NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

GRADUATE PROGRAM SMART TELECOM AND SENSING NETWORKS (SMARTNET)

MSc THESIS

ML-based Adaptive Video Streaming techniques for 5G and beyond mobile data networks

Akhmadjon I. Rajabov

Supervisors: **Dimitrios Syvridis,** Prof.
Nikos Passas, Dr.
Dionysis G. Xenakis, Dr.

ATHENS

OCTOBER 2021

MSc THESIS

ML-based Adaptive Video Streaming techniques for 5G and beyond mobile data networks

Akhmadjon I. Rajabov

S.N.: 120.0002

SUPERVISOR: **Dimitrios Syvridis,** Prof.
Nikos Passas, Dr.
Dionysis G. Xenakis, Dr.

Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus Mundus Joint Master's Degree

"SMART Telecom and Sensing NETworks" (SMARTNET) (**2019/2021 intake**)

Aston University, Triangle, B4 7ET / Birmingham, UK

Email: aipt_smartnet@aston.ac.uk / Web-site: smartnet.astonphotonics.uk/

Acknowledgement

This Master Thesis has been accomplished in the framework of the European Funded Project: **SMART Telecom and Sensing Networks (SMARTNET)** - Erasmus+ Programme Key Action 1: Erasmus Mundus Joint Master Degrees – Ref. Number 2017 – 2734/001 – 001, Project number - 586686-EPP-1-2017-1-UK-EPPKA1-JMD-MOB, coordinated by **Aston University,** and with the participation of **Télécom SudParis, member of IP Paris** and **National and Kapodistrian University of Athens.**

Aston University
BIRMINGHAM UK

TELECOM SudParis
TELECOM ParisTech

ABSTRACT

An artificial neural network application for adaptive video streaming has been used and studied for several years. However, for some applications, they are still under development and have become one of the academic and industrial research lines in machine learning. One of these applications focuses on perceptual end-users prediction for adaptive video streaming techniques in 5G mobile networks.

This dissertation looks at how different neural network topologies are represented, with the goal of influencing the development of QoE prediction models for streaming video. In addition, this paper presents a cutting-edge neural network architecture for QoE prediction targets that connects the convolutional layer to the bidirectional LSTM layer. For comparison, the efficacy of several previously suggested neural network models - a three-layer CNN and a two-layer LSTM perceptron network - has been built and assessed. To explain their hyperparameters and topologies, this dissertation presented two-layered biLSTM, three-layered FNN, and mixed CNN and LSTM QoE models. These neural netowrks models were trained using real experimental data from the University of Texas at Austin – Image and Video Engineering Lab's LIVE NETFLIX video QoE database.

Simulation results were evaluated using PCC, SROCC and RMSE metrics to demonstrate the effectiveness of accurate QoE prediction for an adaptive 5G video streaming system. Additionally, the complexity of the proposed architecture of neural networks was calculated. After analyzing the comparison results of the studied QoE models, the FNN model provided the best level of forecasting accuracy by the RSME value and, at the same time, occupied one of the lowest levels of computational complexity. This indicates that FNN can be the best method for QoE prediction for 5G video streaming due to its relatively low complexity and competitively high prediction accuracy.

Subject area: Machine Learning, Quality of Experience, Adaptive Video Streaming.

Keywords: Quality of Experience, video streaming, convolutional neural networks, feed-forward neural networks, long-short term memory, 5G, mobile networks, bidirectional LSTM.

ACKNOWLEDGMENTS

I want to declare my appreciation for the Smart Telecom and Sensing Networks (SMARTNET) project within the Erasmus + program of the European Union for the scholarship and funding. I was delighted to take part in this great program. In addition, I would like to express my gratitude to my advisors and supervisors, Assoc. Prof. Stylianos Sigletos, Professor Dimitris Sivridis, Dr. Nikos Passas, and Dionysis G. Xenakis, for their support and advice during the dissertation.

I'd like to express my gratitude to Tetyana Gordienko, Erasmus Project Administrator, and Zorina Bousboura for their assistance and organization during the SMARTNET program.

Also, I would like to show respect and recognition to Tatiana Gordienko, Erasmus Project Administrator, and Zorina Bousbura for their coordination and assistance while I engaged in the SMARTNET program.

I am very appreciative to my SMARTNET friends who become my friends of life, especially my best friends, Sasipim, Khojiakbar, Yevhenii, Mahmoud, Ameen, and Marc, share the moments and help me get through this tough COVID time with their sincere assistance. Special thanks to Yevhenii for his time to proofread and his suggestions on the thesis writing. Additionally, I could not make it through the two-year master’s degree abroad without my family, who encouraged me and always got in touch to cheer me up. Without these people, this thesis would not have been successful.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Traffic outlook in today's networks

Video is currently the most widely consumed content on the Internet. According to the latest Ericsson Mobility Report [1], during 2020, the average annual traffic growth rate remained at an intermediate level of about 46%, despite an unusual peak in 2019. As a result, total mobile data traffic in Q1 2021 reached 66 exabytes. Over time, the increase in traffic is attributed to the fascination with the number of smartphone subscriptions and the average amount of data per subscription, primarily due to the more significant number of video content views. Figure 1 illustrates the total global network data and voice traffic per month from Q1 2014 to Q1 2021 and the percentage change in mobile data traffic on an annualised basis.

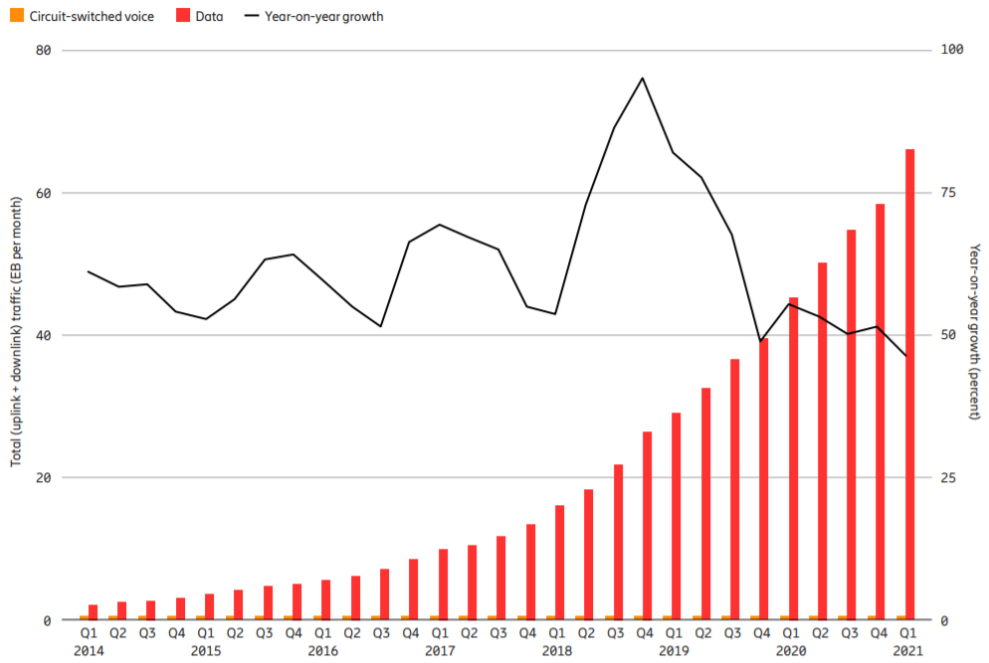


Figure 1. Data traffic on mobile networks around the world and year-over-year growth (exabytes per month) [1].

The total mobile traffic is estimated to grow five times by 2026, approximately 300 EB per month. At the moment, smartphones are at the centre of the development and generation of traffic growth, which generates about 95% of mobile traffic, of which 66% is video traffic, one smartphone user monthly exceeds 10 GB of traffic, and by 2026 this figure could reach about 35 GB. In addition, markets that launch 5G faster are also driving traffic growth, and 5G is forecast to carry 53% of total mobile traffic, of which 77% could be video traffic by 2026 [1].

Another interesting Cisco Annual Report [2] highlights that video devices, especially HD TV with internet access, produce a massive amount of traffic as a whole household today on average in two to three hours. Furthermore, due to the implementation of Ultra High Definition (UHD), or 4K, video streaming, since 4K requires a bit rate between 15 and 18 Mbps which is more than twice the bit rate for HD video and nine times more significant for standard definition video. Furthermore, several countries users currently experience 125 Mbps broadband speed, which opens the route to future video demands. Nowadays, video applications are of tremendous need in homes, but there could be considerable bandwidth demands in the future and beyond the prediction time of 2023 [2]. In figure 2 demonstrates the mobile data traffic consumption and growth.

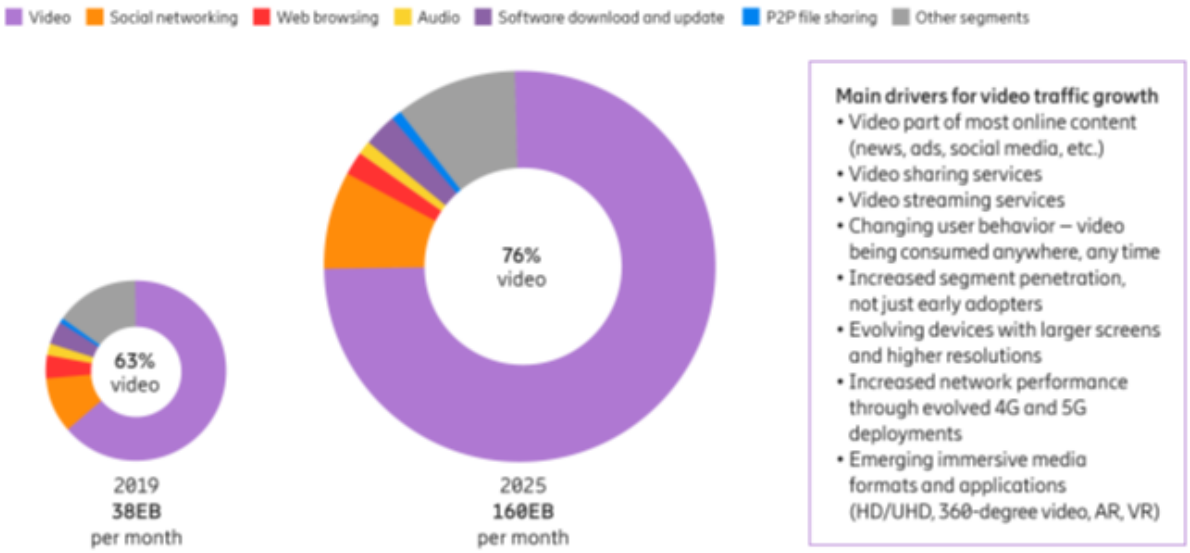


Figure 2. Global mobile network data traffic [2].

According to the Cisco Annual Report [2], Internet of Things equipment's would count for 14,7 billion of all global network devices by 2023. Equipment producers, business software companies, mobile operators, system integrators, and infrastructure vendors would have special but supplementary parts across the Internet of Things environment. Edge computing and networks enable corporate architectures to optimise working for business analysis critical data sets from the Internet of Things applications and communications. Multi-access edge enabled by 5G and Wi-Fi 6 would advantage the low-latency real-time connections and high definition video applications, which maintains better video streaming to end-users [2].

1.2 What is Video Streaming

The process of constantly transmitting and consuming information from a source with little or no intermediate storage in network components is known as video streaming. Video is split into data packets and played on the client side when it is streamed over the Internet. H.264, VP8, and VP9 are some of the coding types used in video streaming. The encoded video stream is containerized into a "bitstream" by utilizing MP4, FLV, or WebM, and it is sent over the Internet through a transport protocol from the streaming server to the client. The conventional approach for video streaming used to start from an event's location or from a single user to the cloud, where transcoding [3] took place. The Content Delivery Network (CDN) [6] then utilizes Hypertext Transfer Protocol (HTTP)-based transport protocols to distribute video to clients, with management protocols such as Real-Time Protocol or Real-Time Streaming Protocol used for communication (RTP or RTSP). A conventional technique for deploying streaming server clusters for streaming services, including video streaming, is also available. The central server, which maintains all copies of media files and the Internet Protocol (IP) addresses of distributed servers, is in charge of this method, which specifies that in-network dispersed regional servers are managed by the central server. After then, the central server employs scheduling and load balancing algorithms to route customers to one of the dispersed servers closest to them.

As previously mentioned, growing video traffic causes major difficulties for mobile networks during data transfer, necessitating infrastructure that provides reduced latency and higher data rates for multimedia transmission in a time-varying wireless environment in order to meet high QoE requirements. To address these issues, the HTTP Adaptive Streaming (HAS) [4] technology was created to enable video transmission over mobile networks. The basic idea behind HTTP adaptive video streaming over cellular networks is that the video sequence is split into chunks, each of which is encoded with several discrete bit rates. The appropriate video bitrate is then chosen based on the achievable bit rate over time in the wireless channel [4].

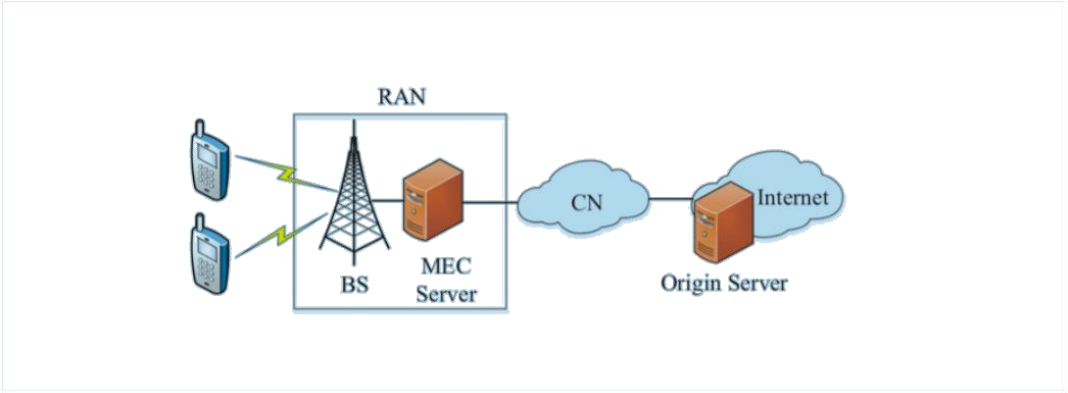


Figure 3. MEC-based architecture for 5G mobile networks [6].

The European Telecommunications Standards Institute (ETSI) has proposed a new possible method called multi-access edge computing (MEC), which amplifies 5G networks to enable data transmission [4]. MEC may use cloud computing and edge IT capabilities to manage video content placement and perform video conversion, smoothing out transient traffic changeability, response latency, and reducing congestion while increasing QoE. Furthermore, Dynamic Adaptive Streaming over HTTP (DASH) is rapidly becoming the preferred standard for Video-on-Demand (VoD) and live streaming over HTTP in a 5G network [4]. DASH is a client-side method that begins with a multimedia presentation description (MPD) file extraction. Data about the video chunks that are available, called segments for a specific context, bitrate, codecs, length, and the Uniform Resource Locator (URL). The segments loaded using data from the MPD are then chosen, extracted, and rendered by DASH. Segments might be gathered in the identical content environment as the MPD or disseminated through a CDN [4]. DASH clients determine the available buffer capacity, bandwidth, and potentially other parameters in the background and choose the next acceptable segment based on this information. If a bandwidth deterioration is shown, for example, clients choose a lower quality, and therefore a smaller segment, to avoid buffer subsidence and allow continued media viewing [6]. Figure 4 illustrates the MPEG-DASH standard's conceptual design.

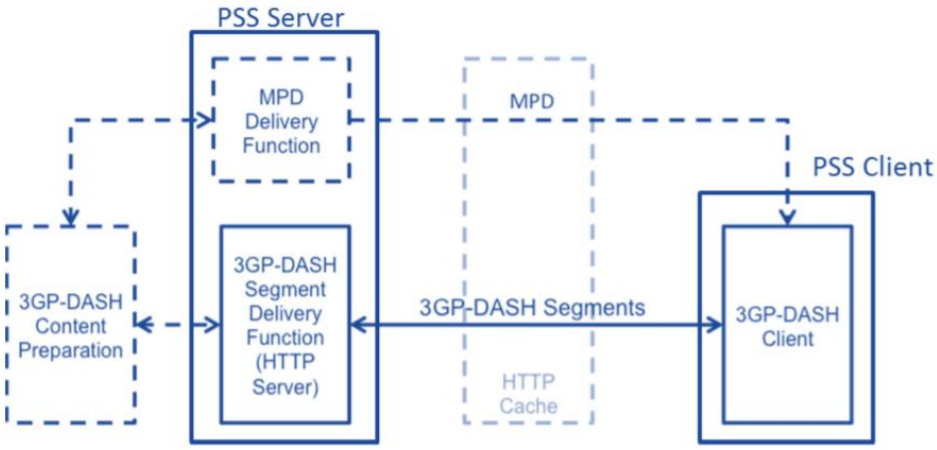


Figure 4. MPEG-DASH standard's conceptual design [10]

The Cisco forecast [2] that users would be producing more than 50 trillion bytes of information until the end of the decade introduces the Yottabyte era. However, there are restrictions to the traditional CDN model. For example, reaching 5G capacity becomes super expensive, and low latency requirements could not be solved. In addition, limitations would prevent implementing real-time interactive video streaming and other new services [2]. Fully optimised 5G claims a new CDN that is as closer to users as possible, able to maintain more capacity per unit, low latency, and dynamic allocation of resources even in the farthest corner of the 5G network achievable by CDN virtualisation. CDN solves object distribution problems for HTTP traffic and CDN in new-generation networks to determine video stream distribution problems the same way [5]. However, there is a considerable distinction between streaming CDN and regular CDN. Streaming applications are commonly bigger than web applications which put pressure on caches.

Moreover, video streaming, except caching, demands much closer collaboration between content makers and the CDN. In addition, servers and clients latency are more significant in video streaming than in the context of HTTP delivery because startup time for video streaming is usually 2 - 5 seconds which is relatively high compared to milliseconds for HTTP. Finally, from an enterprise outlook, a CDN supports several features. For example, removing content from the network as it was placed in a single centralised server, statistics of traffic, and controlling their cost structure by defining the number of streams to operate control access end-users streams [5].

Over-The-Top (OTT) is a digital service that comprises transferring video content peer-to-peer through the Internet without Internet Service Provider (ISP) control and management [6]. The road is a combination of CDN and the Internet in substance where the user equipment connects through Wi-Fi, mobile and fixed network as access points. The principle of operation starts from Online Video Distributor (OVD), the source of uncompressed video. Then, as needed, the video goes through the process of deinterlacing. Next, the critical process is the frame rate, subsampling, and resizing processing to create various levels and encoding using Scalable Video Coding. The last process is redirecting ready video streams to CDN, where users can request online content using any device [6], as demonstrated in figure 5. OTT services are constructed on top of a third-telecommunications party's infrastructure, with no extra network operator approval. Voice, video, messaging, music, games, and other forms based on customer requirements, with OTT streaming services having a huge impact on the telecommunications sector.

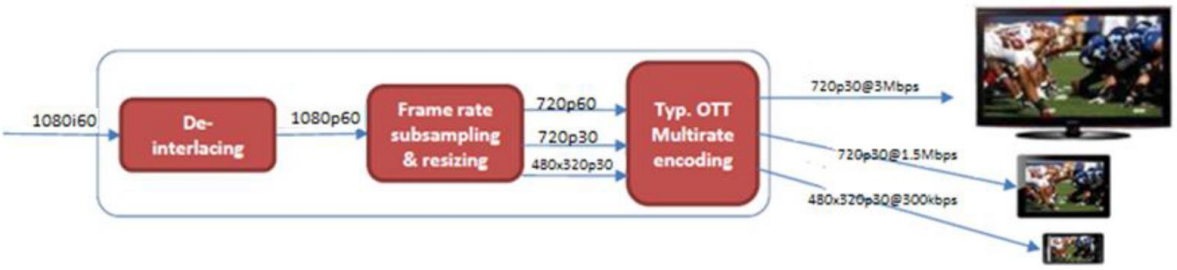


Figure 5. OTT System [6].

Conventional RTP/RTSP video services use a single multimedia server and are sent over UDP, which prevents the video data from being retransmitted. On the other hand, OTT video services utilise the TCP protocol, which offers congestion control and flows control engines to guarantee secure transmission. However, packet loss and retransmission influence the OTT video quality, so the QoE management presents new features [6].

1.3 Service provisioning based on Quality of Experience

Quality of Experience (QoE) refers to the subjective experience that a user has when interacting with applications or services [7]. The Quality of Experience (QoE) is a subjective statistic that evaluates how customers perceive the entire value of a service. HTTP video streaming is affected by stalling, initial delay, adaptation, and other QoE

factors [7]. As a consequence, two types of variables have an impact on QoE: user-related and service-related elements, often known as technical and perceptual aspects. The end-user of the program perceives the perceptual elements accurately, although they are unrelated to technical advancement. The first delay, for example, may be due to technical causes, but the end-user perceives the time spent waiting. As a result, it's critical to examine technological effect variables that influence end-user perceptions. For example, in a video streaming service, the technical impact factor initial delay is often used, such as the time it takes to transmit a specific portion of a video for decoding and playing the video [7].

In reality, the available transmission data rate as well as the encoder configuration define the shortest feasible initial delay. In order to fill the playout buffer with more time, video playback is often delayed. This playout buffer is essential for reducing short-term throughput variations [7]. The amount of initially stored playtime, on the other hand, should strike a balance between the length of the associated delay (more buffered playtime = longer initial delay) and the risk of buffer depletion. Stalling is a phrase that describes a longer buffered playtime that is less susceptible to short-term throughput changes. In video streaming, the phrase "event stalling" refers to video playback being stopped due to a buffer underrun. This occurs when the video bitrate exceeds the video streaming throughput. Finally, the buffer is full, thus the movie can't be played endlessly until additional video data is gathered. Traditional HTTP video streaming is primitive, and it can only balance playout buffer size and stalling time in response to network variations. In contrast, HTTP adaptive streaming is more flexible and may balance video stream delivery with current network conditions, decreasing stalling time [7].

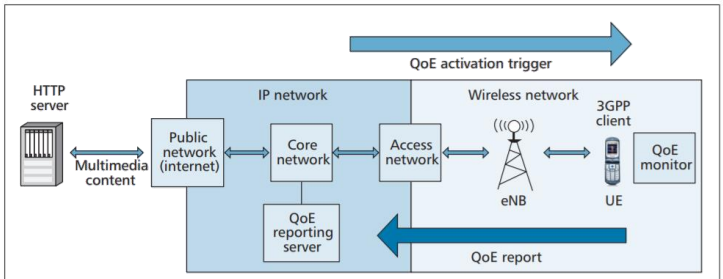
The present methodology for assessing QoE may be classified as a subjective assessment method, an objective assessment method, or a mixture, depending on whether the user is actively engaged in the evaluation and whether QoE and its model of correlation of affecting factors. Subjective QoE assessment methods are the most accurate and direct means of evaluating QoE since they are acquired when users are asked to rate the quality of a service as they see it, but they are cumbersome and expensive to manage [8]. The objective assessment, on the other hand, assesses the user's impression of service quality using numerical quality measures. In this way, the assessment process may be repeated and automated. Application service

management, on the other hand, is required to integrate input and output signals, which is challenging for network operators to do for OTT services and overlooks user concerns. Artificial intelligence, statistics, and other disciplines are utilized as theoretical support in a pseudo-subjective evaluation approach that combines subjective and objective assessment techniques, allows for real-time application, and is very accurate. More advanced model learning, on the other hand, requires a large amount of data [8].

Average throughput, initial playout delay, buffer level, playlist, MPD information, HTTP request-response transactions, and representation transition events are all included in the 3GPP DASH standard [10]. HTTP request and response transaction metric substantially result from each HTTP request and appropriate response. The client-side determines and reports:

- Request type, for example, media segment, MPD, initialisation segment.
- Time of when HTTP request was made and the appropriate response.
- HTTP header content in the byte-range specification.
- TCP connection identifier
- Successful throughput trace values.

More capacity metrics such as MPD fetch duration, initial segment, and media segments can also be obtained from the HTTP request and response transactions [10].



3GPP QoE metric	Progressive Download	DASH / HAS
HTTP request/response transactions	Yes	Yes
Representation switch events	No	Yes
Average throughput	Yes	Yes
Initial playout delay	Yes	Yes
Buffer level	Yes	Yes
Play list	Yes	Yes
MPD information	No	Yes

Figure 6. 3GPP DASH and progressive download QoE measurements and reporting framework [9].

The representation switch events metric is used to inform switch events by computing the time. A representation changeover event is triggered by the client signals for switching from one representation to another. Within each representation switch event, the client gives the ID for the new representation, the time of the switch event when the client sends the first HTTP request for the new representation, and the media time of the first media sample played from the new representation [10]. The entire quantity of content bytes, such as HTTP responses, activity time - at least one "GET" request has yet to be completed, measurement period duration, inactivity type, and TCP channel access are all included in the average throughput statistic. The time between when the client requests the first media segment and when the content is received from the client buffer is measured by the first playback delay metric [10]. Finally, during playback, the metric buffer level is a computation of the buffer fill level. The media data is provided starting with the current playing time and the buffer level measurement time. The MPD metric is used to hold media presentation information so that servers without direct access to the MPD may learn about the media. The client may use this measure to provide media display attributes including resolution, bit rate, and quality rating, as well as media information like profile and level [10]. The MPEG-DASH Media Presentation Description (MPD) is an XML document that includes information about media segments, their connections, and the knowledge needed to choose between them, as well as any additional metadata that clients may need. MPD has the following structure [10]:

- The top-level MPD element's periods define material with a start and end time. For scenes or chapters, or to divide advertising from program material, several periods may be utilized.
- Views enable the same information to be encoded in various ways in distinct adaption sets. In most instances, opinions will be provided in a variety of screen sizes and bandwidths, allowing customers to seek the best quality material possible without having to wait for buffering or wasting bandwidth.
- Subviews in a view include information related to just one media stream. They also provide you the information you need to get a single stream from a multiplexed container or a lower-quality version of a stream.
- Media segments are the media files that the DASH client plays in order, as if they were one and the same. For a single-segment view, a list of segments

(SegmentList), or a template, the location of media segments may be specified using BaseURL (SegmentTemplate).

- There are two kinds of index segments: one for the entire view (which is always a distinct file) and one for each media segment (which may be a range of bytes in the same file as the media segment).

The effect of re-buffering on QoE is considerable. Re-buffering time, re-buffering position, and the number of re-buffering occurrences are the metrics used in this thesis (NR). As a result, the playback indicator (PI) metric is a continuous-time binary variable that shows the current playback status, with 1 indicating re-buffering and 0 indicating normal playing. In addition, since re-buffering events irritate users, the total number of re-buffering events from the beginning to the conclusion of the session is counted. Variables in memory may have an effect on QoE. Consider the recency effect, which occurs when recent events have an impact on quality of life. As a consequence, the recency effect for QoE is calculated using the time elapsed (TE) measure since the last video degradation, bitrate change, or re-buffering.

Additionally, more QoE metrics influence the adaptive video streaming in the network.

- Average resolution refers to the average of all segments selected by the media player, where each component can take on resolution values from this set {2160, 1440, 1080, 720, 480, 360}.
- Toggle sum refers to the sum of the resolution changes that occurred during video playback.
- Average elevation refers to the average of the hops between the previous and current segments across all segments selected by VLC, where each component can take on hills from the set [0.6]. So, for example, if the first segment is 480p and the second is 1440p, the height or jumps are 3.
- Average video bitrate (Mbps) refers to the average bitrate across all segments selected by the media player, where the size of each component is divided by 5 seconds, which is its duration.
- Network usage time in seconds refers to the duration of video playback from when the first segment is delivered to when the last element is given.

- R1 Network Average Throughput (Mbps) refers to the size of all segments (cached and uncached) that the media player has fetched during network usage.
- R2 Network Average Throughput (Mbps) refers to the size of all uncached segments over the time the network is in use.

In networks and telecommunications, the Average Opinion Score (MOS) for latency measures the overall quality of an event, system, or experience. It is a crucial QoE measure that is expressed as a single rational number, usually ranging from 1 to 5, with 1 representing the lowest perceived quality and 5 representing the highest. $\|MOS\| = 3.5 * e^{(-(0.15M+0.19)N)} + 1.5$, where *M* is the average stop length and *N* is the number of pauses, and where M is the average stop duration and N is the number of pauses. MOS may be utilized in a number of ways, including adaptive streaming and progressive download. However, audio coding and video quality deterioration owing to encoding, spatial scaling or video frame rate changes, and delivery quality degradation all affect the end-perception user's of quality. Furthermore, because of the initial latency in loading, halting, and adjusting to the environment [24].

1.4 Machine Learning

Machine Learning (ML) is a term that refers to a set of algorithms that are based on prior knowledge. As can be seen, ML has started to replace conventional optimisation techniques in many domains due to the flexibility of ML models to integrate new constraints and inputs without having to start from scratch and solve mathematically complex equations. As with today's computer systems, machine learning models adapt quickly to changing circumstances. User location, connection information, cell load, local traffic patterns, application types, and dedicated bandwidth are all examples of contextual data that may be utilized to benefit from machine learning at edge devices. Additionally, doing machine learning activities at the edge may decrease the load on the main network. To get the most out of MEC and ML's partnership, ML models should be designed to utilize the fewest resources possible while yet providing accurate and relevant results as they scale over massive communication networks. Due to memory and device power constraints, ML cannot currently offer high inference in MEC in contrast to industrial data centers. However, there are two main benefits of utilizing machine learning in MEC:

- For particular tasks, effective ML models need less energy, memory, or training time.
- Apportioned ML models that distribute learning and inference assignments among big data centers and smaller peripherals for improved efficiency and parallel processing.

Machine learning models that are state-of-the-art have been developed to function on edge devices using simple models that need minimal computational resources and may be utilized on IoT devices [16]. When fewer granular answers are required, decreasing the amount of the model input for classification applications, on the other hand, may improve MEC learning. Additionally, by adding many exit points in models to get outcomes, as well as establishing human-machine partnerships utilizing neural networks and designing new efficient ML architectures, computational costs may be lowered [17]. Model replication, on the other hand, is just the first step in achieving efficient ML in MEC.

1.1.1. Deep Learning

Deep Learning (DL), a subset of machine learning, has attracted a lot of attention in recent years for computer vision, new optimal gaming strategies, and other uses, all without the need for costly human development. DL, on the other hand, uses neural networks to automate feature extraction from huge data sets for later use in categorization, decision-making, or data creation [18]. As seen in Figure 11, several conventional deep neural networks (DNN) are presented. These models are split into three groups based on the training methods used: supervised, unsupervised, and reinforced.

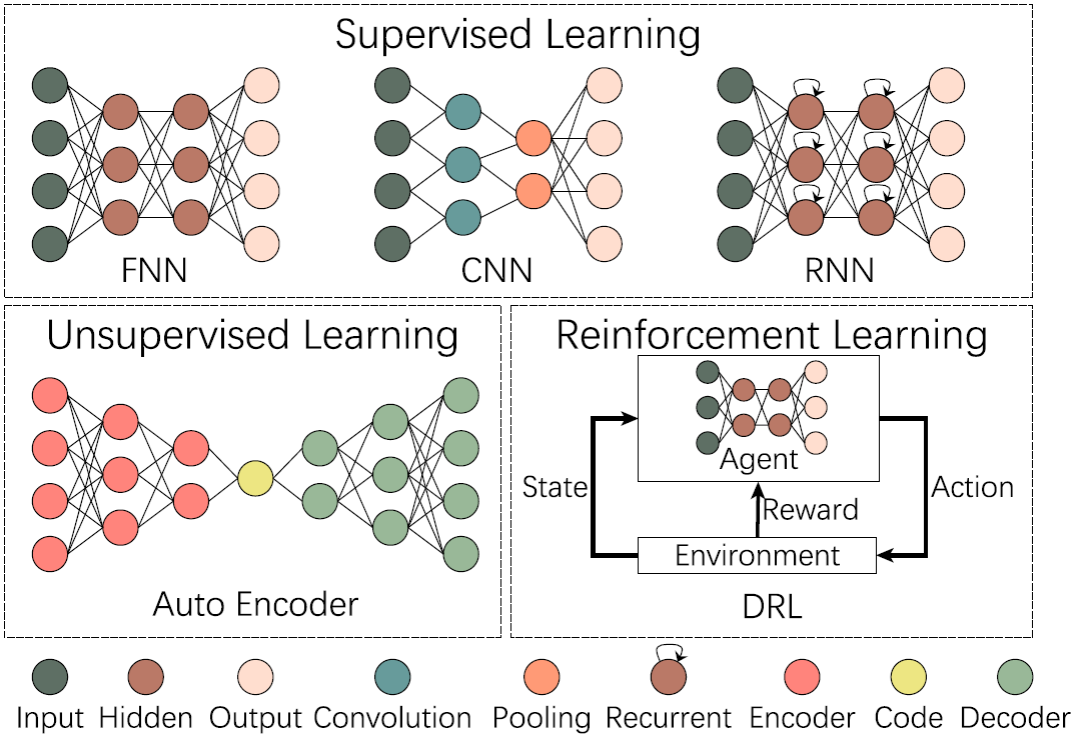


Figure 7. Common DNN structures [18].

In DNNs, which are a step ahead from conventional perceptron systems, hidden layers and internal functions are utilized to approximate non-linear connections between input and output. To minimize a cost function, all DNN models utilize gradient descent. This technique is known as "back-propagation" in optimisation and training, for example, mean square error and maximum likelihood. In 5G networks, deep learning lowers cost processes and minimizes operational expenses, latency, and downtime. Deep Learning is also used in 5G networks for traffic categorization, routing choices, and network security [17].

1.1.2. Feed-forward Neural Network

A feed-forward neural network (FNN) is a kind of deep neural network (DNN) in which neurons do not cycle and information is sent directly. According to the Universal Approximation Theorem [19], FNN can estimate any closed and constrained function with enough neurons in a hidden layer. The hidden layer extracts the properties of the input vector, which is then passed to the output layer, which acts as a classifier or regression. For example, the classifier $y = f^*(x)$, connects input x to category y. By establishing the mapping $f(x; \theta)$, the FNN learns the values of the parameters that lead to the best approximation of the function. The data is sent from *x* through the intermediate computations needed to compute *f*, and then to the final output *y*. As a

consequence, no feedback loops exist in which the output of the model feeds back on itself.

1.1.3. Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a kind of deep neural network (DNN) used to evaluate large, high-resolution images [18]. CNNs, depending on their location, use connections in nearby data, such as pixels in a picture. CNNs use mathematical "convolutions," linear processes that weighted average neighboring samples, and "pooling," which usually uses a maximum combination or similar function calculated on the target [26], to totalize data across a region. CNN was used in 5G networks to predict data flows between base stations and detect objects, allowing for the development of 5G-enabled industrial robots. To overcome the above-mentioned shortcomings of FNN, CNN uses convolution and concatenation techniques. The first is used to downsample and the second is used to apply slide convolutional filters on the input vector. It typically happens when a maximum or average concatenation occurs. CNN is renowned for utilizing smaller convolutional filters and deeper layers. As a result, the system develops into a fully convolutional network, with a decreasing percentage of pooling layers compared to fully connected layers. CNN is used in image recognition, video analysis, natural language processing, and other applications. One of the drawbacks of FNN, which includes CNN, is that the output is solely determined by the input vectors. As a consequence, completing several activities in a row is challenging [17].

1.1.4. Recurrent Neural Network

The recurrent neural network covers DNN models for estimating functions for processing time sequences (RNN). The previous time step's output has an impact on the network's solution for the next time step [18]. As a consequence, in addition to the actual model inputs, RNNs need the use of "memory" to recover data from earlier time steps. Gradient descent learning on RNNs produced explosive gradients, which were corrected using new machine learning models called Long-term Short-term Memory (LSTM) models with extra information flow structures called "gates." LSTM models have shown to be useful and accurate in solving traffic forecasting and mobility problems in communication networks. RNNs, on the other hand, may suffer from long-term dependency issues [19], such as gradient explosion and disappearance.

The RNN has extra parameters for training because of the double weights. A simple RNN, as previously mentioned, has more parameters in the learning phase, making learning exact weights and input weights more difficult [19]. The Echo-State Network (ESN) adds these two weights and only knows the output weights to solve these problems. The buried layer was renamed "reservoir" in ESN, and neurons are loosely connected with weights given at random. Because the consequences of repeated acts are continuous, information about precious events builds up in a constant weight reservoir, akin to a voice echo. In reality, employing a closed RNN, such as LSTM [18], to deal with long-term dependencies is a novel and effective approach. LSTM is a kind of RNN that was created to solve issues with long-term dependencies and RNN. As with any RNN with repeated modules of a neural network, LSTM has a chain structure. By default, RNN modules consist of only the *tanh* layer. However, LSTM has four layers: *three sigmoid and one tanh*. LSTM has a memory cell and three gates: *forget, input and output*. "Forget gate" is where information goes through LSTM, the sigmoid function (1) decides to keep or not the data by considering the previous out but h_{t-1} and x_t, as shown in figure 12, the decision presented as 1 or 0, respectively [19].

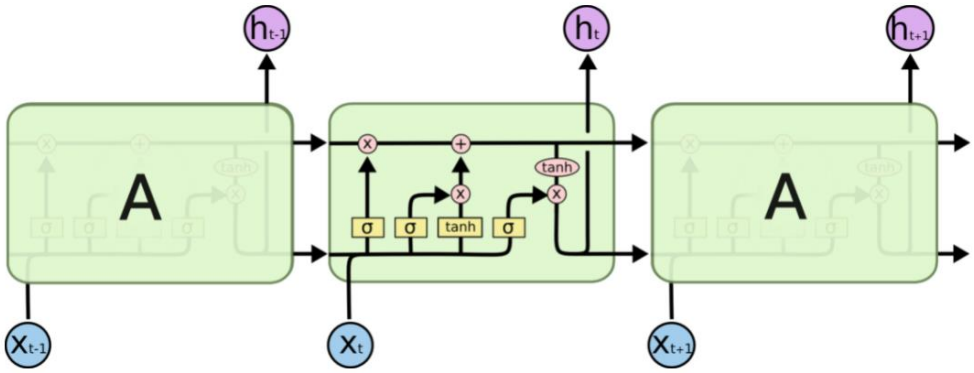


Figure 8. LSTM chain structure [19].

"Input gate" combining sigmoid (2) and tanh (3) functions in the LSTM to update the cell state, where tanh creates a vector of new $\tilde{C}_t$ values. By multiplying the old state C_{t-1} by forgetting things f_t and adding $i_t * \tilde{C}_t$ to update new state values (4). The "Output gate" moves the state value between -1 and 1 using the tanh function layer (6) and multiplies it by the output of the sigmoid function (5), after which a portion of the output can be taken as determined.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (3)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

To sum up, "Input gate" is chargeable for storing and controlling content in the cell, "Forget gate" for rebooting the cell by demand, and "Output gate" defines the current state of the cell and content which is sent to the next level. The state declares short-term memory, whereas the cell supports a long memory range. These parts together in one unit does LSTM network [19]. Additionally, plenty of machine learning tools exist to utilise all the above-mentioned neural networks, and section 1.4.5 describes these tools more precisely.

1.1.5. Machine Learning tools

Weka. The University of Waikato in New Zealand created WEKA (Waikato Environment for Knowledge Analysis), a freely accessible machine learning software program. Weka is a data analysis platform with easy graphical user interfaces, algorithms, visualization tools, and prediction models [20]. It is developed in Java and operates on virtually any platform. As a result, users may call and apply algorithms directly to the dataset or from Java code created by hand.

Weka's initial version was non-Java and based on the programming language C, and it was primarily used to implement third-party modeling methods. The initial version is mostly used for data analysis, but the complete Java-based version of Weka is now used in a variety of areas, particularly for educational and research purposes [20]. The Weka tool has a number of benefits, including mobility, a large number of data pre-processing and modeling methods, and an easy-to-use user interface. Typical tasks include data pre-processing, clustering, classification, regression, visualization, and feature selection [20]. Weka methods are based on estimating data from a single file or a connection, with each data point described by a set of attributes, usually numeric

or nominal. Weka also enables you to utilize SQL databases and modify the results of a database query using Java Database Connectivity. Another significant area that is not covered by the Weka methodologies and approaches is sequence modeling [20].

Scikit-learn. Python's programming language has been determined as one of the most well-known languages for scientific calculations due to the high-level interactive pattern and growing ecosystem of scientific libraries, which are an attractive option for research data analysis and algorithmic engineering [21]. Scikit-learn leverages the above-mentioned affluent framework to maintain state-of-the-art realisations of many popular machine learning algorithms while supporting a user-friendly interface regardfully when coupled with Python language. Thus, Scikit-learn responds to the rising demand for statistical data analysis by non-technicians in the software and web industry and areas beyond computer science such as biology or physics. Scikit-learn operates with "Numpy" and "Scipy" python libraries, focusing on imperative programming. In addition, although the package is created in Python, it includes C++ libraries such as LibSVM and LibLinear, which maintain standard realisation of SVMs and generalised linear models with compatible licenses [21].

Furthermore, Scikit-learn can estimate the productivity of the evaluator or select parameters utilising cross-validation, distributing the calculations across multiple cores if necessary. It is experienced by comprising an evaluator in the GridSearchCV object, where "CV" stands for "cross-validation". At the call of function "fit", it fetches features in a given feature grid, maximising the estimate, the forecast, estimation, or transformation is then deputed to the configured estimator [21]. Therefore, objects can be utilised transparently as any other evaluator. However, cross-validation can be more effective for some evaluators by applying specific attributes, such as warm restarts or normalisation paths. Finally, a Pipeline object can unite multiple transformers and an evaluator to build a joint evaluator, for instance, to apply downsizing before fitting. Despite the fact that Scikit-learn is mainly built in a high-level language, it has been carefully designed to maximize computing efficiency [21].

TensorFlow. TensorFlow tool [22] is a machine learning system that works highly and in heterogeneous conditions. To conduct computation, general state, and operations that alter that state, TensorFlow uses data flow graphs. It displays the nodes of a data flow graph across many computers in a cluster, as well as inside a single machine spanning multiple processing devices such as multi-core CPUs, general-purpose

GPUs, and specialized GPUs. With an emphasis on deep neural network training and analysis, TensorFlow offers a broad variety of applications. Furthermore, TensorFlow [22], an open-source framework extensively utilized in machine learning research, is used in the production of many Google services. Figure 13 shows a TensorFlow data flow graph for a training pipeline, which includes subgraphs for reading input, pre-processing, training, and checking the checkpoint state. The data flow graph explicitly signifies the interactions between sub-calculations, becoming it easy to run autonomous calculations in parallel and split measures throughout multiple devices.

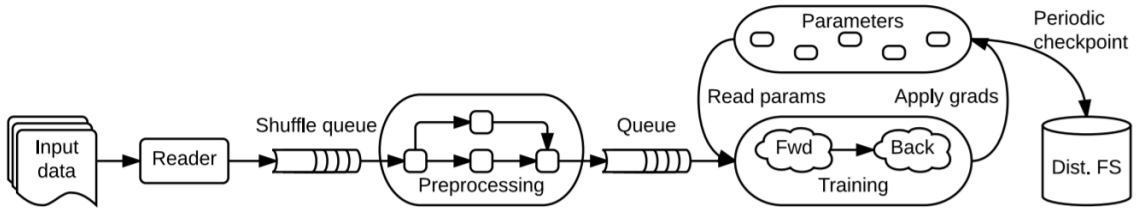


Figure 9. A training pipeline's TensorFlow data flow graph, including subgraphs for insight data-in, pre-processing, training, and checkpoint state [22].

Each vertex introduces a unit of local processing, and each edge provides an exit or entrance to a vertex in a TensorFlow graph. As a result, TensorFlow refers to operations at vertices as "Operations" and flow values along edges as "Tensors" [22]. Tensors are n-dimensional arrays containing a limited number of basic types, such as Int32, float32, and others, as units. In many machine learning methods, tensors naturally execute the input and output data of the typical mathematical processes. For adding sparse data, TensorFlow offers two options: encode the data into variable-length string components of a dense tensor or use a tuple of dense tensors. In coordinate list format, an n-D sparse tensor with *m* nonzero components, for example, may be represented as a *m* x *n* matrix of coordinates and a vector of values of length *m*.

Furthermore, a tensor's shape may vary in one or more dimensions, allowing sparse tensors with various numbers of units to be shown [22]. A named "type" for an operation, such as Const, MatMul, or Assign, may contain zero or more compile-time characteristics that specify its behavior. At compile time, a process may be polymorphic and variable, with its characteristics defining the common types of its inputs and outputs. The "AddN" operation, for example, adds several tensors of the same unit type, and it has a type signature defined by a feature type T and an integer feature N. TensorFlow has conditional and iterative control flow, as well as

sophisticated machine learning techniques. A recurrent neural network (RNN) like LSTM, for example, can predict based on sequential data. TensorFlow is used by Google's Neural Machine Translation engine to train deep LSTM, which gives the best results for many translation tasks [22].

Google Colaboratory. Jupyter Notebooks, the technology underlying Google Colaboratory, were launched before Google Colaboratory. Jupyter is an open-source browser-based program that integrates interpreted languages, libraries, and visualization tools. The Jupyter notebook may be used locally as well as on the cloud. The output is included, and the page is made up of several cells, each of which includes a scripting language or markdown code. In many works, text, tables, charts, and pictures may be found. Because experiments and results are published separately, this technique makes scientific papers easier to share and replicate [23]. The Google Colaboratory is a project that aims to spread machine learning research and education. Collaboration notebooks are based on Jupyter and operate as a shared Google Docs document, enabling many users to collaborate on the same project. Python runtimes pre-configured with essential machine learning and AI libraries such as TensorFlow, Matplotlib, and Keras are included in Collaboratory. The runtime virtual computer is deactivated after a season, and all user data and settings are deleted. The notebook, on the other hand, is kept, and data on the virtual machine's hard drive may be moved to the user's Google Drive account. Finally, utilizing the tools described earlier [23], this Colab service has a GPU-accelerated runtime that can be fully modified.

1.5 Problem Statement

Streaming video services are often impacted by changing network conditions, which may result in corrupted incidents and a decrease in QoE. As a consequence, developing QoE models that can accurately and quickly predict a customer's QoE in live time may help applications that prioritize Quality of Experience preservation. Continuous QoE assessment, on the other hand, is challenging because it must account for complex temporal correlations in serial QoE data as well as non-linear relationships between factors that influence QoE such as video quality, bit rate switching, and re-buffering.

The main goal of QoE forecasting study is to investigate a framework that includes a variety of activities. To begin, propose a new model that may improve the accuracy of the forecasting model while simultaneously lowering its complexity. Second, in terms of multiplication by output, memory usage, and output time, to make the proposed model easier to execute on real devices and edge devices. To provide an appropriate level of service, end-user QoE should be reviewed on a regular basis. By constantly monitoring user QoE and improving network resource usage and video streaming, network operators may improve QoE. QoE is influenced not just by video quality, but also by a mix of factors such as rate adaptation and re-buffering activities that happen at various times throughout a video session. The user's QoE is affected by rate adaptation, which causes video quality to vary over time. The data rate of a video user over a wireless network is constantly changing due to channel oscillations, user mobility, resource sharing, and other variables.

2 BACKGROUND

2.1. QoE prediction models

2.1.1. Streaming Video QoE Modelling and Prediction: A Long Short-Term Memory Approach.

In study [11], an LSTM-based method for QoE prediction was presented, with the authors using the LIVE Netflix and LFOVIA1 databases for Quality of Experience as starting data. The continuous Quality of Experience is a random process with non-Markov time dynamics that is non-linear. The authors utilize LSTM, a kind of RNN that has been shown to successfully represent long term short term relationship in serial data, to capture these dynamics. LSTMs memory regions and gating structures are positioned in such a way that they can effectively capture complicated relationships and practically avoid the vanishing gradient issue that plagues conventional RNNs. The actual and projected QoE at time *t* are represented by *y(t)* and *y(t)*, respectively, in the system model of [11]. Let $x(t) \in \mathbb{R}\mathbb{m} \geq 0$ handle the feature set that captures non-negative real-number values in an m-dimensional space. The QoE impact factors that govern the development of QoE are delegated by the feature set *x(t)*. So, although the time-indexed feature vector $x(t) = [x_1(t), x_2(t), \dots, x_m(t)]$ may predict the current QoE *y(t)* at every given time instant *t*, the actual QoE *y(t)* is non-Markovian:

$$p\big(y(t)\big|y(t-1), y(t-2), \dots, y(1)\big) \neq p\big(y(t)\big|y(t-1)\big), \tag{1}$$

The conditional probability $p\big(y(t)\big|y(t-1), y(t-2), \dots, y(1)\big)$ points to the QoE takes into account temporal connections at a higher level. These relationships are complicated, and capturing them effectively with a single LSTM is impossible. As a result, as shown in figure 7, a framework of LSTMs was created to train these subjections that are part of the QoE process..

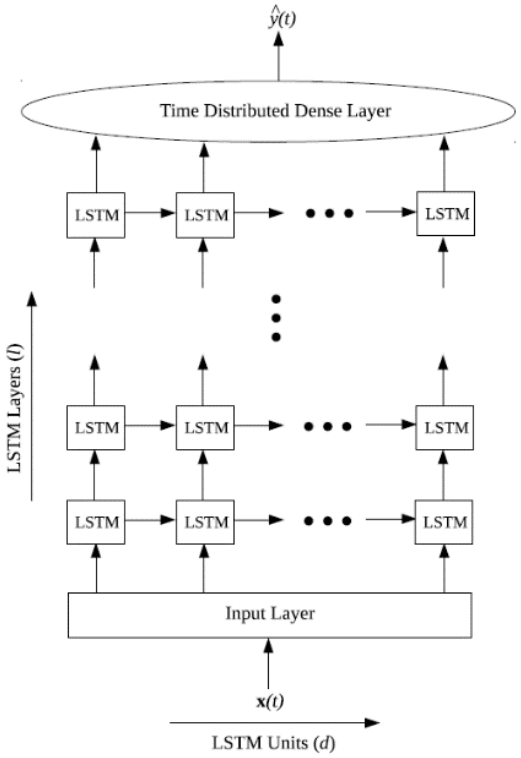


Figure 10. QoE prediction using an LSTM network [11].

The provided network is made up of a series of LSTM modules that are organized into layers. The LSTM network is abbreviated as $LSTM_{l,d}$, with *l* denoting the number of layers and *d* denoting the each layer's amount of units. The design choices to be controlled by the parameters *l* and *d* are dependent on the complexity of the connections and the nature of the underlying process. The LSTM network continually computes the estimated QoE $\hat{y}(t)$ at each time *t* using the input attributes *x(t)*. Each LSTM unit maintains track of a random process by maintaining a hidden state within each cell, with state transitions controlled by input functions *x(t)*. The collection of states of LSTM cells in the network is denoted by *c(t)*. To investigate the underlying complicated state transition control allocation and forecast the QoE at each point in time, LSTMs are used in the following way.

$$p\big(y(t)\big|y(t-1), y(t-2), \dots, y(1)\big) = p\left(y(t); g\big(\boldsymbol{c}(t)\big)\right), \tag{2}$$

where $g(.)$ is a differentiable function that translates the parameters of the main unknown QoE allocation to the *c(t)* LSTM. $LSTM_{l,d}$ has two functions: $LSTM^{o}_{l,d}$, which predicts output QoE, and $LSTM^{c}_{l,d}$, which updates the cell's state. The expression determines the expected QoE $\hat{y}(t)$.

$$\hat{y}(t) = LSTM_{l,d}^{o}\big(x(t), \boldsymbol{c}(t-1)\big), \tag{3}$$

$$\boldsymbol{c}(t) = LSTM_{l,d}^{c}\big(\boldsymbol{c}(1:t-1), \hat{y}(1:t-1)\big), \forall t > 1, \tag{4}$$

Using the LSTM network function $LSTM_{l,d}^{c}$, condition of the cell *c(t)* is a function that is deterministic of previous QoE $\hat{y}(1:t-1)$ and previous cell conditions. The state vector *c(t)* enables the LSTM to represent sequential data by monitoring complicated timing throughout the QoE process. The predicted QoE $\hat{y}(t)$ is calculated based on the current input function *x(t)* and the cell's condiotn prior to the renew $c(t-1)$, as described in (3). Since LSTMs are fundamentally non-linear in design, the Nonlinearities in the estimation of User experience are also taken into account. The selection of input features *x(t)* is important for continuous QoE prediction, according to the authors of [11]. The input characteristics must be chosen in such a way that they effectively take and combine the different influences that drive the development of QoE across LSTM states. Short Time Subjective Quality (STSQ), Playback Indicator (PI), and Time elapsed since last re-buffering were chosen as the main three characteristics in study [11]. (TR). The perceived quality of the video clip that is being shown to the user is cited by STSQ. Standard Video Quality Assessment (VQA) measures like STRRED and MS-SSIM may be used to assess STSQ. The video is presently playing or re-buffering, as indicated by a PI binary pointer variable. Because the recurrence of re-buffering events has a significant impact on a user's QoE, TR is a variable that tracks the time since the previous re-buffering event. Linear Correlation Coefficient (LCC), Spearman Rank Order Correlation Coefficient (SROCCC), Outage Rate (OR), and Normalized Root Mean Squared Error were also employed as metrics for evaluating QoE forecasts (NRMSE). The correlation between anticipated and accurate QoE estimations is measured using LCC and SROCC. NRMSE and OR are indicators of how close predicted and objective scores are.

2.1.2. Network Traffic Type-Based Quality of Experience (QoE) Assessment for Universal Services.

The authors of [12] suggested a method for calculating QoE scores, which is shown in figure 11. The task of [12] is split into three major steps: extraction of QoE measurements, categorization of flow traffic, and quantification of QoE. Time delay,

loss rate, and throughput rate are all QoE measures that are utilized in the "QoE Quantization Model" to assess the QoE for a network flow with a known traffic type.

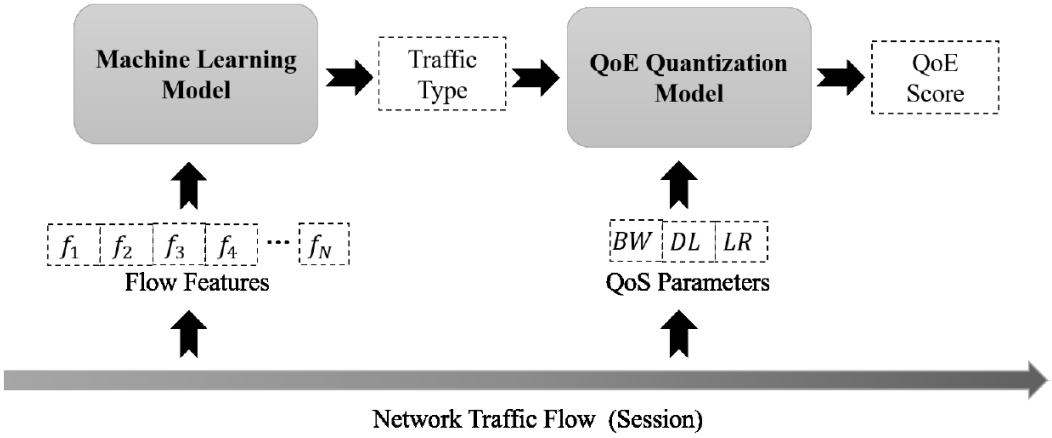


Figure 11. System model overview of work [12].

Metrics in work [12] are given as the loss rate, flow throughput rate, and packet delay defined as DL, LR, and TH, respectively. To address the drawback of the linear mapping function $f_k(.)$, the authors separately consider the connections between QoS and network parameters. Above mentioned parameters of the network were calculated as

$$QoS(DL) = a * \exp(b * DL) + \varphi_1, \tag{5}$$

$$QoS(LR) = \frac{c}{LR+d} + \varphi_2, \tag{6}$$

$$QoS(TH) = p * \log(TH) + \varphi_3, \tag{7}$$

then combining these there influencing factors, the authors get the QoS expression

$$\begin{aligned} QoS &= QoS(DL) + QoS(LR) + QoS(TH) \\ &= a * \exp(b * DL) + \frac{c}{LR+d} + p * \log(TH) + q, \end{aligned} \tag{8}$$

where $q = \varphi_1 + \varphi_2 + \varphi_3$ and arguments *a, b, c, d, p, q* need to be indicated for many kinds of network services. The link between QoS and QoE in work [12] is given as

$$\frac{\partial QoE}{\partial QoS} = -(QoE - \gamma), \tag{9}$$

$$QoE = (\alpha * \exp(-\beta * QoS) + \gamma), \tag{10}$$

When the rounding function, a > 0, alters according on the kind of network traffic, such as video-on-demand, audio-on-demand, live video, video streaming, and so on. Authors built up mathematical equations connecting the QoE and QoS values with DL, LR, and TH using formulae (8) and (10). In Figure 12, the Precision, Recall Rate, and F1 Score of all network service types were likewise analyzed and found as follows:

$$Precision = \frac{TP}{TP+FP}, \tag{11}$$

$$Recall = \frac{TP}{TP+FN}, \tag{12}$$

$$F1Score = \frac{2*Precision*Recall}{Precision+Recall}, \tag{13}$$

The number of instances correctly classified as X is TP (True Positive), the number of instances incorrectly classified as X is FP (False Positive), and the number of instances incorrectly classified as Not-X is FN (False Negative). The percentage of genuine positive samples identified as positive is referred to as precision. The recall rate is the percentage of properly recognized positive models to the total number of positive instances. The F1 score is the harmonic mean of the accuracy and recall rates, and it is only high when both metrics are high. On a test dataset, the authors claim that their method can obtain an average competitive classification accuracy of almost 90% for virtually all kinds of network traffic.

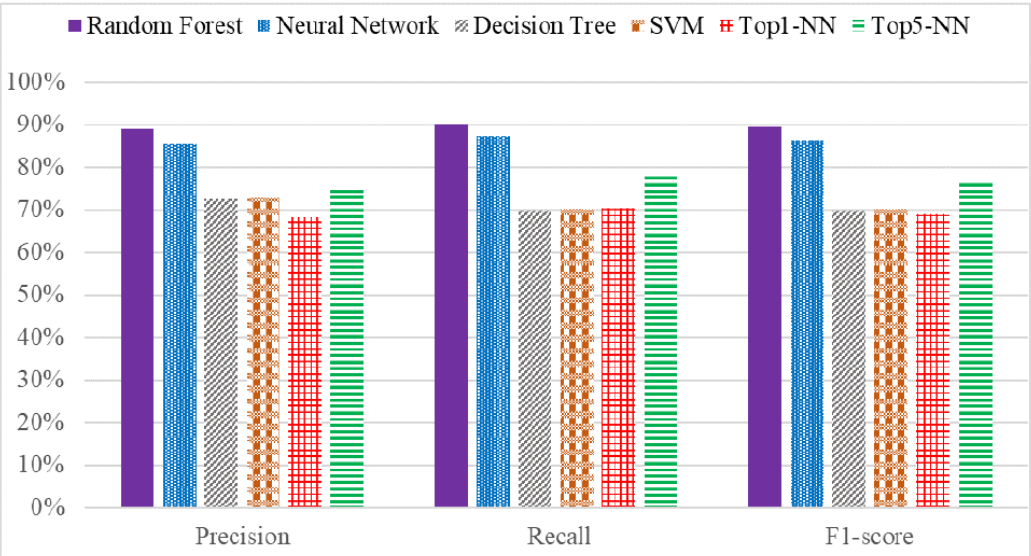


Figure 12. The efficiency of traffic classification for different NN [12].

The normalised root-mean-square error (RMSE) is often used to quantify the error between the model projected MOS score and the ground truth RMSE is computed in work [12] for evaluating the QoE model.

$$NRMSE = \frac{1}{N}\sqrt{\sum_{q=1}^{N}(QoE_p - QoE_T)^2}, \quad (14)$$

where N defines the number of test samples, QoE_p and QoE_T are the QoE predicted by the model and the actual QoE value, respectively. The results of [12] show that, depending on the model settings, the QoE responsiveness to QoS measures changes with the kinds of network traffics.

2.1.3. Convolutional Neural Networks for Continuous QoE Prediction in Video Streaming Services.

Temporal Convolutional Network (TCN), a kind of Convolutional Neural Network (CNN), has been proposed by way of promising alternate approach for serial modeling issues in a recent paper [14]. TCN makes use of extended causal convolutions to uncover temporal relationships in sequential data in a strong manner. TCN computations, unlike LSTM, may be run in parallel, preserving computational and simulation advantages. In a broad variety of sequence modeling problems, TCN outperforms conventional recurrent architectures, such as LSTMs and Gated Recurrent Units (GRU), in realistic deployments. Recognizing TCN's immense potential, [14] article maintained a refined TCN-based model, QoE-CNN, for constantly forecasting QoE across various devices for watching. Let *x(t)* be a vector of input features at time *t* in a *T*-second session of streaming. Let $y_t \; and \; \hat{y}_t$, respectively, represent subjective and expected QoE at time *t*. For example, the following non-linear function may be used to constantly forecast personal QoE at any time *t*:

$$\hat{y}_t = g(x_t, x_{t-1}, \dots, x_{t-r+1}), \quad (15)$$

The suggested model consists of one layer of causal convolution and a stack of extended causal convolutions, where *r* is the number of input delays. TCN, on the other hand, is made up of just a few prolonged causal convolutions. As shown in figure 13, a causal convolution layer is initially supplied with input time-series and the first residual block.

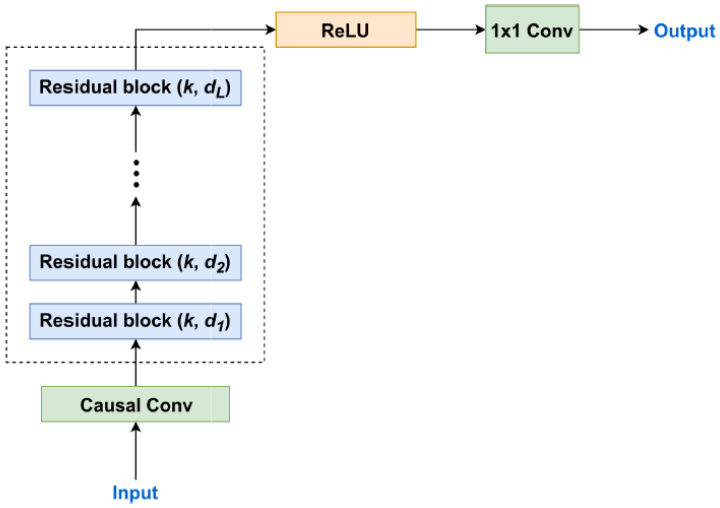


Figure 13. CNN-QoE model design [14].

The causal convolutional layer can extract local features of adjacent time steps in serial QoE data. The extended causal convolutional layers are then used to extract global characteristics between time steps separated by a significant period of time. These layers let the model learn the most relevant features of the time series input, improving accuracy. The activation function is crucial in enabling the model to detect the input data's non-linear nature. Vanishing and increasing gradient g are the most challenging issues that prevent the network from learning the optimal function *g(.)* when training a DL model. The supplied CNN-QoE model replaces these levels with SeLU to take use of SeLU's benefits and simplify the residual block. SeLU is a unit variance, zero mean self-normalizing activation function. It propagates over several layers during network training, guaranteeing that it is unaffected by fading and increasing gradient issues.

$$SeLU(x) = \lambda \begin{pmatrix} x, & if\ x > 0 \\ \alpha \exp(x) - \alpha & if\ x \leq 0 \end{pmatrix} \quad (16)$$

Video quality, also called as brief subjective quality, has a major influence on video streaming users (STSQ). STSQ, which may be predicted using any metric for reliable video quality assessment, defines the visual quality of the video presented to the viewer (VQA). STRRED (space-time reduced reference entropy differences), MS-SSIM (multiscale structural similarity), PSNR (peak signal-to-noise ratio), and others are only a few examples. Work [14] indicates that STRRED is a robust and high-performing VQA model when tested on a broad variety of video quality datasets,

resolutions, and device types. As a consequence, the STSQ is calculated using STRRED.

The quality of the user's experience suffers as a result of re-buffering. As a consequence, data on re-buffering must be reviewed, including re-buffering length, re-buffering location, and the number of re-buffering occurrences. As a result, two re-buffering-related inputs are used in article [14]. To begin, the play indicator (PI) is a continuous-time binary variable that shows the current play state, with 1 representing re-buffering and 0 representing regular play. Second, because user annoyance grows every time a re-buffering event occurs, the number of re-buffering (NR) occurrences from the beginning to the current point in the session is counted.

Memory concerns may affect a user's quality of life. When a video is more current, the novelty effect, for example, has a larger impact on its perceived quality. Capture, on the other hand, is the relationship between the novelty effect and the user's quality of experience, i.e., the length of time since the last video degradation, including such bitrate switching or re-buffering. Pearson's correlation coefficient (PCC), Spearman's rank-order correlation coefficient (SROCC), and root mean square error are three metrics used to assess the accuracy of QoE prediction (RMSE). SROCC measures the monotonic relationship, while PCC measures the degree of linearity between subjective and expected QoE. For PCC and SROCC, a higher number implies a better result, whereas for RMSE, a lower value indicates a better result. Accurate and effective QoE prediction models are essential for establishing and maintaining streaming video services across many viewing devices. The proposed CNN-QoE model may also improve current QoE prediction accuracy while decreasing computational complexity. As a consequence, CNN-QoE may be a good match for upcoming QoE prediction systems or mobile video streaming applications based on QoE [14].

2.2. QoE metrics and protocol design for video streaming

DASH protocols are used on the client side in work [13] as a way to improve viewer QoE by dynamically adapting to the most suitable video bitrate in response to changing network conditions. Furthermore, with the deployment of MEC, Internet service providers may save money by caching and delivering video at the network edge, reducing backhaul and data traffic. The authors want to discover how much

money they can save in collaborative mobile edge caching settings by integrating client QoE with network data traffic optimization. Next, compare joint edge caching to non-collaborative approaches to determine whether it increases QoE traffic in any manner. Simulation results indicate that network-assisted bitrate adjustment combined with collaborative edge caching enhances client QoE and lowers network traffic burden significantly when compared to prior joint edge caching techniques. The system model of work [13] examines the caches of the local and then the neighbourhood edge servers unless the block or bitrate of the customer's desired video does not exist in the local store. When the edges are unable to find the chunk/bitrate, transferring from the origin server is considered. Finally, the actual bitrate allocation is determined by maximizing the joint QoE-traffic utility objective. Following the assignment of bitrates, the possibility of caching among the different chunks/bitrates given for future client access is investigated.

It's worth noting that the collaborative edge caching isn't required for edge-assisted bitrate adaption. Clients and edge servers work together in the former to ensure optimum and equitable bitrate distribution. The edge servers work together to enhance the overall cache hit rate in the latter. The authors examine the scheduling of *S*, DASH mobile clients across |*T*| time slots, with each space lasting *t* seconds. Initially, the contents of numerous movies are stored on the origin server. Each movie is split into *C* second chunks, with each chunk having a set size of *C* seconds. Set *R* represents a number of distinct bitrates that are available. The base station distributes downlink resource blocks to subscribing clients according to a proportionate fair policy, with *K* edge servers installed in the system and associated with each server. $W_k^{(t)}$ represents the available downlink resource blocks at edge server *k* in time slot *t*. Client *i*'s arrival and departure timings are indicated by $A_i \; and \; D_i$, respectively. In addition, each client's media player has a video buffer with a maximum capacity of B_i^{max}, where $B_i^{(t)}$ represents the amount of video data in the client's buffer during time slot *t*. The received signal to noise ratio of the client *i* from the base station *k* at time slot *t* is represented as $SNR_{ik}^{(t)}$ depending on the client mobility. Furthermore, $r_{ik}^{(t)}$ – denotes the bitrate given to the current chunk of client *i* assigned to server *k* during time slot *t*. The binary variable $\alpha_{ik}^{(t)}$ is defined in such a way that $\alpha_{ik}^{(t)} = 1$ denotes the client *i*'s assignment to edge server *k* at time slot *t*. Video stalling, video quality, bitrate

switching, and initial playback buffering time, also known as initial or startup delay, are the major variables that influence QoE in dynamic adaptive video streaming, according to DASH literature. Video quality, starting buffer, stalling ratio, and fairness are among the measures provided. The video quality that customers experience during a streaming session is directly proportional to the streaming bitrate. As video chunks are broadcast at a high bitrate, the viewing quality improves. The authors of [13] depend on the average bitrate at which client *i* watches the movie throughout the course of its streaming session, which can be calculated using the following equation:

$$AQ_i = \frac{C}{|D_i - A_i|} \sum_{\forall A_i \le t \le D_i} \sum_{1 \le k \le K} \alpha_{ik}^{(t)} * r_{ik}^{(t)}, \tag{17}$$

The initial buffer delay, indicated by L, is the time it takes for data in the playback buffer to reach its full capacity from the moment the client arrives. A larger initial buffer delay, on the other hand, helps to minimize video stopping occurrences during the streaming session. Although a lengthy wait has a little effect on QoE, most customers are willing to put up with lengthier initial delays in exchange for smoother, interruption-free video. For the sake of simplicity, the buffer delay was omitted in our analytical model since the effect on QoE is not as important as the impact of video bitrate and switching. Ratio of Stalling When the client's buffer fills up (playback stops) as a result of streaming video at high bitrates with limited attainable throughput, a video stall event occurs. To represent stalling quantitatively in the optimisation issue, we must first establish the relation for the client's buffer filling level at each time slot [13].

$$B_i^{(t)} = \begin{cases} B_i^{(t-1)} + \widehat{Thr}_{ik}^{(t)} * \Delta t, \ A_i \le t \le A_i + L_i \\ B_i^{(t-1)} + \left(\widehat{Thr}_{ik}^{(t)} - r_{ik}^{(p)}\right) * \Delta t, \ A_i + L_i < t \le D_i \end{cases}, \tag{18}$$

where $\widehat{Thr}_{ik}^{(t)}$ – represents the theoretical throughput by a client *i* allotted to server *k* at time slot *t*. [13] points out that during the startup period, which lasts during the first buffer delay, there is no video streaming on the client's player. As a result, the effective throughput of each client at a particular time slot is calculated using its theoretical throughput, and the number of other concurrent clients at that time slot. Fairness. Client-based systems may not distribute bitrates properly in certain cases, resulting in injustice among competing clients. Strive to distribute the optimum sustainable bitrate for each current client, with the smallest variation from the average of bitrates given to

the other simultaneous active clients on the same server, to prevent unfair issues. The goal of fair bitrate allocation is to reduce the total divergence of the assigned bitrates to each client *i* from the average throughout the course of its whole streaming session, as shown by the following summation: $F_i = \sum_{t=A_i}^{D_i} \sum_{1 \le k \le K} \alpha_{ik}^{(t)} * \left| r_{ik}^{(t)} - \bar{r}^{(t)} \right|$. Where $\bar{r}^{(t)}$ is the average bitrate of other concurrent clients during time slot *t*. The equation's minimization should also fulfill the base station's instantly accessible resource blocks [13].

2.3. Contributions

Convolutional Neural Networks and Long Short-Term Memory (LSTM) are utilized in this studied for a QoE prediction model to deal with this issue (CNN). Because it incorporates temporal relationships in sequential QoE data, the LSTM-based QoE prediction model has the greatest accuracy.

Nonetheless, due of the usage of consistent processing across time, the chained composition in the LSTM architecture is prudently costly to forecast user QoE. It means, for example, that the next processing steps must wait until the preceding one's outcome is known. It also raises the issue of the model's efficiency on machines with low power consumption, such as mobile devices, which may lack the computing capacity to execute such algorithms with QoE.

This thesis aims to develop CNN and LSTM models for QoE prediction using the LIVE Netflix database. The main objectives of work:

- Review and analyse the various ANN designs presented in the current literature, focusing on RNN and CNN architectures and their different types.
- Explore the variety of ANN architectures and applications used for continuous QoE prediction.
- Investigate the connection between the fundamental hyperparameters NN and the performance of various NNs in terms of final prediction accuracy.
- Compare and contrast the performance and complexity of the ANNs described in the literature with CNN and LSTM.

2.4. Take away results from the current state-of-the-art.

Overall, the most valuable works were [11] and [14], mentioned in section 2.1, the output from articles and given models. These works defined the main idea of the thesis and demonstrated that in academia, a video streaming QoE prediction issue existed. Besides, studies gave basis to the solution of the video streaming QoE prediction problem, the importance of the topic in the 5G network. CNN-QoE and LSTM-QoE models offered there are considered state-of-the-art models and showed a direction to develop the main idea of QoE prediction for video streaming services. Articles were utilised as a baseline and demonstrated CNN-QoE, LSTM-QoE models architecture, number of layers, hyperparameters, and input metrics. Input metrics in baseline works were four, but we used only three, reducing the model complexity. Parameters of models architecture:

- batch size,
- epoch number,
- activation function,
- optimizer

and others were analyzed from baseline work and improved in newly proposed models. Furthermore, evaluation metrics were analyzed through these works, which were applied and compared later in section 4.

3. PROPOSED APPROACH

The proposed approach consists of studying, analysing, and comparing to investigate the trade-off between different ML tools' performance and complexity. After analyzing the previous studies, this thesis found the most valuable work as a starting point was [11] and [14]. The proposed models are motivated because of the rapid increase in video streaming in mobile networks. These QoE models can improve the video streaming to end-users by evaluating their feedback from playback. Therefore, there were given initial metrics and QoE model, which was extended in this thesis. In current literature was given four metrics: *continuous subjective score, number of re-buffering events, playback indicator, and STRRED.* Nevertheless, to decline the complexity of the QoE model in this thesis, three out of four metrics except playback indicator was proposed.

Additionally, it should be noted that in work from section 2.1. was presented to utilise LIVE Netflix, LIVE Netflix II, and LFOVIA QoE databases. These databases contain many metrics, but current literature proposed using only the mentioned metrics. In work [11] was offered to LSTM-QoE model for prediction, while work [14] suggested a CNN-QoE and Temporal Convolutional Network (TCN) QoE model.

The structure of the section begins with a description of the database from the current literature and describes the models, pre-processing and normalization with the attached code in 3.1. In section 3.2. presented novel NN models, their hyperparameters with code implementation.

3.1. Dataset and code used from current literature

3.1.1. Datasets

The LIVE-Netflix Database has a comprehensive subjective QoE, which combines with real-time network and buffer data, perceptive video coding and quality ratings are possible, as well as client-based adaptability [25]. Several elements of streaming adaptation are captured in the database, including changes in video quality, over buffering events of variable length and recurrence, changes in spatial resolution, and video content kinds. Subjective data is excellent for training different QoE models since it includes both historical and ongoing assessments. Human opinion scores, on the other hand, are used in streaming applications to analyze the deterioration of streaming video and to evaluate objective video quality models and QoE forecasts.

While the database is broad and realistic, it still has certain limitations based on the most up-to-date video encoding and streaming concepts. Visual quality, novelty, re-buffering or quality switching, as well as other variables such as audio quality or contextual considerations such as user expectations for video streaming service watching and display device environments, all affect QoE. One of the database's primary design goals is to serve as a testbed for developing streaming video quality models and QoE prediction [25].

Figure 14. Some footage from the LIVE-Netflix dataset. Content 5, 6, and 8 from dataset [25].

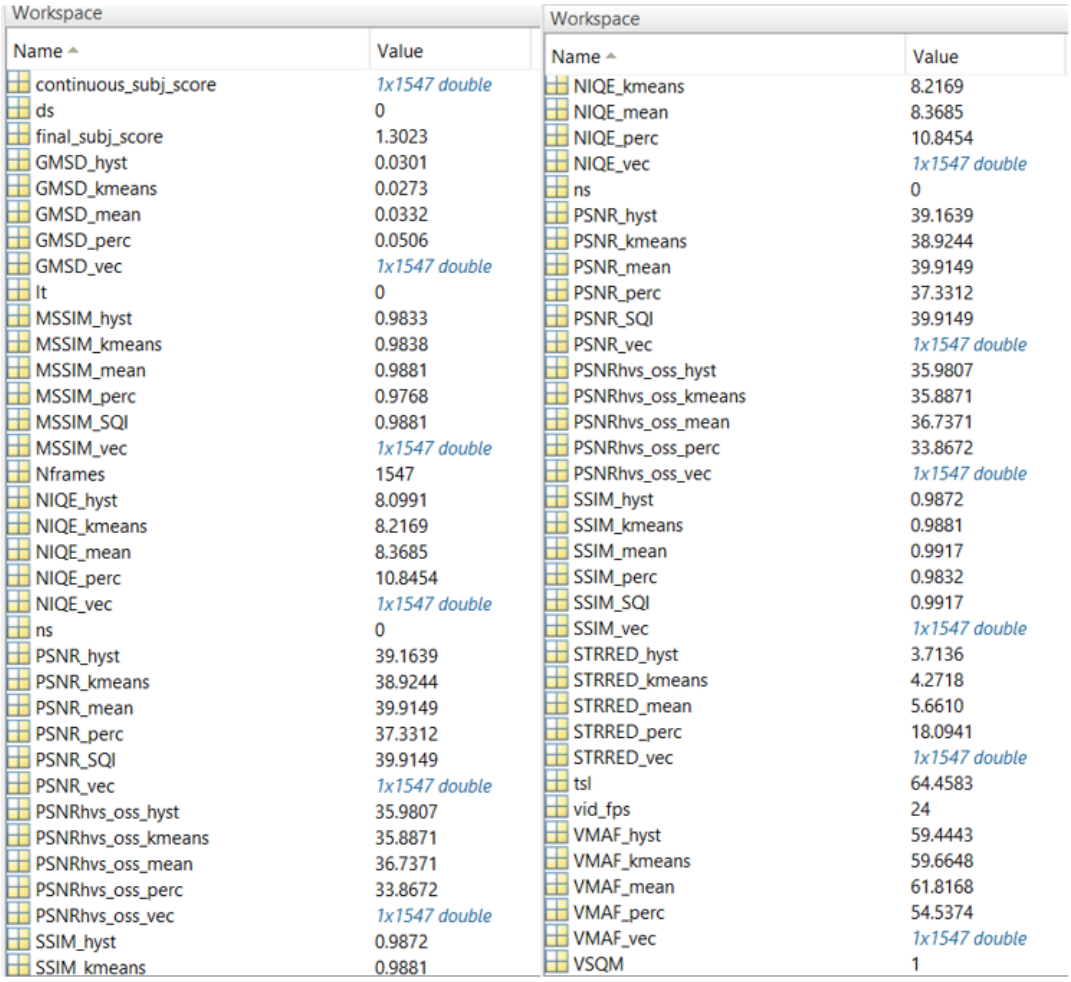
The LIVE-Netflix video QoE database was used to obtain input parameters for neural networks [26]. There are 112 distorted videos in the collection, which were assessed by 55 mobile users. By controlling a collection of 8 distinct playout patterns, the movie was created from 14 video contents with 1080p quality at 24, 25, and 30 fps, respectively. Several frames from video sequences are shown in figure 14. The video in the database includes compression rate H.264, re-buffering events and their combination, despite newer compression standards are being developed like H.265/HEVC and VP9. Furthermore, the database comprises 11 content contributed by Netflix, such as drama, action, comedy. Another three belongs to Consumer Digital Video Library (CDVL), which is public video content [26].

The database format was given as a Matlab file with the extension ".mat" in figure 15, which shown the existing information in the database and values. There is a plant of metrics, but this thesis utilised metrics, such as *continuous subjective score*, *re-buffering events*, *the time elapsed since the last video impairment,* and *Spatio-Temporal Reduced Reference Entropic Differences (STRRED).* All metrics were described in the database as follows:

- The *continuous subjective score* in the database is called "continuous_subj_score". The *continuous subjective score* obtains reliable results for traditional and compressed television systems are presented, as well as an evaluation of scalar MOS in a variety of static and moving video

applications. The broad range of potential techniques and tests that must be examined, the high number of needed witnesses, and the time required are all drawbacks. Objective testing may be done in a variety of ways. Algorithms for analyzing video picture features are more closely linked to subjective findings than pixel techniques, it may be said. The greatest findings and correlations between subjective and objective evaluations come from different measures and functions, although they are unlikely to be technology-dependent [25].

- The *number of re-buffering events* is defined by the variable "ns". When a re-buffering event happens between the start of the session and the present time, the user becomes more irritated.
- The *time elapsed* is named the variable "tsl" — for example, bitrate switch or re-buffering occurrence.
- The metric *STRRED* has several forms such as "STRRED_mean", "STRRED_kmeans" and others in the database, but in this thesis was used "STRRED_vec" which contains a vector of *STRRED* values. When compared across databases, the *STRRED* score correlates extremely well with subjective ratings. *Spatio Reduced Reference Entropic Differences (SRRED)* and *Temporal Reduced Reference Entropic Differences (TRRED)* indices are combined in the *STRRED* index. It's worth noting that the *SRRED* and *TRRED* indices each work with data derived from certain spatial and temporal frequency components. Furthermore, whereas *SRRED* pointes are computed utilising just spatial frequency data, *TRRED* indices are computed using both spatial and temporal data, with spatial data being used to weight the time data. As a consequence, only the TRRED index is affected by temporal distortion, while the *SRRED* and *TRRED* indexes are affected by spatial distortion. Compounding a quality score based on spatial and temporal data is concerned with processing that takes place in the latter stages of human visual processing when there is records of interaction between them [27].



Workspace

Name	Value
continuous_subj_score	1x1547 double
ds	0
final_subj_score	1.3023
GMSD_hyst	0.0301
GMSD_kmeans	0.0273
GMSD_mean	0.0332
GMSD_perc	0.0506
GMSD_vec	1x1547 double
It	0
MSSIM_hyst	0.9833
MSSIM_kmeans	0.9838
MSSIM_mean	0.9881
MSSIM_perc	0.9768
MSSIM_SQI	0.9881
MSSIM_vec	1x1547 double
Nframes	1547
NIQE_hyst	8.0991
NIQE_kmeans	8.2169
NIQE_mean	8.3685
NIQE_perc	10.8454
NIQE_vec	1x1547 double
ns	0
PSNR_hyst	39.1639
PSNR_kmeans	38.9244
PSNR_mean	39.9149
PSNR_perc	37.3312
PSNR_SQI	39.9149
PSNR_vec	1x1547 double
PSNRhvs_oss_hyst	35.9807
PSNRhvs_oss_kmeans	35.8871
PSNRhvs_oss_mean	36.7371
PSNRhvs_oss_perc	33.8672
PSNRhvs_oss_vec	1x1547 double
SSIM_hyst	0.9872
SSIM_kmeans	0.9881

Workspace

Name	Value
NIQE_kmeans	8.2169
NIQE_mean	8.3685
NIQE_perc	10.8454
NIQE_vec	1x1547 double
ns	0
PSNR_hyst	39.1639
PSNR_kmeans	38.9244
PSNR_mean	39.9149
PSNR_perc	37.3312
PSNR_SQI	39.9149
PSNR_vec	1x1547 double
PSNRhvs_oss_hyst	35.9807
PSNRhvs_oss_kmeans	35.8871
PSNRhvs_oss_mean	36.7371
PSNRhvs_oss_perc	33.8672
PSNRhvs_oss_vec	1x1547 double
SSIM_hyst	0.9872
SSIM_kmeans	0.9881
SSIM_mean	0.9917
SSIM_perc	0.9832
SSIM_SQI	0.9917
SSIM_vec	1x1547 double
STRRED_hyst	3.7136
STRRED_kmeans	4.2718
STRRED_mean	5.6610
STRRED_perc	18.0941
STRRED_vec	1x1547 double
tsl	64.4583
vid_fps	24
VMAF_hyst	59.4443
VMAF_kmeans	59.6648
VMAF_mean	61.8168
VMAF_perc	54.5374
VMAF_vec	1x1547 double
VSQM	1

Figure 15. Information in the Database.

Additionally, adding more metrics to thesis work from the database is possible, but it increases the complexity of the proposed QoE model. It is worth mentioning that efficient and successful model design requires a wise and scrupulous decision on choosing the set of indicative features. Therefore, the thesis work used three metrics because after investigating which metrics are more suitable for the evaluation of QoE was decided to utilise the metrics as mentioned earlier and fundaments of thesis work [14].

3.1.2. Code.

The suggested approaches of work [11] and [14] are given in the GitHub platform [28]. First of all, before pre-processing part, it was to import necessary libraries such as NumPy, Keras, Tensorflow, Sci-kit, Pandas, SciPy, and Matplotllib. Then, the authors of [11] and [14] in the pre-processing part extracted the four metrics: continuous subjective score, re-buffering events, playback indicator, and STRRED from the LIVE-Netflix QoE database. In figure 16 is demonstrated a pre-processing part which

indicates that only three metrics were retrieved for this thesis from the MATLAB ".mat" file using the "hdf5storage" python package.

```
cont = 15
seq = 8
dic_continuous_subj_score = {}

score_continuous = []
score_continuous = np.array(score_continuous)

Nrebuffers = []
Nrebuffers = np.array(Nrebuffers)

TSL = []
TSL = np.array(TSL)

STRRED = []
STRRED = np.array(STRRED)


for i in range(1, cont):
  if i == 10 or i == 11:
    np.array(hdf5storage.loadmat('content_' + str(i) + '_seq_' + str(j) + '.mat'))
  else:
    for j in range(seq):
      continuous_subj_score = np.array(hdf5storage.loadmat('content_' + str(i) + '_seq_' + str(j) + '.mat')['continuous_subj_score_avg1sec'])
      num_rebuf = np.array(hdf5storage.loadmat('content_' + str(i) + '_seq_' + str(j) + '.mat')['Nrebuffers_avg1sec'])
      time_elapsed = np.array(hdf5storage.loadmat('content_' + str(i) + '_seq_' + str(j) + '.mat')['TSL_avg1sec'])
      strred = np.array(hdf5storage.loadmat('content_' + str(i) + '_seq_' + str(j) + '.mat')['STRRED_vec_avg1sec_with_rebuf'])

      if i == 1 and j==0:
        score_continuous = continuous_subj_score
        Nrebuffers = num_rebuf
        TSL = time_elapsed
        STRRED = strred

      else:
        score_continuous = np.concatenate((score_continuous, continuous_subj_score), axis=None)
        Nrebuffers = np.concatenate((Nrebuffers, num_rebuf), axis=None)
        TSL = np.concatenate((TSL, time_elapsed), axis=None)
        STRRED = np.concatenate((STRRED, strred), axis=None)
```

Figure 16. The Pre-processing part of the proposed approach.

Feature extraction was made using a loop and collected into arrays. Subsequently, normalisation was applied to the extracted objects. Normalisation consists of rescaling numeric attributes with absolute values ranging from 0 to 1. Machine learning algorithms tend to execute superior or congregate faster when different variables are on a smaller scale. Hence, it is general practice to normalise the data before training machine learning models on it. Normalisation also makes the learning process less sensitive to the function scale, leading to better odds after training. This process of increasing the suitability of features for learning by scaling is called feature scaling, and the normalisation formula is

$$\mathrm{x}_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{19}$$

Subtract the minimum value from each input and then divide the result by the span, where the span is the distinction between the maximum and minimum values. Figure 17 shows the normalisation of features for this thesis. The normalisation range for the extracted metrics was 2 to -2 for the continuous subjective score, and STRRED was

between 3150 and 0 for maximum and minimum, respectively. Next, extracted features were concatenated to one variable to feed as one input vector to the neural network. One of the final steps is to apply neural networks.

LSTM-QoE and CNN-QoE models were provided in works [11] and [14], and these models were evaluated using Pearson Correlation Coefficient (PCC), Spearman Rank Order Correlation Coefficient (SROCC), and Root Mean Squared Error (RMSE). Pearson's correlation coefficient is a statistic that measures how strong a linear relationship between two variables is. In essence, Pearson's correlation coefficient attempts to build a line of best fit among the two variables and displays how near all of the data points are to that line. Consider how well the data points correspond to this new best-fit model. Pearson's correlation coefficient may vary from +1 to -1. A value of 0 implies that there is no connection between the two variables. A number greater than 0 indicates a good relationship. When one variable's value increases, the value of the other increases as well. A negative link is shown by a number smaller than zero, indicating that the value of one variable increases while the value of the other decreases. The following is the PCC calculation equation:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \quad (20)$$

Pearson's correlation coefficient will be closer to +1 or -1, depending on whether the connection is positive or negative, the stronger the link between the two variables. When you receive a +1 or -1 as a result, it means that all of your data indicates are on the line of best fit and that none of them depart from it.

```
#NORMALIZATTION
# Normalizing factor for STRRED
STSQ_max = 3150
STSQ_min = 0

STSQ_norm = (STRRED - STSQ_min)/(STSQ_max - STSQ_min)
for kk in range(1,len(STSQ_norm)):
  if STSQ_norm[kk] > 1.0:
    STSQ_norm[kk] = 1

#NORMILIZE SCORE CONTINUOUS
score_continuous_max = 1.9
score_continuous_min = -2.6

SC_norm = (score_continuous - score_continuous_min)/(score_continuous_max - score_continuous_min)
```

Figure 17. The normalisation of extracted features.

Values between +1 and -1, such as 0.8 or -0.4, indicate variation around the line of best fit. The greater the deviation from the best-fit line, the closer the material is to 0. Spearman Correlation of Rank Order A nonparametric variant of the PCC is the coefficient. The strength and direction of the monotonic link between two ranking variables is determined by SROCC, not the magnitude and direction of a linear connection among two variables, which is determined by PCC. The SROCC may accept values ranging from +1 to -1. A perfect connection of rankings is worth +1 points, no association between levels is worth 0 points, and an ideal negative correlation of grades is worth -1 points. The weaker the link between the rankings, the closer it gets to zero. SROCC may be computed in two ways, depending on whether the data has no related levels or whether the information has associated rankings. When there are no equivalent rankings, use the following formula:

$$p = 1 - \frac{\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{21}$$

where d_i denotes the difference between paired rankings and *n* is the number of instances. The formula for determining equal grades is the same as for determining equal grades (17).

The Root Mean Square Error is another common method for calculating the error in a collection of regression predictions. When forecasting quantitative data, the RMSE metric is a common method to evaluate model error. The square root of the mean square of the discrepancies between actual outcomes and predictions is used to calculate the RMSE. Positive values are obtained by squaring each error, and the square root of the root mean square error restores the error metric to its original units for comparison.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\widehat{y_i}-y_i)^2}{n}}, \tag{22}$$

where, $\widehat{y_1}, \widehat{y_2}, \dots, \widehat{y_n}$ are predicted values, $y_1, y_2, \dots, y_n$ are observed values, and *n* is the number of observations.

Convolution layers have been proven to be effective in univariate time sequence analysis in a variety of applications, including time sequence sensors, audio signals, and natural language processing. The CNN-QoE model is shown in Figure 18, and it

should be noted that the Keras built-in "Conv1D" function was utilized in conjunction with the "RelU" activation function. The Rectified Linear Activation Function (ReLU) is a linear function that, if positive, will output the input directly. Otherwise, it will return a value of zero. Because the model in which it is employed is simpler to train and frequently offers higher productivity, it has become the default activation function for many kinds of neural networks. One input, three hidden convolutional layers, and a dense output layer make up the CNN-QoE model architecture. The "Conv1D" layer produces an output tensor by convolutioning the input layer in one spatial or temporal dimension. The numbers 32 and 2 in the "Conv1D" parameter denote the number of filters and kernel size, respectively. The kernel size for "Conv1D" is a vector of original data is multiplied by the filter, and the filter is a vector multiplied by the original data.

```
inputs = tf.keras.Input(shape=(steps, features))

x = tf.keras.layers.Conv1D(32, 2, activation='relu')(inputs)
x = tf.keras.layers.Conv1D(32, 4, activation='relu')(x)
x = tf.keras.layers.Conv1D(32, 5, activation='relu')(x)

x = tf.keras.layers.Flatten()(x)
outputs = tf.keras.layers.Dense(1, activation = 'relu', use_bias=False)(x)
```

Figure 18. The CNN-QoE model approach.

An LSTM-QoE is the suggested model from studies [11] and [14]. LSTM has established itself as a reliable technique for tackling complicated sequence prediction issues. Rather of sending a single value, the LSTM layer above transmits a sequence of values to the LSTM layer below. Instead of having one output time step for all input time steps, each input time step should have its own output time step. One input, two hidden bidirectional LSTM layers, and a dense output layer make up the LSTM-QoE model architecture, as illustrated in figure 19. The number of neurons in the LSTM was selected to be 22 and the activation function to be "tanh." The activation function of the "sigmoid" activation function is extremely similar to "tanh," and even has the same S-shape. The procedure accepts any real number as input and returns values between -1 and 1. The more positive the information, the closer the output value is to 1, and the smaller the entry, the more negative the information, the closer the output value is to -1. The activation function "tanh" is calculated as follows: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.

```
inputs = tf.keras.Input(shape=(steps, features))

x = tf.keras.layers.LSTM(22, return_sequences=True, activation='tanh')(inputs)
x = tf.keras.layers.LSTM(22, return_sequences=True, activation='tanh')(x)

x = tf.keras.layers.Flatten()(x)
outputs = tf.keras.layers.Dense(1, activation = 'relu', use_bias=False)(x)
```

Figure 19. The LSTM-QoE model approach.

To sum up, work [11] and [14] studied which metrics are better to select, how CNN-QoE and LSTM-QoE models operate, what kind of activation functions to apply, and the difference between activation functions and other models parameters. Additionally, the evaluation and comparison part is provided in next section 3.2.

3.2. Proposed solution

In this thesis, QoE prediction performance of different types of NNs based on work [11] and [14] have been analyzed and studied. Furthermore, this thesis offered other types of neural networks based on work reviewed in current literature. More precisely, combined bidirectional LSTM and convolution neural network, and regular densely connected neural network. Compared to the previous study, the difference starts with three metrics: continuous subjective score, re-buffering events, and STRRED. The proposed models are done using the Google Colab tool for machine learning, including the Jupiter notebook platform.

Additionally, in preparing the training set for the model wasn't used a shuffle method for data. Also, it is worth mentioning that the “Flatten” layer is used in this work as a reshaping layer that performs the transformation procedure of the input 3D tensor into a 2D shape. The primary machine learning parameters used in this work will be discussed in the following paragraphs.

3.2.1. Batch size

The batch size hyperparameter specifies how many samples will be delivered simultaneously across the network. If you have 1050 training samples and the batch size is set to 100, for example. The method trains the network using the first 100 samples (1–100) from the training dataset. The network is then taught again using the second 100 samples, 101 through 200. The process may be repeated until the whole system has been distributed with all of the choices. The problem may occur with the final batch of samples. The number 1050 was used in the example above, which is not

evenly divisible by 100. Obtaining the past 50 patterns and training the network is the easiest approach. Batch size has the benefit of using less memory. The entire training process uses less memory since the network is trained with fewer samples. If you can't fit the whole dataset into your computer's memory, less RAM is important. The disadvantages of utilizing batch size are as follows: The gradient estimate becomes less precise as the batch size decreases. All models in this study were given a batch size of 64, which was determined after many optimization assessments to match our sample size.

The epoch hyperparameter was utilized in all of these models to assess the QoE prediction and improve machine learning accuracy. The number of times the training algorithm processes the whole training dataset is determined by the epoch. Each sample in the training dataset has the chance to change the model's internal parameters once every epoch. One or more parties make up an era. A single burst epoch, for example, is referred to as a batch gradient descent learning method. Traditionally, the number of epochs is high, typically hundreds or thousands, to enable the learning process to run until the model error is suitably reduced. The number of epochs used in this study ranged from 150 to 200, which is sufficient for a suggested neural network to converge to the best outcome. The distinction between batch size and epoch is as follows:

- Each epoch represents one forward and backward trip through all of the training instances.
- The number of training instances in one forward or backward pass determines the batch size. More memory space is required as the batch size grows.

Several passes are equal to the number of iterations, with each pass utilizing a batch size number of samples. One pass equivalent to one forward pass plus one backward pass should not be recorded as two passes since the forward and backward passes are two distinct passes.

3.2.2. Learning rate

Each time the model weights are changed, the learning rate is a parameter that regulates how much the model changes in reaction to the predicted mistake. Picking a learning rate is challenging since there are so many variables to consider low number may result in a long learning process that becomes stuck, while a high value can result

in learning too fast, a poor collection of weights, or an unstable learning process. The learning rate, in particular, is an adjustable hyperparameter with a modest positive value, often in the range of 0.0 to 1.0, that is utilized in training neural networks. Error back-propagation measures the amount of error caused by the node weights in the network during training. It scales with the learning rate rather than updating the significance with the entire amount. It indicates that the learning rate is 0.1. Traditionally, the weights on the network are updated by 0.1 multiplied by each time the values are changed, the calculated weight loss or 10% of the anticipated weight error is calculated. The learning rate was found to be 0.001 in this thesis. In this instance, it is an optimum value since lowering or raising the rate has resulted in substantial deterioration of the neural network gradient descent algorithm's convergence.

3.2.3. Adam optimizer

Adam was selected as an optimization method since it is one of the most extensively utilized and widely used optimization approaches in recent literature. Adam's optimization method is an extension of stochastic gradient descent, which has lately gained traction in computer vision and natural language processing for deep learning applications. Adam isn't your typical stochastic gradient descent algorithm. For all weight updates, stochastic gradient descent maintains a constant learning rate, this remains constant throughout the workout. As the learning progresses, the learning rate of each network weight is recorded and adjusted individually. Adam uses the mean of the second gradient moments and off-centre variance in addition to the average of the first moment to modify the parameter learning rate. The method computes the gradient's exponential moving average, and the angle square determines the pace at which these moving averages decay. The starting value of the moving averages, which should be close to 1.0, causes the torque estimations to be biased towards zero.

3.2.4. Feed-Forward Neural Network QoE model.

As with earlier models that used the built-in dense layer function, the suggested model is a Feed-Forward Neural Network. The dense layer is the first layer of a neural network, and it contains parameters like the number of units, activation function, and bias boolean parameter. The number of units in a layer corresponds to the size of the

neurons. This job is worth 100 neurons, which was determined after considering the input characteristics and output correctness. Similarly, the activation function "RelU" was found experimentally.

```
inputs = tf.keras.Input(shape=(steps, features))

x = tf.keras.layers.Flatten()(inputs)
x = tf.keras.layers.Dense(units=100, activation='relu')(x)
x = tf.keras.layers.Dense(units=100, activation='relu')(x)
x = tf.keras.layers.Dense(units=100, activation='relu')(x)

outputs = tf.keras.layers.Dense(1, activation = 'relu', use_bias=False)(x)
```

Figure 20. Proposed FNN-QoE approach

In a neural network, a dense layer is a densely linked layer in which each neuron gets input from all neurons in the preceding layer. The thick layer was the most often utilized in the models. A thick layer in the background conducts matrix-vector multiplication. The matrix's values reflect practical arguments that may be taught and modified via mistake back-propagation. As a result of the dense layer, an m-dimensional vector is produced. As a result, the dense layer is mostly utilized to reshape the vector. Dense layers may also execute operations like rotation, scaling, and translation on a vector. It's also worth noting that the FNN-QoE model's architecture is made up of four dense layers: one input, three hidden, and one output layer, as shown in Figure 20.

3.2.5. Bidirectional LSTM QoE model

The basic concept behind bidirectional recurrent neural networks (RNNs) is straightforward. It entails duplicating the network's initial repeating layer. There are now two levels side by side, with the first level receiving the input sequence and the second receiving a reverse duplicate of the input sequence. The usage of bidirectional sequence representation was first approved in the area of voice recognition because the context of the whole statement is used to understand what is said rather than linear interpretation. Keras supports bidirectional LSTMs through the bidirectional layer wrapper. A recurrent layer, such as a Gated Recurrent Unit (GRU) or Long Short-Term Memory, is sent as an input to this wrapper (LSTM). It's also expected to define the merging mode, which controls how the forward and backward outputs are combined before being sent on to the next layer. The following are some merging mode options:

- "Sum": the results are added together.

- "Mul" means that the outputs are multiplied.
- "Concatenate": The number of outputs for the following level is doubled by concatenating the outputs together.
- "Ave": Calculates the output's average.

The default option is " Concatenate," and bidirectional LSTM research often utilize this technique.

```
inputs = tf.keras.Input(shape=(steps, features))

x = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(22, return_sequences=True, activation='tanh'))(inputs)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(22, return_sequences=True, activation='tanh'))(x)

x = tf.keras.layers.Flatten()(x)
outputs = tf.keras.layers.Dense(1, activation = 'relu', use_bias=False)(x)
```

Figure 21. Proposed bidirectional LSTM approach.

The dropout function is needed to remove some random neurons from the model during the training phase, which helps prevent overfitting.

3.2.6. Combined CNN and LSTM QoE model

A hybrid CNN and biLSTM QoE model is also presented in this thesis. Figure 22 shows the combined model architecture, which consists of one input layer, one hidden convolutional layer, one bidirectional LSTM layer, and a dense output layer. It's also worth mentioning that the combined QoE model is more difficult to understand than the other alternatives. Therefore, compiling the model takes much more time and computation resources. However, the result is much better than suggested in work [11] and [14]. The results are discussed more precisely in section 4.2. First, the input convolutional layer parameters are set to the same as before mentioned: filter size was 32 and kernel size was 2 with activation function – “RelU”. Then, the next bidirectional layer was utilised LSTM with parameters such as the number of neurons was set to 22 and activation function was “tanh”.

```
inputs = tf.keras.Input(shape=(steps, features))

x = tf.keras.layers.Conv1D(32, 2, activation='relu')(inputs)
x = tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(22, return_sequences=True, activation='tanh'))(x)

x = tf.keras.layers.Flatten()(x)
outputs = tf.keras.layers.Dense(1, activation = 'relu', use_bias=False)(x)
```

Figure 22. Proposed combined CNN and biLSTM, QoE model approach.

One of the leading modern approaches to finding an appropriate and highly performative neural network is to combine existing pieces of different NNs into one big

and efficient model. In this work, we connect one single convolution layer with one bidirectional LSTM layer to create a semi-RNN model. In the first place, the convolution layer can be interpreted as a feature extracting technique – that is, by sliding the kernel window through the input data and multiplying it, the layer can pull essential elements out of the sequence. This procedure can be compared with a typical 2D image extraction. At the same time, commonly used for time sequences recurrent neural network in the shape of bidirectional LSTM – can utilize these previously extracted features and process them accordingly with the recurrent approach. Therefore, this technique helps to obtain an actual high performance while being one of the competitive models in a complex way.

4. PERFORMANCE EVALUATION

4.1. Simulation model and parameters

The main goal of this work is to study and compare new and effective machine learning methods applied to the problem of continuous QoE prediction for 5G networks. Here, the expected result is an experimental evaluation and an estimation of the practical complexity of the analyzed models, which clarifies the possibilities of using tools based on ML. The following paragraphs summarise the conclusions by analysing the results obtained from an exhaustive and repetitive training procedure for the previously described neural networks. The work methodology consists of extracting features from the database, creating sequences, pre-processing, normalization, and training a model. To properly compare the complexity of the proposed models, we first need to get a clear idea of how to calculate the multiplicative complexity index. All reviewed NN structures calculate the computational complexity in actual accumulations on the restored output QoE symbol. Traning complexity was not considered as we evaluated computational complexity in real-time - the evaluation stage, which is the essential part, while NN training was done offline - the calibration stage. However, to better understand the complexity, in our work, we directly associate these complexities with the number of multiplications of the Keras and TensorFlow machine learning frameworks utilised without losing abstraction [29].

Let *B* be the batch size, t_s be the input time series size, $t_s = M$, where *M* denotes memory steps, and f_n is the number of features, which in our work was 3. The real and imaginary portions of each symbol were then restored, and the number of results on the symbol, s_0, was set to 1. Because biLSTM and CNN layers need inputs in the form of level 3 tensors, the NN's input may be parametrized as $[B,\ t_s,\ f_n]$. The FNN has a standard parametrization, with $[B,\ t_s * f_n]$ defining the dimensions of the 2D tensor input. The Flatten layer is usually used to reshape the input tensor. The following formula was used to determine the complexity of FNN:

$$C_{FNN} = t_s f_n n_1 + n_1 n_2 + n_2 n_3 + n_3 s_0 \tag{23}$$

n_1 refers to the number of neurons in the input layer, n_3 to the number of neurons in the hidden layer, and n 3 to the number of neurons in the output layer. The output QoE representation, which in our instance is equal to 1, is represented by s_0. The

computational complexity of a NN based on the biLSTM layer is discussed in the next section. Given the n_h hidden neurons in the biLSTM layer, the complexity of a single biLSTM NN is given by:

$$C_{biLSTM} = 2t_s n_h (4f_n + 4n_h + 3 + s_0) \quad (24)$$

Finally, supposed that the number of filters n_f defines the convolutional layer, and the kernel size k_s determines the complication of the combined model CNN and biLSTM. The output size for each CNN percolator is (t s-k s+1) [29], and the number of time steps is$(t_s - k_s + 1)$ [29].

$$C_{CNN+biLSTM} = f_n n_f k_s (t_s - k_s + 1) + (t_s - k_s + 1) 2n_h (4f_n + 4n_h + 3 + s_0) \quad (25)$$

Additionally, having described the CNN and biLSTM layers complexity - the CNN network from work [11] can be calculated straightforwardly by using a similar approach to the combined NN, but just with three additional convolution layers.

Table 1. Calculated NNs complexity.

Layers	Complexity
Two LSTM layers [14]	2.49E+04
Two biLSTM layers	1.87E+05
Three FNN layers	2.46E+04
Combined Conv1D and biLSTM layers	6.38E+04
Three CNN layers [11]	5.27E+03

As for the biLSTM - downscaling to just LSTM network without a bidirectional feature - can be done by removing coefficient two from the central equation. On the other hand, adding a new layer will give an almost a doubled number of multiplications - since the layer is composed of the same number of neurons with minor differences in output and input multiplications.

4.2. Comparative results

The correctness of the suggested model is evaluated in this thesis using a set of hyperparameters and input characteristics. Furthermore, biLSTM-QoE, combined

CNN + LSTM QoE, and FNN-QoE, from [11] and [14], CNN-QoE, and LSTM-QoE are utilized for scoring since these models are fair in their study and their score is based on the same QoE database. Three evaluation metrics are used to evaluate the prediction accuracy of QoE: Pearson's Correlation Coefficient (PCC), Spearman's Rank Order Correlation Coefficient (SROCC), and Root Mean Square Error (RMSE). The monotonic connection is measured by SROCC, whereas the degree of linearity between subjective and anticipated QoE is measured by PCC. A greater number indicates a better outcome for PCC and SROCC, whereas a lower value indicates a better result for RMSE. These metrics were previously discussed in section 2.1. Table 2 summarizes the findings and assessment of the suggested NN-QoE models using the RMSE, PCC, and SROCC metrics. It's worth noting that the RMSE assessment measure is inversely related to the PCC and SROCC metrics.

The multiplicative complexity of the analyzed NNs is shown in figure 23. As can be seen from the table, the 2-layered biLSTM network has the highest level of complexity - since this type of recurrent neural network has the most significant number of multiplications inside the recurrent and gated structure of LSTM cells by the obtained QoE value. In addition, because this network is bi-directional, the complexity is further increased. The combined Conv1D with the biLSTM network has the second-highest level of complexity since the memory steps used for training are equal to 15, and the convolutional and recurrent structure of the network brings more multiplications per single obtained QoE output.

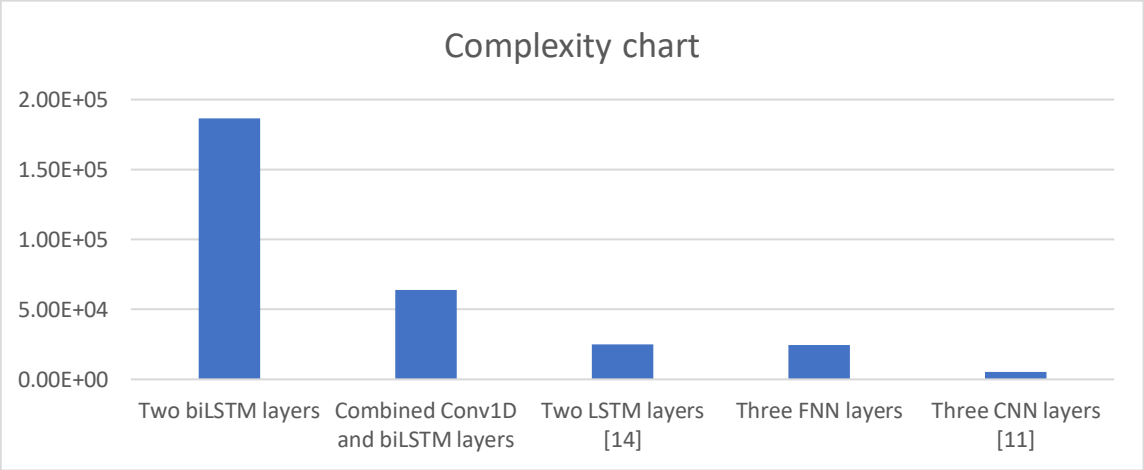


Figure 23. The multiplicative complexity of the analyzed NNs

At the same time, the proposed 2-layered LSTM neural network proposed by [14] has a 2.49E+04 number of multiplications – which takes third place out of the analysed

networks. It is essential to mention that the results obtained by the authors are those for the time steps of number 4, while in our work, we utilized the number of steps equal to 15. This introduces much more memory, and therefore the 2-layered biLSTM has more than twice a higher complexity than the 2-layered LSTM. On the other hand, the FNN neural network is an exciting solution to the QoE prediction has an almost identical complexity level as the previously mentioned 2-layered LSTM network, even with a much higher number of neurons, 100 in FNN to 22 in LSTM, and layers, 3 in FNN to 2 in LSTM. However, due to the specific propagation network type – this model requires much fewer multiplications between the layers of neurons. Finally, the proposed by [11] CNN neural network is the leader in complexity factor – as the convolution layer with the dilation factor requires much fewer multiplications in each convolving layer. Additionally, the number of steps used in work is equal to 8, while mentioning again, in our case, was 15 – which is almost twice memory occupation. Therefore CNN becomes the minor complex network in our analysis.

Considering the complexity information from table 1 and figure 23, the FNN model provides the best level of predictive accuracy by RSME value and occupies one of the lowest levels of the computational complexity. This indicates that the application of FNN could potentially be an optimal technique for QoE prediction, again due to the relatively low complexity level and competitively high prediction accuracy. Additionally, the LSTM [14] NN depicting the most insufficient accuracy with the memory steps of 4 has also shown a similar complexity to the FNN. At this point, it is worth discussing that FNN and 2-layered LSTM occupy the same sophistication level. However, FNN gives higher accuracy than LSTM, which is much better for the prediction application.

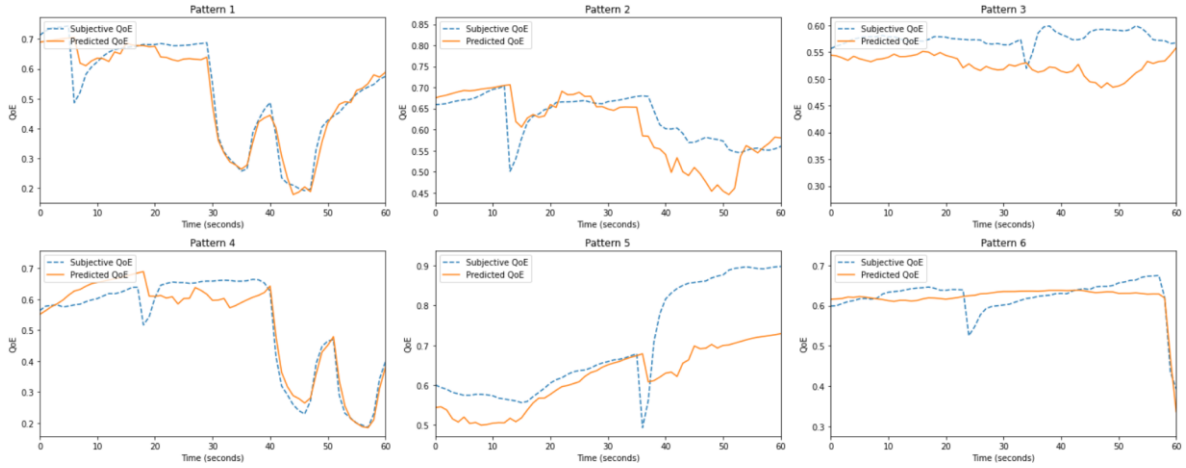


Figure 24. The FNN-QoE model's QoE prediction productivity on the LIVE-Netflix video QoE Database.

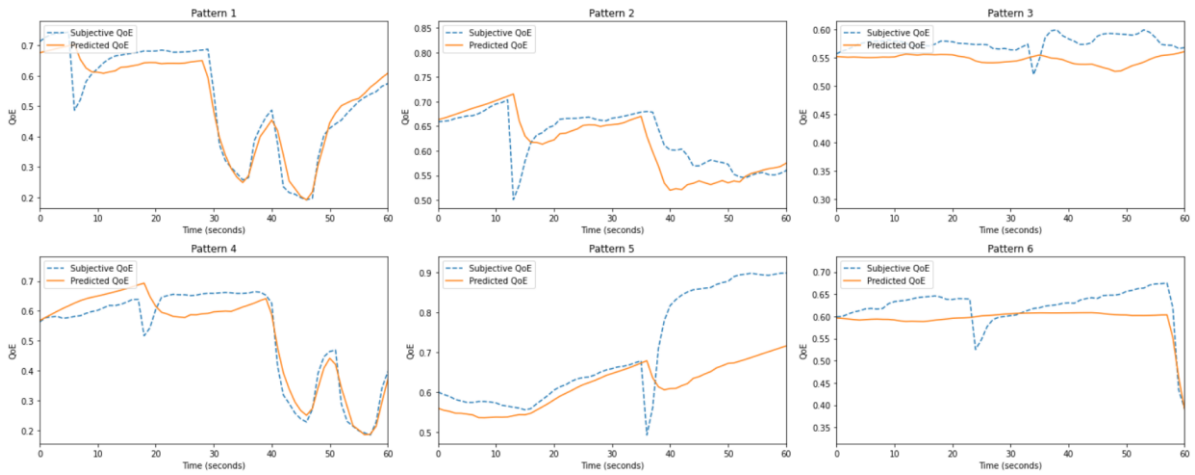


Figure 25. Over the LIVE-Netflix video QoE Database, the combined CNN and LSTM QoE-model performed well in terms of QoE prediction.

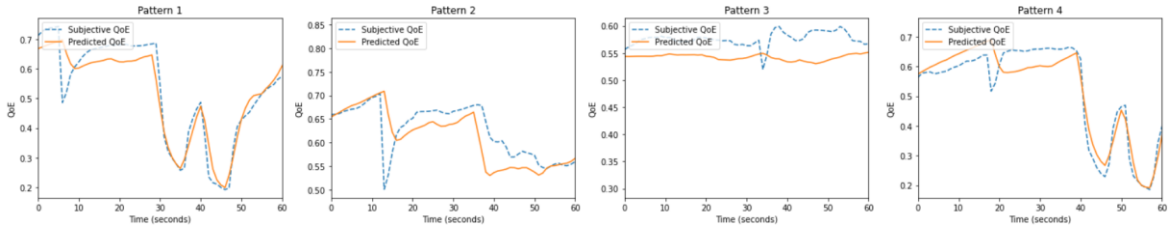


Figure 26. Performance of the biLSTM layers QoE models in predicting QoE via the LIVE-Netflix video QoE Database.

Figures 23-25 illustrates plots to represent the accuracy of prediction for each model. Graphs showing the similarity can be explained as a slight difference in evaluation metrics. In some parts of charts, prediction accuracy is very high, but other factors are very different. Moreover, in plots, the QoE prediction is shown in the 60-second range and when in graphs, the QoE score went rapidly down, indicating a rebuffering event during video streaming. Also, the Pattern defines the video sequence. For example, in figure 25, Patterns 2 and 5 depict rebuffering events in 13 and 36 seconds, respectively.

Table 2. The suggested NN-QoE models' QoE prediction efficiency over the LIVE-Netflix video QoE database. The highest result is shown with a bold typeface.

NN-QoE models	*PCC*	*SROCC*	*RMSE*
CNN-QoE [11]	0.848	0.733	6.97
LSTM-QoE [14]	0.802	0.714	7.78

FNN-QoE	**0.881**	**0.819**	**6.69**
BiLSTM-QoE	0.876	0.806	7.07
Combined CNN+LSTM QoE	0.882	0.809	6.80

Table 2 demonstrates the outcome of this dissertation work. As can be seen from the comparison between all models, the best QoE prediction result was the FNN-QoE model. FNN-QoE model showed the lowest RMSE and highest PCC and SROCC metrics. As mentioned before, the small RMSE means better prediction accuracy. However, the combined QoE model also demonstrated fair results, slightly 0.11 more than the FNN-QoE model. In addition, the thesis results are better than base papers [11] and [14], but concerning the fact that other models were used. However, it would be troublesome to perform and evaluate the offered models without more research work.

Moreover, table 1 and figure 23 shows that the most increased complexity has the biLSTM model comparing to others. Also, it should be recorded that during the simulation, the biLSTM network took much more time to compile than other networks, which denotes the high complexity. Another QoE model, which demonstrated high accuracy in table 2 combined QoE-model, in figure 23 combined CNN and biLSTM model located in second place by complexity, but it has fair prediction accuracy.

4.3. Summary of results

Modern 5G networks are widely recognized for requiring ultra-low latency, high speed, and ultra-wide bandwidth. Even if these high standards are pursued, frequent disruptions or network changes may cause end-user activities to be disrupted. All of these variables, without a doubt, have a major impact on consumers' QoE. As a result, maximizing user experience knowledge requires quick and adaptable video streaming, and consistent QoE prediction may be helpful and illuminating in the case of DASH video streaming.

Comparing the performance of the proposed NN with the models from the current literature in Table 2, we can see that the efficiency of the new FNN-QoE model is more accurate and differs from the LSTM-QoE model by 14% on the RMSE metric.

Additionally, three-layered CNN has only 5% lower performance on the RMSE metric compare to three-layered FNN.

According to the above analysis, we can argue that using modern machine learning tools to apply QoE prediction can be very beneficial due to the high performance and accuracy of modern supervised methods, especially when valuable data is available. At the same time, high-precision networks usually use a lot of application memory and become very difficult to implement in real-time applications on devices in 5G networks. Thus, there is a trade-off between a well-performing NN-based tool that can predict QoE efficiently and continuously and the level of complexity these NN-based tools occupy. An example is that for the FNN neural network, which is undoubtedly one of the most straightforward and most understandable tools in terms of complexity - it gives relatively competitive accuracy results - and at the same time does not require a large amount of memory, like very complex recurrent neural networks. Therefore, this type of NN may become the number one method for QoE prediction. At the same time, using more memory steps, more neurons, together with enhanced activation functions, can lead to better results for repetitive NN, as does biLSTM. On the other hand, more memory means more complexity, leading to a longer learning cycle and higher resource requirements for this network.

5. CONCLUSION

Finally, for video streaming in 5G networks, this thesis investigated, analyzed, developed, and assessed QoE prediction models such as FNN, biLSTM, and a combination of CNN and biLSTM. In suitable network congestion conditions, systems based on NN-QoE prediction models will serve as the first line of defense and inform intelligent traffic management choices, such as when and what levels of scalable video streaming should be discarded to optimize network advantage. In a 5G network, activities with little effect on perceived QoE are possible. Three metrics were used to evaluate the results of all models: PCC, SROCC, and RMSE. Furthermore, two additional recent studies [11] and [14] in the literature inspired this thesis, demonstrating that there is presently an issue in 5G that requires academic attention.

Additionally, was calculated the complexity of provided QoE-NN architectures. However, the dissertation used a small database to evaluate the result required to more precisely utilise work with different databases. Therefore, evaluation on other datasets and real-time implementation can be completed in future work. Furthermore, in the thesis, there were only three types of QoE models, and there are plenty of variations of NNs that can be performed to compare better QoE prediction models for video streaming.

In this thesis, multiple tasks have been performed:

- Different NN has been studied and analyzed.
- The modern complexity approach has been learned and implemented
- Three additional NN was proposed, namely, FNN-QoE, combined CNN and biLSTM QoE, and biLSTM-QoE.
- It was found that FNN gives the best performance of all NN. Comparing to LSTM-QoE prediction, FNN-QoE performed better by 14%.
- On the other hand, the FNN model with an increasing number of neurons provides a similar complexity level to the recurrent LSTM.
- It was shown that the compromise among the multiplication complexity of NN and capacity accuracy has to be taken into account. Therefore, this problem has to be addressed when proposing and applying novel NNs for QoE prediction.

ABBREVIATION

Abbreviation	Explanation	Abbreviation	Explanation
RTMP	Real-Time Messaging Protocol	**STSQ**	Short Time Subjective Quality
RTSP	Real-Time Streaming Protocol	**MDP**	Media Presentation Description
ETSI	European Telecommunications Standards Institute	**PI**	Playback Indicator
QoE	Quality of Experience	**VQA**	Video Quality Assessment
ISP	Internet Service Provider	**TCN**	Temporal Convolutional Network
OTT	Over-The-Top	**STRRED**	Spatio-Temporal Reduced Reference Entropic Differences
QoE	Quality of Experience	**SRRED**	Spatio Reduced Reference Entropic Differences
DASH	Dynamic Adaptive Video Streaming over HTTP	**TRRED**	Temporal Reduced Reference Entropic Differences
ML	Machine Learning	**DL**	Deep Learning
MEC	Multi-access Edge Computing	**STSQ**	Short Time Subjective Quality
LSTM	Long Short-Term Memory	**MDP**	Media Presentation Description
CNN	Convolutional Neural Network	**PI**	Playback Indicator
NN	Neural Network	**VQA**	Video Quality Assessment

ANN	Artificial Neural Network	**TCN**	Temporal Convolutional Network
RNN	Recurrent Neural Network	**STRRED**	Spatio-Temporal Reduced Reference Entropic Differences
CDN	Content Delivery Network	**SRRED**	Spatio Reduced Reference Entropic Differences
DNN	Deep Neural Network	**PCC**	Pearson Correlation Coefficient
FNN	Feed-forward Neural Network	**SROCC**	Spearman Rank Order Correlation Coefficient
GRU	Gated Recurrent Units	**RMSE**	Root Mean Squared Error
CDVL	Consumer Digital Video Library		

REFERENCES

[1] Ericsson, 2021. Ericsson Mobility Report June 2021. [online] Ericsson.

[2] Cisco, 2021. Cisco Annual Internet Report (2018–2023). [online] Cisco.

[3] Guo, Y., Yu, F., An, J., Yang, K., Yu, C. and Leung, V., 2020. Adaptive Bitrate Streaming in Wireless Networks With Transcoding at Network Edge Using Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology, [online] 69(4), pp.3879-3892.

[4] Guo, Y., Yu, F., An, J., Yang, K., Yu, C. and Leung, V., 2020. Adaptive Bitrate Streaming in Wireless Networks With Transcoding at Network Edge Using Deep Reinforcement Learning. IEEE Transactions on Vehicular Technology, 69(4), pp.3879-3892.

[5] Kontothanassis, L., Sitaraman, R., Wein, J., Hong, D., Kleinberg, R., Mancus, B., Shaw, D. and Stodolsky, D., 2004. A transport layer for live streaming in a content delivery network. Proceedings of the IEEE, 92(9), pp.1408-1419.

[6] Feng-hui, H., Wen-an, Z. and Yu, D., 2021. QoE Issues of OTT Services over 5G Network. [online] IEEE Xplore.

[7] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469-492, Firstquarter 2015, DOI: 10.1109/COMST.2014.2360940.

[8] Malik, S., 2020. Moving Toward 5G: Significance, Differences, and Impact on Quality of Experience. IEEE Consumer Electronics Magazine, 9(6), pp.9-14.

[9] Oyman, O. and Singh, S., 2012. Quality of experience for HTTP adaptive streaming services. IEEE Communications Magazine, 50(4), pp.20-27.

[10] European Telecommunications Standards Institute, 2021. Universal Mobile Telecommunications System (UMTS); LTE; Study on improved streaming Quality of Experience (QoE) reporting in 3GPP services and networks (3GPP TR 26.909 version 14.0.0 Release 14). 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - FRANCE: European Telecommunications Standards Institute.

[11] Eswara, N., Ashique, S., Panchbhai, A., Chakraborty, S., Sethuram, H., Kuchi, K., Kumar, A. and Channappayya, S., 2020. Streaming Video QoE Modeling and Prediction: A Long Short-Term Memory Approach. IEEE Transactions on Circuits and Systems for Video Technology, 30(3), pp.661-673.

[12] Xu, Z. and Zhang, A., 2019. Network Traffic Type-Based Quality of Experience (QoE) Assessment for Universal Services. Applied Sciences, 9(19), p.4107.

[13] Mehrabi, A., Siekkinen, M. and Yla-Jaaski, A., 2018. QoE-Traffic Optimization Through Collaborative Edge Caching in Adaptive Mobile Video Streaming. IEEE Access, 6, pp.52261-52276.

[14] Duc, T., Minh, C., Xuan, T. and Kamioka, E., 2020. Convolutional Neural Networks for Continuous QoE Prediction in Video Streaming Services. IEEE Access, 8, pp.116268-116278.

[15] BEN LETAIFA, A., 2018. An adaptive machine learning-based QoE approach in SDN context for video-streaming services. TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, 26(6), pp.2859-2871.

[16] Agarwal, P. and Alam, M., 2020. A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices. Procedia Computer Science, [online] 167, pp.2364-2373.

[17] McClellan, M., Cervelló-Pastor, C. and Sallent, S., 2020. Deep Learning at the Mobile Edge: Opportunities for 5G Networks. Applied Sciences, [online] 10(14), p.4735.

[18] Wang, Y. and Friderikos, V., 2020. A Survey of Deep Learning for Data Caching in Edge Network. Informatics, 7(4), p.43.

[19] Koul, A., Ganju, S. and Kasam, M., 2020. Practical Deep Learning for Cloud, Mobile, and Edge Real-World AI and Computer-Vision Projects Using Python, Keras, and TensorFlow. 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc.

[20] Sondhi, D., 2017. Application of Data Mining in Census Data Analysis using Weka. International Journal of Engineering Trends and Technology, 52(3), pp.157-161.

[21] Kravchenko, S., Grishkun, E. and Vlasenko, O., 2020. CLASSIFICATION METHODS FOR MACHINE LEARNING USING THE SCIKIT-LEARN LIBRARY. Scientific notes of Taurida National VI Vernadsky University. Series: Technical Sciences, 1(3), pp.121-125.

[22] Syed, M., 2020. Overview on Open Source Machine Learning Platforms-TensorFlow. SSRN Electronic Journal,

[23] Carneiro, T., Medeiros Da Nobrega, R., Nepomuceno, T., Bian, G., De Albuquerque, V. and Filho, P., 2018. Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. IEEE Access, 6, pp.61677-61685.

[24] Dl.acm.org. 2021. HTTP/2-Based Methods to Improve the Live Experience of Adaptive Streaming | Proceedings of the 23rd ACM international conference on Multimedia. [online] Available at: <https://dl.acm.org/doi/10.1145/2733373.2806264>

[25] Bampis, C., Li, Z., Katsavounidis, I., Huang, T., Ekanadham, C. and Bovik, A., 2021. Towards Perceptually Optimized Adaptive Video Streaming-A Realistic Quality of Experience Database. IEEE Transactions on Image Processing, 30, pp.5182-5197.

[26] Bampis, C., Li, Z., Moorthy, A., Katsavounidis, I., Aaron, A. and Bovik, A., 2017. Study of Temporal Effects on Subjective Video Quality of Experience. IEEE Transactions on Image Processing, 26(11), pp.5217-5231.

[27] Soundararajan, R. and Bovik, A., 2013. Video Quality Assessment by Reduced Reference Spatio-Temporal Entropic Differencing. IEEE Transactions on Circuits and Systems for Video Technology, 23(4), pp.684-694.

[28] GitHub. 2021. GitHub - lfovia/lstm_qoe. [online] Available at: <https://github.com/lfovia/lstm_qoe>

[29] Freire, P., Osadchuk, Y., Spinnler, B., Napoli, A., Schairer, W., Costa, N., Prilepsky, J. and Turitsyn, S., 2021. Performance Versus Complexity Study of Neural Network Equalizers in Coherent Optical Systems. Journal of Lightwave Technology, 39(19), pp.6085-6096.