems*, vol. 20, no. 5, pp. 503–519, Oct. 2014.

[44] M. Claypool, "Motion and Scene Complexity for Streaming Video Games," in *Proceedings of the 4th International Conference on Foundations of Digital Game,ACM*, 2009, pp. 34–41.

[45] P. B. Beskow, P. Halvorsen, and C. Griwodz, "Latency Reduction in Massively Multi-player Online Games by Partial Migration of Game State," *Second International Conference on Internet Technologies and Applications, Wrexham, Wales*, pp. 153–163, 2007.

[46] Y. Gao, L. Wang, and J. Zhou, "zCost-Efficient and Quality of Experience-Aware Provisioning of Virtual Machines for Multiplayer Cloud Gaming in Geographically Distributed Data Centers," *IEEE Access*, vol. 7, pp. 142574–142585, Oct. 2019.

[47] Y. Deng, Y. Li, R. Seet, X. Tang, and W. Cai, "The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1233–1245, May 2018.

[48] X. Zhang *et al.*, "Improving Cloud Gaming Experience through Mobile Edge Computing," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 178–183, Aug. 2019.

[49] Y. Li, Y. Deng, X. Tang, W. Cai, X. Liu, and G. Wang, "Cost-efficient server provisioning for cloud gaming," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 14, no. 3s, Jun. 2018.

[50] K.-T. Chen, Y.-C. Chang, H.-J. Hsu, D.-Y. Chen, C.-Y. Huang, and C.-H. Hsu, "On the Quality of Service of Cloud Gaming Systems," *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 480–495, Feb. 2014.

[51] H.-J. Hong, D.-Y. Chen, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Placing Virtual

Machines to Optimize Cloud Gaming Experience," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 42–53, Jan. 2015.

[52] H. Tian, D. Wu, J. He, Y. Xu, and M. Chen, "On Achieving Cost-Effective Adaptive Cloud Gaming in Geo-Distributed Data Centers," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 2064–2077, Dec. 2015.

[53] C. V. Murudkar and R. D. Gitlin, "QoE-driven Anomaly Detection in Self-Organizing Mobile Networks using Machine Learning," in *Wireless Telecommunications Symposium*, 2019, vol. 2019-April, p. New York City, NY, USA, 1-5.

[54] M. Hemmati, B. McCormick, and S. Shirmohammadi, "QoE-Aware Bandwidth Allocation for Video Traffic Using Sigmoidal Programming," *IEEE Multimedia*, vol. 24, no. 4, pp. 80–90, Oct. 2017.

[55] O. Dobrijevic, M. Santl, and M. Matijasevic, "Ant colony optimization for QoE-centric flow routing in software-defined networks," in *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*, 2015, pp. 274–278.

[56] C. Wang, "QoE Based Management and Control for Large- Scale VoD System in the Cloud – chapter 5," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2017.

[57] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, K. Papagiannaki, and P. Steenkiste, "Identifying the root cause of video streaming issues on mobile devices," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT 2015*, 2015, p. Heidelberg, Germany, 1-13.

[58] X. Wei, Z. Wu, L. Zhou, and Z. Dong, "An Integrated Quality Assessment for IPTV Operation and Maintenance," in *IEEE Vehicular Technology Conf.*, vol. 2017-June, p. Sydney, NSW, 1-5.

[59] J. Ahmed, A. D. N. Junior, C. Kilinc, D. Pan, J. R. I Riu, and J. Gustafsson, "Using Blackbox ML Techniques to Diagnose QoE Problems for an IPTV Service," in *Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020*, 2020.

[60] H. H. Song *et al.*, "Q-score: Proactive service quality assessment in a large IPTV system," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 2011, pp. 195–208.

[61] V. Vasilev, J. Leguay, S. Paris, L. Maggi, and M. Debbah, "Predicting QoE Factors with Machine Learning," in *IEEE International Conference on Communications*, 2018, vol. 2018-May.

[62] R. Shalala, R. Dubin, O. Hadar, and A. Dvir, "Video QoE Prediction Based on User Profile," in *2018 International Conference on Computing, Networking and Communications, ICNC 2018*, 2018, pp. 588–592.

[63] U. Bulkan and T. Dagiuklas, "Predicting quality of experience for online video service provisioning," *Multimedia Tools and Applications*, vol. 78, no. 13, pp. 18787–18811, Jul. 2019.

[64] C. G. Bampis, Z. Li, and A. C. Bovik, "Continuous prediction of streaming video QoE using dynamic networks," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 1083–1087, Jul. 2017.

[65] C. G. Bampis, Z. Li, I. Katsavounidis, and A. C. Bovik, "Recurrent and Dynamic Models for Predicting Streaming Video Quality of Experience," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3316–3331, Jul. 2018.

[66] S. Morishima, M. Okazaki, and H. Matsutani, "A Case for Remote GPUs over 10GbE Network for VR Applications," in *Proceedings of the 8th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, 2017, vol. 6.

[67] F. Metzger, A. Rafetseder, and C. Schwartz, "A Comprehensive End-to-End Lag Model for Online and Cloud Video Gaming," in *5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)*, 2016, pp. 15–19.

[68] "GitHub - mas-ude/onlinegame-lag-sim." [Online]. Available: https://github.com/mas-ude/onlinegame-lag-sim.

[69] T. Zinner, O. Hohlfeld, O. Abboud, and T. Hossfeld, "Impact of frame rate and resolution on objective QoE metrics," in *2010 Second International Workshop on Quality of Multimedia Experience (QoMEX)*, 2010, pp. 29–34.

[70] "FFmpeg." [Online]. Available: http://ffmpeg.org/.

[71] I. Slivar, L. Skorin-Kapov, and M. Suznjevic, "Cloud gaming qoe models for deriving video encoding adaptation strategies," in *Proceedings of the 7th International Conference on Multimedia Systems, MMSys 2016*, 2016, pp. 185–196.

[72] "George Millington, 'New AMD Radeon™ Pro V340 Graphics Card Delivers Accelerated Performance and High User Density to Power Datacenter Visualization Workloads', AMD Press Release, August 26, 2018." [Online]. Available: https://www.globenewswire.com/news-release/2018/08/26/1556638/0/en/New-AMD-Radeon-Pro-V340-Graphics-Card-Delivers-Accelerated-Performance-and-High-User-Density-to-Power-Datacenter-Visualization-Workloads.html.

[73] M. R. Johnson, David S and Garey, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[74] E. G. Cooman and J. M. R Garey D S Johnson, "Approximation Algorithms for Bin Packing: A Survey," in *Approximation algorithms for NP-hard problems*, 1996, pp. 46–93.

[75] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pp. 1186–1192.

[76] R. Kennedy, J and Eberhart, "Particle swarm optimization (PSO)," in *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, 1995, p. {1942--1948.

[77] H. Moayedi, M. Mehrabi, M. Mosallanezhad, A. S. A. Rashid, and B. Pradhan, "Modification of landslide susceptibility mapping using optimized PSO-ANN technique," *Engineering with Computers*, vol. 35, no. 3, pp. 967–984, Jul. 2019.

[78] A. Jaafari, E. K. Zenner, M. Panahi, and H. Shahabi, "Hybrid artificial intelligence models based on a neuro-fuzzy system and metaheuristic optimization algorithms for spatial prediction of wildfire probability," *Agricultural and Forest Meteorology*, vol. 266–267, pp. 198–207, Mar. 2019.

[79] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[80] D. E. Goldberg and J. H. Holland, "Genetic Algorithms and Machine Learning," *Machine Learning*, vol. 3, no. 2/3, pp. 95–99, 1988.

[81] J. H. Holland, "Adaptation in natural and artificial systems," *Ann Arbor, MI: University of Michigan Press and*, 1975.

[82] D. Finkel, M. Claypool, S. Jaffe, T. Nguyen, and B. Stephen, "Assignment of games to servers in the OnLive cloud game system," in *Annual Workshop on Network and Systems Support for Games*, 2014.

[83] R. M. Nasiri, J. Wang, A. Rehman, S. Wang, and Z. Wang, "Perceptual quality assessment of high frame rate video," in *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, 2015, pp. 1–6.

[84] S. Boraa *et al.*, "A Comparison of Programming Languages In Economics," NBER Working Paper Series, 2014.

[85] "Kaltura." [Online]. Available: https://corp.kaltura.com/products/video-delivery-video-streaming-platform/. [Accessed: 24-Feb-2021].

[86] "Ixia Makes Networks Stronger." [Online]. Available: https://www.ixiacom.com/. [Accessed: 24-Feb-2021].

[87] "Nice People At Work (NPAW)." [Online]. Available: https://nicepeopleatwork.com/. [Accessed: 24-Feb-2021].

[88] "Getting Started with Headless Chrome | Web | Google Developers." [Online]. Available: https://developers.google.com/web/updates/2017/04/headless-chrome. [Accessed: 24-Feb-2021].

[89] "BT.500 : Methodologies for the subjective assessment of the quality of television images," *ITU*, Oct-. [Online]. Available: www.itu.int/rec/R-REC-BT.500-14-201910-I/en. [Accessed: 24-Feb-2021].

[90] "P.917 : Subjective test methodology for assessing impact of initial loading delay on quality of experience." [Online]. Available: www.itu.int/rec/T-REC-P.917-201901-P. [Accessed: 24-Feb-2021].

[91] M. H. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312–322, Sep. 2004.

[92] D. Z. Rodriguez, J. Abrahao, D. Begazo, R. L. Rosa, and G. Bressan, "Quality metric to assess video streaming service over TCP considering temporal location of pauses," *Transactions on Consumer Electronics*, p. Vol. 58, No. 3, 985-992, 2012.

[93] "G.1070 : Opinion model for video-telephony applications." [Online]. Available: https://www.itu.int/rec/T-REC-G.1070. [Accessed: 24-Feb-2021].

[94] H. Liang, X. Sun, Y. Sun, and Y. Gao, "Text feature extraction based on deep learning: a review," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, p. 211, 2017.

[95] I. G. and Y. B. and A. Courville, *Deep Learning*. MIT Press, 2016.

[96] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java." [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 26-Feb-2021].

[97] "scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation." [Online]. Available: https://scikit-learn.org/stable/. [Accessed: 26-Feb-2021].

[98] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[99] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002.

[100] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Lecture Notes in Computer Science*, 2005, vol. 3644, no. PART I, pp. 878–887.

[101] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proceedings of the International Joint Conference on Neural Networks*, 2008, pp. 1322–1328.

[102] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, "Classification with class imbalance problem: A Review," *Int. J. Advance Soft Compu. Appl*, vol. 7, no. 3, pp. 176–204, 2015.

[103] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 1126–1165, Apr. 2015.

[104] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 2s, pp. 1–29, May 2018.

[105] C. G. Bampis, Z. Li, A. K. Moorthy, I. Katsavounidis, A. Aaron, and A. C. Bovik, "Study of temporal effects on subjective video quality of experience," *IEEE Transactions on*

Image Processing, vol. 26, no. 11, pp. 5217–5231, Nov. 2017.

[106] A. J. Greene, C. Prepscius, and W. B. Levy, "Primacy versus recency in a quantitative model: Activity is the critical distinction," *Learning and Memory*, vol. 7, no. 1, pp. 48–57, Jan. 2000.

[107] M. S. Anwar, J. Wang, A. Ullah, W. Khan, S. Ahmad, and Z. Fei, "Measuring quality of experience for 360-degree videos in virtual reality," *Science China Information Sciences*, vol. 63, no. 10, p. 15, Oct. 2020.

[108] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, vol. 07-12-June-2015, pp. 3156–3164.

[109] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *ACL-IJCNLP 2015 - 53rd Annual Meeting*, 2015, vol. 2, pp. 707–712.

[110] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 972–981, Jun. 2017.

[111] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The Performance of LSTM and BiLSTM in Forecasting Time Series," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, 2019, pp. 3285–3292.

[112] T. Dozat, "Workshop track-ICLR 2016 INCORPORATING NESTEROV MOMENTUM INTO ADAM," Feb. 2016.

[113] "Netflix | Open Connect." [Online]. Available: https://openconnect.netflix.com/en/. [Accessed: 24-Apr-2021].