# Video QoE Inference with Machine Learning

**Conference Paper** · June 2021

DOI: 10.1109/IWCMC51323.2021.9498579



**3 authors**, including:

Abdelhak Bentaleb
Concordia University Montreal
**62** PUBLICATIONS **1,202** CITATIONS
SEE PROFILE

S. Harous
University of Sharjah
**228** PUBLICATIONS **2,034** CITATIONS
SEE PROFILE

# Video QoE Inference with Machine Learning

Tisa-Selma*, Abdelhak Bentaleb§, and Saad Harous*
*United Arab Emirates University, §National University of Singapore
{tisa,harous}@uaeu.ac.ae, bentaleb@comp.nus.edu.sg

***Abstract*—HTTP adaptive streaming (HAS) has become the de-facto standard for delivering video over the Internet. More content providers like YouTube and Twitch have started generating and delivering high quality streams (usually 4k resolution) with advanced end-to-end encryption mechanisms. This huge increase in HAS encrypted traffic, creates a significant challenge for network providers in understanding what is happening on their infrastructures which limits their ability to manage network infrastructures properly. Due to such invisibility, the network providers could not take appropriate decisions for better optimizations, resulting in significant revenue lost. Inferring the quality of experience (QoE) of HAS-based streaming video services is important, but recent studies highlight that most of existing solutions that rely on packet inspections, showing low performance in inference accuracy. To address this issue, we develop a machine learning powered system that infers QoE factors such as startup delay, rebuffering and selected quality, for encrypted on-demand HAS streaming video services. Our solution uses two data-driven techniques: Deep Self Organizing Map (DSOM) and Multi Layer Perceptron Backpropagation (MLPB), allowing efficient accuracy with low error in inferring QoE factors over several public video datasets, compared to some state-of-the-art approaches.**

***Index Terms*—QoE, encrypted video traffic, inferring QoE, deep learning, Machine learning, SOM, MLPB.**

## I. Introduction

Security, privacy, and quality of experience (QoE) are some of the requirement for end user satisfactions. Currently, most of the available video streaming services over Internet use encryption to serve secure and private video sessions for their users. Because of such encryption, network operators face limitation to monitor, manage, and react accordingly to QoE impairments. To deal with encryption issues, there are two potential solutions for the QoE inference [1]: learning-based and session-based. Session-based solution mainly uses deep packet inspection (DPI) which shows many limitations during real-time QoE inference of encrypted video traffic due to advances in encryption techniques [2].

Quality of Service (QoS) shifted to QoE measurement due to the changing in service point of view. While QoS depends mostly on the measurement of packet loss, jitter (delay variance), delay and throughput, which are objective and can be calculated based on the network traffic. QoE is mainly end-user centric where low packet loss, low jitter, low delay, and high throughput do not necessary translate to the best user experience and engagement. As highlighted in MUX report [3], 85.1% of viewers stop watching the video due to too long waiting loading time, 57.3% of them due to poor quality, 85% of them due to rebuffering events, and 67.5% of them due to repeated playback errors. Although many algorithms have been applied to handle these problems, the number of user engagement are still low. These problems are frequently occurring mainly because of the poor or hard understanding on what is happening in the delivery path due to encryption as well as lack of coordination between content providers and network operators to improve user experience of their customers as mutual benefits for both.

To address the above mentioned problems for QoE inference, our main contributions are described as follow:

- We develop a robust machine learning based solution to infer QoE factors such as startup delay, rebuffering events, and video quality (bitrate selected). The main core of our solution is enabling the power of : $(i)$ unsupervised learning in the form of Deep Self Organizing Map (DSOM), and $(ii)$ supervised learning in the form of Multi Layer Perceptron Backpropagation (MLPB), to perform video traffic clustering and classification tasks for achieving high accuracy during the QoE inference process in real-time.
- We list the possible limitations and enhancements from several current published works in video quality inference that utilize deep learning and its variances.
- We provide a practical implementation of the proposed solution for QoE inference problem of YouTube encrypted traffics.
- We completed our evaluations through extensive trace-driven using twelve models that includes regression and deep learning models on several public Youtube-based video datasets. The results show that in all considered datasets, the proposed solution resulted in competitive results compared to existing recent models.

The rest of the paper is organized as follows: recent related work on QoE inference is given in Section II. Section III describes our solution and its design. Experimental evaluation, results analysis, limitations and potential enhancements for the future are highlighted is Section IV. Section V concludes the paper with possible future direction.

## II. Related Work

To understand the relationship between QoE and QoS, Xu *et al.* [4] developed a deep belief network to find and build QoE/QoS correlation model considering metrics like: jitter, delay, and packet loss, for each video streaming service over a satellite network. Similarly, Malekmohammadi *et al.* [5] used multilayer back propagation algorithm to find the correlation between QoS and QoE metrics for various on-demand video

streaming services. Zheng *et al.* [6], designed a QoE assessment method over 3G LET networks. The proposed solution uses particle swarm optimization and back propagation neural network (NN), where the former is used as error reducer between estimated models and real mean opinion score (MOS) in NN weights post-processing. They claimed that traditional NN does not perform well as indicated by low coverage and accuracy. However, recent work [7] in fully connected back propagation NN found four layers are optimal with 10 to 30 neurons to deliver better quality compared to existing works.

Cheng *et al.* [8], proposed a forward error correction (FEC) model termed deep reed Solomon (DeepRS) which aims to estimate future packet loss from receiver feedback using deep NN. DeepRS is also able to calculate redundant packet amount while running its algorithms to encode video block. A recent work on QoE measurement and inference called CNN-QoE is proposed by Nguyen *et al.* [9]. They showed that their proposed model outperformed several methods including NLSS-QoEe [10], SVR-QoE [11], and NARXe [12], on several QoE-based datasets such as LFOVIA [11], LIVE Netlfix [12], and LIVE Mobile [13]. Another notable work by Forest *et al.* [14] named Deep Embedded Self Organizing Map (DESOM) that combines learning representation and code vector in SOM. DESOM uses autoencoder and a custom SOM layer that work together to maximize the accuracy during the training phase. The proposed model is used in unsupervised clustering tasks and can achieve more than 93.9% accuracy due to such joint representation between learning and clustering.

Schmitt *et al.* [15] used ten seconds bin that contains complete video segment downloads to have better precision of inferring resolution. They used network and application layer to get best accuracy inference for all services such as Netflix, YouTube and Twitch metrics and focused on player and buffer conditions, playback position, available buffer and resolutions. Similarly, Gutterman *et al.* [16] proposed an QoE inference system termed Requet that aims to predict events that lead to QoE impairment ahead of time and the current video resolution. Requet predicts four playback buffer states: buffer increase, buffer decay, steady, and stall. It is based on machine learning models to predict QoE metrics in real-time: buffer warning, video state, and video resolution. Emimics is proposed in [2] which relied on passive network traffic to infer real-time key metrics that influence video QoE.

Recent work in video quality inference is CSI [17] which develops Chunk Sequence Inferencer for active measurements and infer video streaming adaptation behavior. It can deliver third party mobile video services with encrypted HTTPs such as Netflix (TCP) or Youtube (QUIC). CSI infers video quality and rebuffering events by analyzing the identity of each downloaded object (*e.g.,* the index, track, audio or video chunk). For some services, CSI assumes that all objects as video chunk due to the lack of visibility. It also considers constant and variable bitrate variations. Wu *et al.* [18] developed a solution to identify video stream services (*e.g.,* YouTube video) leveraging Application Data Unit (ADU). They utilized Adaptive Boosting, Random Forest, Decision Tree, Naive Bayes and K-Neighbors separately to classify the traffics. Another work published by Bronzino *et al.* [19] has notable benefits such as its high inference accuracy and shorter time inference. They developed general methods to infer streaming video QoE metrics from encrypted traffic using lightweight features from popular services (*e.g.,* YouTube, Netflix, Amazon Instant Video, Twitch). Their methods are to generate labeled video traffic and resulting dataset to evaluate video quality inference. Next, they computed the input matrices to get the set of features and build the models to infer the targets output matrices.

## III. Methodology And Architecture

In a nutshell, our model training and testing setup for both DSOM and MLPB is depicted in Figure 1. We used the set of features that are extracted from various publicly available YouTube datasets as an input during models training. Then, we captured YouTube video streaming traffic using *Wireshark*[1] and extract features in real-time during the testing phase. For the output, we checked whether the test attribute will belong to one of our target classes or not. Moreover, we calculate the classification accuracy by dividing the correctly classified cases and total cases. Next, we present the detail of DSOM and MLPB models.

### A. DSOM Model

DSOM is an extension of Self Organizing Map (SOM) [20] with deep neural network (NN). DSOM is an unsupervised learning technique for clustering and classification tasks that can be used as a dimensionality reduction. It is feed forward NN which discretizes representation of high dimensionality input training data into lower dimensionality data. DSOM differs with other NN-based techniques as it utilizes competitive learning rather than error correction learning like in Multi-Layer Perceptron Backpropagation (MLBP). It also applies neighborhood function to preserve topological properties of the input space. DSOM shows a great performance in many computer vision applications and it is a good candidate for QoE metrics inference. It can simplify complex information as well as learn by updating weights depending on input features. Regular DSOM used euclidean distance which is not suited for high dimensionality or data clustering. For this reason, we used Manhattan distance instead for the QoE inference task. In addition, during the training phase, we fixed the learning rate (denoted by $\alpha$) to 0.6 and cluster number to two with 15x15 nodes. The main insights of this selection is to learn the model in fast fishing without deviating much from the optimal weights to be selected. We have also fixed the epochs to 20000 (stop criterion) as after this value their were no accuracy improvement. Our setup for DSOM is described briefly as follows:

*1) Training Phase:* During the training phase, we used four public Youtube-based over QUIC datasets: one from Mazhar *et al.* [21] and three from Muexlab [22], [23]. Each

[1]https://www.wireshark.org/

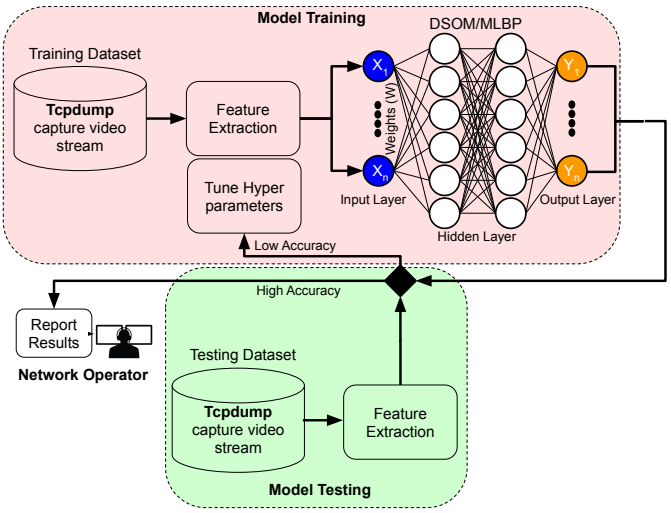


Figure 1. Training and Testing Architecture

dataset consists of various video traffic features from network- and transport-layers (in CSV format) that are collected using *Wireshark* (through Tcpdump[2]). During the data collection, the authors of these datasets have used *tc-netem*[3] to shape the network bandwidth between the Youtube player and its content delivery networks for a realistic last-mile network. First, the system extracts important video traffic features from each dataset which are then given as input to our DSOM model. Then, it uses Xavier initialization [24] for initial weights selection to reduce the training time. This initial weights selection technique sets a layer's weights to values chosen from a random uniform distribution.

While we give the initial weights, the system will select randomly which input node will be processed first. Second, the system calculates Manhattan distance between the initial weights in each and every node and the considered features. Furthermore, the system retains all the Manhattan distances and compare them altogether to get the lowest distance. The lowest distance means the closer to the real data. Therefore, the winning node is the node that has the lowest distance value. After selecting the winning node, the system updates all the weights of all neighbors nodes according to the winning node for each epoch $t$ as follows:

$$w_{j,i} = \alpha(t) \times T_{j,I(x)}(t) \times (x_i - w_{j,i}),$$

where, $w$ is a weight, $x$ is an input attribute, $i$ is a node (neuron), $j$ is a neighbor node, $I(x)$ is the winning node, $\alpha$ is the learning rate, $S$ is a distance between neuron $i$ to $j$ and $T()$ is topological neighborhood. $\alpha(t)$ and $T_{j,I(x)}(t)$ with their hyperparameters ($\alpha$, $\sigma$, and $\tau$) are given as follows:

$$\begin{cases} \alpha(t) = \alpha_0 \times e^{\frac{-t}{\tau_\alpha}} \text{ and } T_{j,I(x)}(t) = e^{\frac{-S^2_{j,I(x)}}{2\sigma(t)^2}} \\ S_{j,i} = ||w_j - w_i||, \sigma(t) = \sigma_0 e^{\frac{-t}{\tau_0}} \end{cases}$$

[2]https://www.tcpdump.org/

[3]https://man7.org/linux/man-pages/man8/tc-netem.8.html

*2) Testing Phase:* For each dataset, we randomized and divided them into two sets: 80% for training and 20% for testing. With 80-20 train-test split, we performed 5-fold and 10-fold walk-forward cross-validation on each dataset and our results of the DSOM model are shown in Section IV. In addition, we performed the QoE inference in real-time using our Youtube's traffic collection to test the validity of our models.

### B. MLPB Model

We followed similar steps (Section III-A) for training and testing the proposed MLPB model [25]. For training this model, we used ten total attributes divided by two for hidden layers. The main insight behind MLPB is the partial derivative of cost function which is equal to partial derivative of certain cost divided by partial derivative of weight $C$ ($\frac{\partial C}{\partial w}$). In our MLPB model, we used the following three notations: $w^l_{j,i}$ is denoted by the weight in $i$ node in the $(l-1)$ layer to $j$ node in the $l$ layer, $b^l_j$ is denoted by the biases of $j$ node in $l$ layer, and $a^l_j$ is denoted by activation on $j$ node in $l$ layer. The function $C$ shows how the change in weights and biases will impact drastically the cost, and thus the overall NN performance. Such capability makes the MLPB as one of fast-learner-algorithm [19], [25]. The reasons behind this outcome are input and output nodes condition. If the input node is low-activated, and/or the output node is saturated, then it will affect the node to be a slow learner. While errors are being propagated to output. The MLPB model strives to minimize the error as well as the cost by checking which nodes, weights, bias and activation that are among the biggest error contributors and fix them to meet the better performance. In our context, we used MLPB model with the cost gradient descent function as described in [25].

## IV. Experimental Results

To evaluate the QoE inference effectiveness of the proposed solution (DSOM and MLPB), we use random split 80% of the four datasets (Mazhar *et al.* [21] and Muexlab [22], [23]) to train twelve models to compare with. These models are: DNN, GRU, LSTM, Random Forest (RF), Gradient Boosting Regressor (GBR), Lasso, ElasticNet (EN), Ridge, Logistic Regression (LR), Perceptron, Passive Aggressive Classifier (PAC), Grave LSTM (GLSTM), and Mazhar *et al.* [21] model (referred to as Base). We implemented all models in python v3.7 with *tensorflow*[4] v2.1.0. All the hyperparameters used in these experiments are based on trial and error. We have tried 5-fold and 10-fold cross-validations, {50-50, 60-40, 70-30, 80-20 and 90-10}% training-testing split and tried using {10000, 20000, 200000} epochs to train the model. We picked up only the best values of each hyperparameter that can achieve the highest possible results. Number of layers are fixed based on the standard of each model. The accuracy of each model in the four datasets are given by the mean absolute error (MAE) as highlighted in Table I. In addition, we used population-based

[4]https://www.tensorflow.org/

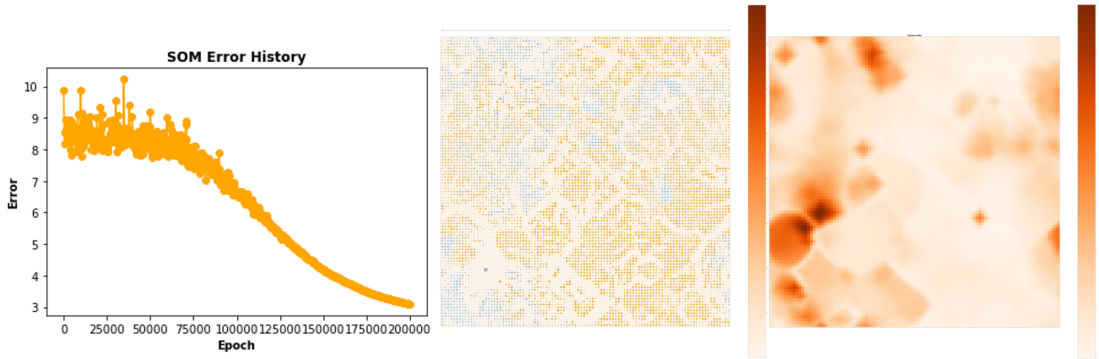


Figure 2. Training error, distance map and point map of HTTPS rebuffering events in Mazhar *et al.* [21] dataset. The distance map means clustering map of classes, meanwhile the darker color of point map means the worse accuracy and vice versa.

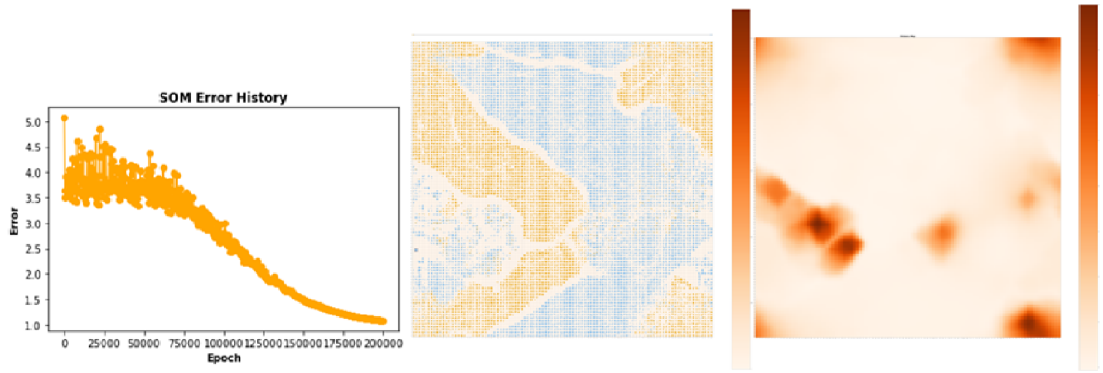


Figure 3. Training error, distance map and point map of QUIC rebuffering events in Mazhar *et al.* [21] dataset.The distance map means clustering map of classes, meanwhile the darker color of point map means the worse accuracy and vice versa.

convergence [26] to evaluate the DSOM model. Specifically, we used accuracy percentage to compare between generated clusters in training map and clusters in testing phase. The average results of accuracy percentage is given in Figures. 2–3 and MAE results are highlighted in Table I.

Figures. 2–5 highlighted the training error, distance map and point map rate graph to two QoE metrics rebuffering events and startup delay in Mazhar *et al.* [21] dataset. The better model is the one that has the lowest error both in training and testing. The lowest error on experiment observed is achieved after 10000 epoch. Since DSOM using Xavier random uniform weights initialization, each single run will have different results compared to other in a same model and dataset as well as different training time. As we can see in point map (right), the lighter color in the map means the better performance. Further, in distance map (middle), we can see two classes 0 (in blue; no rebuffering or no startup delay) and 1 (in orange, rebuffering or startup delay). Two main observations can be drawn from these figures: ($i$) The DSOM model is able to reach the best performance (the lowest possible error) only after 200000 epochs (stopping criterion), ($ii$) The DSOM is able to detect the startup delay efficiently compared to rebuffering events.

Table I highlights the MAE of different QoE metrics inference corresponding to each dataset. In general, MLPB and DSOM can be used as an alternative unsupervised learning approach compared to other models. We found that DSOM is more suitable to be used in numerical datasets, meanwhile MLPB is more suitable to be used in combination of numerical and string value of data. They can be used specifically in high dimensional data. The benefit of DSOM over MLPB is that DSOM is able to reduce dimensionalities and detect neighboring label. On the contrary, MLPB can only detect the set of target labels under consideration. In our experiments, DSOM gets up to on training set and gets lower on testing due to class overfitting. The maximum accuracy achieved by MLPB and DSOM on testing phase are up to 95.19% and up to 100% with average 78.43% and 32.96% and the worst accuracy down to 32.37% and 0%, respectively.

In the future, we are planning to improve our models to avoid overfitting issues through more advanced regularization techniques. The best performance is achieved by MLPB and followed by DSOM compared to other models. MLPB is able to achieve more than 99% accuracy while DSOM can achieved 98% accuracy improvement on average among the four datasets compared to other models. MLPB works better in inferring some QoE metrics (Muexlab [23]) while not in other (Mazhar *et al.* [21]). Each MLPB or DSOM has its own merit in sense that both models can complete each other to get good alternative unsupervised learning solution in various datasets with different characteristics.

## V. Conclusion

Due to the widespread deployment of encrypted video traffic by most service providers, network operators face a serious challenge in understanding the main causes of performance

Table I
MAE ACROSS DIFFERENT QOE INFERENCE MODELS AND DATASETS.

| | DNN | GRU | RF | GBR | Ridge | Lasso | EN | LR | Perceptron | PAC | LSTM | GLSTM | MLPB | DSOM | Base |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mazhar *et al.* [21] (SD: Startup Delay, QT: Quality, RE: Rebuffering Events) | | | | | | | | | | | | | | | |
| SD 3s (HTTPS) | 0.613 | 0.482 | 0.255 | 0.249 | 0.25 | 0.25 | 0.25 | 0.21 | 0.209 | 0.21 | 0.192 | 0.2664 | 0.2956 | 0.3333 | 0.1991 |
| SD 6s (HTTPS) | 0.582 | 0.523 | 0.28 | 0.287 | 0.29 | 0.298 | 0.298 | 0.27 | 0.33 | 0.314 | 0.2213 | 0.277 | 0.2773 | 1.783 | 0.2218 |
| SD 10s (HTTPS) | 0.67 | 0.544 | 0.28 | 0.285 | 0.297 | 0.292 | 0.29 | 0.27 | 0.41 | 0.275 | 0.2851 | 0.2978 | 0.1971 | 0.0016 | 0.232 |
| QT (HTTPS) | 0.648 | 0.518 | 0.43 | 0.47 | 0.57 | 0.565 | 0.56 | 0.589 | 0.585 | 0.798 | 0.5373 | 0.1427 | 0.6763 | 1.0672 | 0.144 |
| RE (HTTPS) | 0.798 | 0.529 | 0.131 | 0.164 | 0.208 | 0.212 | 0.212 | 0.22 | 0.405 | 0.18 | 0.2067 | 0.2134 | 0.2901 | 0.4147 | 0.102 |
| SD 3s (QUIC) | 0.77 | 0.516 | 0.186 | 0.202 | 0.26 | 0.262 | 0.26 | 0.19 | 0.275 | 0.19 | 0.1642 | 0.2308 | 0.1868 | 0.2674 | 0.1525 |
| SD 6s (QUIC) | 0.724 | 0.476 | 0.255 | 0.269 | 0.3131 | 0.2285 | 0.313 | 0.228 | 0.298 | 0.226 | 0.3376 | 0.2856 | 0.3412 | 0.4352 | 0.2109 |
| SD 10s (QUIC) | 0.585 | 0.535 | 0.274 | 0.289 | 0.3288 | 0.3299 | 0.329 | 0.23497 | 0.425 | 0.2495 | 0.3792 | 0.2206 | 0.3381 | 0.4791 | 0.224 |
| QT (QUIC) | 0.384 | 0.386 | 1.09 | 1.159 | 1.301 | 1.303 | 1.302 | 0.6033 | 1.423 | 1.54 | 1.2164 | 0.247 | 0.2374 | 1.895 | 0.277 |
| RE (QUIC) | 0.646 | 0.351 | 0.257 | 0.2648 | 0.303 | 0.305 | 0.3048 | 0.2384 | 0.2496 | 0.264 | 0.77692 | 0.29 | 0.3068 | 0.4084 | 0.207 |
| Muexlab [22], [23] (BT: Bitrate, RE: Rebuffering Events, MC: MOS Class) | | | | | | | | | | | | | | | |
| BT (Android 2018 CNSM) | 0.645 | 0.459 | 0.2203 | 0.211 | 124.229 | 95.377 | 94.9 | 1.1 | 0.898 | 1.01 | 0.3544 | 0.2823 | 0.1045 | 0.5556 | 0.1201 |
| BT (iOS 2018 CNSM) | 0.877 | 0.45 | 0.2042 | 0.2105 | 0.241 | 0.241 | 0.239 | 0.623 | 0.6103 | 1.285 | 0.4328 | 0.6613 | 0.1223 | 0.4444 | 0.1123 |
| RE (Android 2018 QoMex) | 0.86 | 0.427 | 0.2014 | 0.183 | 45.179 | 46.831 | 46.89 | 0.86 | 0.139 | 0.36 | 0.2039 | 0.2206 | 0.1849 | 0.3333 | 0.2938 |
| BT (QoMex) | 0.877 | 0.393 | 0.06 | 0.058 | 0.126 | 0.13 | 0.131 | 0.009 | 0.177 | 0.177 | 0.22644 | 0.2183 | 0.0481 | 1 | 0.0079 |
| MC (Android+iOS QoMex) 2019 | 0.254 | 0.355 | 0.266 | 0.271 | 0.498 | 0.49 | 0.46 | 0.408 | 1.391 | 0.541 | 0.3523 | 0.3455 | 0.08 | 0 | 0.0645 |
| Player Abandon | 0.778 | 0.505 | 0.036 | 0.008 | 0.523 | 0.179 | 0.2376 | 0.11 | 0.166 | 0.2521 | 0.2521 | 0.1297 | 0.1954 | 1 | 0.0278 |
| Multiple Players | 0.933 | 0.512 | 0.712 | 0.026 | 4.665 | 0.423 | 0.461 | 0 | 0.38 | 0.05 | 0.6725 | 0.1721 | 0.0584 | 0 | 0.034 |
| Player Pause | 0.656 | 0.546 | 0.007 | 1.8194 | 2.176 | 1.368 | 1.309 | 0.38 | 0.94 | 0.38 | 0.5394 | 0.3366 | 0.4152 | 0 | 0.023 |
| Player Seek | 0.3 | 0.573 | 0.017 | 1.86 | 1.002 | 0.282 | 0.294 | 0.16 | 0.777 | 0.55 | 0.696 | 0.3234 | 0.2722 | 1 | 0.0278 |
| Player Speed | 0.776 | 0.707 | 0.1809 | 0.135 | 2.268 | 0.251 | 0.238 | 0.4 | 0.9 | 0.5 | 0.434 | 0.4488 | 0.0954 | 1 | 0.0409 |

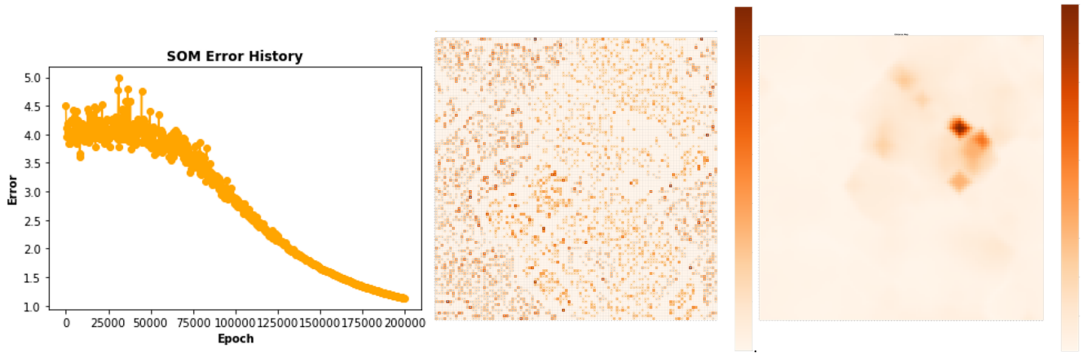


Figure 4. Training error, distance map and point map of HTTPS startup delay of 3 seconds in Mazhar *et al.* [21] dataset.

degradation and taking the suitable decisions to manage their network infrastructures. Many factors that influence viewers need to be taken care of to get an acceptable user experience. In this paper, we proposed a QoE inference system that combines two machine learning techniques for HTTP adaptive streaming (HAS) services. In particular, our solution leverages the capabilities of deep self organizing map (DSOM) and multi-layer perceptron backpropagation (MLPB). Experimental evaluation confirmed the efficiency of the proposed solution. Our trace-driven experiments on popular Youtube datasets confirm that our system can get accuracy up to 99%. In the future work, we plan to extend our models to support live streaming scenarios. Moreover, we also plan to collect our own dataset from various well-know video streaming services including Twitch and Netflix in various environments and network conditions.

## ACKNOWLEDGMENT

This research work is supported by UAEU Grant: 31T102-UPAR-1-2017.

## REFERENCES

[1] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Using session modeling to estimate http-based video qoe metrics from encrypted network traffic," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 1086–1099, 2019.

[2] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Emimic: Estimating http-based video qoe metrics from encrypted network traffic," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–8, IEEE, 2018.

[3] "2017 VIDEO STREAMING PERCEPTIONS REPORT kernel description." https://static.mux.com/downloads/2017-Video-Streaming-Perceptions-Report.pdf. Accessed: 2020-08-15.

[4] S. Xu, X. Wang, and M. Huang, "Modular and deep qoe/qos mapping for multimedia services over satellite networks," *International Journal of Communication Systems*, vol. 31, no. 17, p. e3793, 2018.

[5] H. Malekmohamadi, W. Fernando, E. Danish, and A. M. Kondoz, "Subjective quality estimation based on neural networks for stereoscopic videos," in *2014 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 107–108, IEEE, 2014.

[6] K. Zheng, X. Zhang, Q. Zheng, W. Xiang, and L. Hanzo, "Quality-of-experience assessment and its application to video services in lte networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 70–78, 2015.

[7] J. Guo and S. Wan, "Quality assessment for networked video streaming based on deep learning," in *International Conference on Internet of Things as a Service*, pp. 90–97, Springer, 2018.

[8] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang, "C3: Internet-scale control plane for video quality

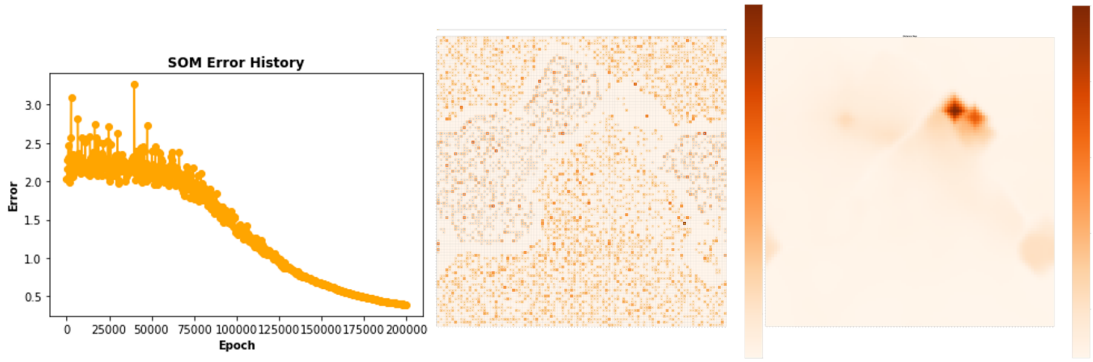


Figure 5. Training error, distance map and point map of QUIC startup delay of 3 seconds in Mazhar *et al.* [21] dataset.

optimization," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pp. 131–144, 2015.

[9] N. Eswara, H. P. Sethuram, S. Chakraborty, K. Kuchi, A. Kumar, and S. S. Channappayya, "Modeling continuous video qoe evolution: A state space approach," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2018.

[10] N. Eswara, K. Manasa, A. Kommineni, S. Chakraborty, H. P. Sethuram, K. Kuchi, A. Kumar, and S. S. Channappayya, "A continuous qoe evaluation framework for video streaming over http," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 11, pp. 3236–3250, 2017.

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

[12] C. G. Bampis, Z. Li, A. K. Moorthy, I. Katsavounidis, A. Aaron, and A. C. Bovik, "Study of temporal effects on subjective video quality of experience," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5217–5231, 2017.

[13] D. Ghadiyaram, J. Pan, and A. C. Bovik, "A subjective and objective study of stalling events in mobile streaming videos," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 183–197, 2017.

[14] F. Forest, M. Lebbah, H. Azzag, and J. Lacaille, "Deep embedded som: Joint representation learning and self-organization," *reconstruction*, vol. 500, p. 500, 2000.

[15] P. Schmitt, F. Bronzino, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *arXiv preprint arXiv:1901.05800*, 2019.

[16] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: Real-time qoe detection for encrypted youtube traffic," in *Proceedings of the 10th ACM Multimedia Systems Conference*, pp. 48–59, 2019.

[17] S. Xu, S. Sen, and Z. M. Mao, "Csi: inferring mobile abr video adaptation behavior under https and quic," in *Proceedings of the Fifteenth European Conference on Computer Systems*, pp. 1–16, 2020.

[18] H. Wu, G. Cheng, and X. Hu, "Inferring adu combinations from encrypted quic stream," in *Proceedings of the 14th International Conference on Future Internet Technologies*, pp. 1–6, 2019.

[19] F. Bronzino, P. Schmitt, S. Ayoubi, N. Feamster, R. Teixeira, S. Wassermann, and S. Sundaresan, "Lightweight, general inference of streaming video quality from encrypted traffic," 2019.

[20] T. Kohonen, *Self-organizing maps*, vol. 30. Springer Science & Business Media, 2012.

[21] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for https and quic," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1331–1339, IEEE, 2018.

[22] I. Oršolić, M. Suznjevic, and L. Skorin-Kapov, "Youtube qoe estimation from encrypted traffic: Comparison of test methodologies and machine learning based models," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, IEEE, 2018.

[23] I. Oršolić, P. Rebernjak, M. Sužnjević, and L. Skorin-Kapov, "In-network qoe and kpi monitoring of mobile youtube traffic: Insights for encrypted ios flows," in *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 233–239, IEEE, 2018.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[25] S. Seung, "Multilayer perceptrons and backpropagation learning," *9.641 Lecture4*, pp. 1–6, 2002.

[26] G. T. Breard, "Evaluating self-organizing map quality measures as convergence criteria," 2017.