Predicting QoE Factors with Machine Learning

Vladislav Vasilev, Jérémie Leguay, Stefano Paris, Lorenzo Maggi, Merouane Debbah

Predicting QoE Factors with Machine Learning

Vladislav Vasilev, Jérémie Leguay, Stefano Paris, Lorenzo Maggi, Mérouane Debbah
Mathematical and Algorithmic Sciences Lab, Paris Research Center - Huawei Technologies Co. Ltd.
Email: {name.surname}@huawei.com

***Abstract*—Classic network control techniques have as sole objective the fulfillment of Quality-of-Service (QoS) metrics, being quantitative and network-centric. Nowadays, the research community envisions a paradigm shift that will put the emphasis on Quality of Experience (QoE) metrics, which relate directly to the user satisfaction. Yet, assessing QoE from QoS measurements is a challenging task that powerful Software Defined Network controllers are now able to tackle via machine learning techniques. In this paper we focus on a few crucial QoE factors and we first propose a Bayesian Network model to predict re-buffering ratio. Then, we derive our own novel Neural Network search method to prove that the BN correctly captures the discovered stalling data patterns. Finally, we show that hidden variable models based and context information boost performance for all QoE related measures.**

Index Terms*—*Software Defined Networking, Quality of Experience, Bayesian Network, Neural Network Search Method, Graph Clustering, Hidden Variable Model

I. Introduction

According to a recent report [1], video traffic will steadily grow in the next years, representing 82% of the whole Internet traffic by 2021. Therefore, handling video traffic so as to maximize the quality perceived by final users is becoming critical both for content and network operators. To this end, Content Delivery Networks (CDNs) operators have adopted coordinated control planes [2] between routing and their streaming systems following the recent trend of Software Defined Networks (SDN) [3], which has deeply transformed the way network architectures are designed and controlled. Nonetheless, Internet Service Providers (ISPs) can also contribute to improve the perceived quality of video traffic by optimizing network resources according to the user needs. However, ISPs can only exploit coarse-grained information on video flows due to the end-to-end encryption that many Over-The-Top (OTT) operators like Facebook, Google, and Amazon employ [4]. ISPs are therefore calling for new methods for handling network resources in order to maximize the perceived quality in video services, which directly reflects the opinion customers have on the network infrastructure [5].

HTTP Adaptive Streaming (HAS), which has been standardized into MPEG-Dynamic Adaptive HTTP Streaming (DASH) [6], represents nowadays the pillar technology for video streaming over the Internet. Indeed, HAS connections can easily pass through intermediate services like NATs, gateways and proxies without the need of complex network configurations. Videos are split into temporal segments whose duration lasts from a couple up to hundreds of seconds. Each segment (also knows as chunk) is encoded at different qualities resulting in different file sizes. The availability of multiple representations for the same video segment enables DASH clients to scale up or down the video quality by simply selecting the best segment to be downloaded according to network status and video player's buffer.

The way final users perceive the quality of a streamed video depends on several factors that cannot be all measured. This perceived quality is denoted as Quality of Experience (QoE). According to [7], the user experience highly depends on three crucial factors: (i) the visual quality and its variation, (ii) the frequency and duration of re-buffering events (i.e., stalls or interruptions), and (iii) the startup delay. While the visual quality and its variation can be measured using PSNR-based metrics when traffic is not encrypted, re-buffering events and start-up delay cannot be directly measured, but only predicted from classic QoS metrics [4]. This allows to infer QoE factors by still relying on legacy, QoS monitoring systems.

Yet, the mapping between QoS and QoE metrics is highly complex, as they often lay in high dimensional spaces and are subject to noise. As a consequence, a closed form modeling and its experimental validation are not practical. We therefore resort to machine learning techniques to derive the complex relationships between QoS and QoE metrics.

On a data set produced with a high-fidelity and fully controllable simulation environment, we show that while *local* linear relationships hold for the video quality variation and network measurements, re-buffering events lay in high dimension clusters of QoS metrics. For re-buffering events, we present a Bayesian Network (BN) classifier based on two Logistic Regressions (LR) which better balances the class accuracy compared to the state of the art method based on random forests [8]. Furthermore, we demonstrate that due to the stochastic nature of re-buffering events, clusters partially overlap, hence increasing the inaccuracy of standard predictors. A pattern exploration model that we specifically design using a novel Neural Network (NN) search method confirms our intuition that other predictors incur the same or worse inaccuracy of BN-based methods. Finally, we turn our attention to hidden variables, namely metrics that cannot be directly measured but can be still inferred from QoS metrics. We show that the use of predicted hidden variables as features can indeed improve accuracy for re-buffering events. Finally, we show that if we have access to information about network congestion (e.g., number of competing sessions, QoS measures on bottlenecks) and basic characteristics on video streams (e.g., type of device, content provider) all predictions of QoE factors can be further improved.

The paper is structured as follow. Sec. II discusses relevant related work. Sec. III describes the problem of predicting QoE from QoS measurements and the data set we produced. Sec. IV illustrates our method to classify and predict re-buffering events, while Sec. V presents results for the video quality and its variation. Finally, Sec. VI concludes our paper.

II. Related Work

Quality of Experience (QoE) has recently gained momentum as a way to assess the user opinion of the network quality while watching videos. An additive log-logistic model that maps video quality, freezing (i.e., stall of the video session), and image artifacts due to compression and re-buffering events into a QoE score has been firstly proposed in [9] and successively adopted by ITU in the Recommendation P.1202.2 as a reference model for quantifying QoE [10]. The investigation performed in [11] on how a user perceives the video quality and the main factors that influence this perception resulted in the definition of eight mathematical models of QoE. Studies like the one presented in [12] provide quantitative methods to measure the distortion of the received bit-stream due to video quality and freezing. While different in the way they compute a score for measuring QoE, all these works agree on three main impairments that affect the QoE, namely re-buffering events, the video quality and its variation. Furthermore, due to the psychological effect known as *memory effect*, the repetition of the same impairment during the video session such as the experience of multiple video stalls due to re-buffering strongly affects the quality perceived by the final user [13]. For this reason, both client-side and network-side mechanisms [7], [14], [15] have been recently proposed to prevent or at least minimize re-buffering events and video quality variations.

Existing client-side DASH adaptation policies base their decisions on several network performance and the internal client state. Rate-Based (RB) policies base their decisions on the measured download throughput, whereas Buffer-Based (BB) [14] approaches use the level of the buffer containing the downloaded segments to decide the quality of the next chunk. A number of hybrid approaches also exist, where the explicit formulation of the optimization problem [7] enables the use of control theoretic methods.

Machine learning has been recently used to predict QoE from network measurements [4], [16]. Dimopoulos et al. [4] shows how the rebuffering ratio, and the average video quality and its variation, can be predicted using random forests. We consider this work as a starting point for our research and present two further contributions: 1) a Bayesian Network model to predict rebuffering events with a better balance in class accuracies and 2) the evidence that additional context information on network congestion and basic characteristics of video streams improve predictions for all QoE factors.

III. From QoS to QoE factors

A. Problem Statement

We consider three main QoE factors which are commonly used to measure user-perceived video quality [5]:

- *Average video bitrate* of the downloaded segments.
- *Average video bitrate variation:* the standard deviation of the video bitrate. It quantifies quality changes over the different downloaded segments.
- *Re-buffering ratio*: freezing (or stalling) time over the duration of the video streaming session.

Our aim in this paper is to infer the three aforementioned QoE factor from the observable QoS metrics described in Tab. I using machine learning techniques.

B. Dataset Description

To build and evaluate QoS to QoE mapping functions, we have used a high-fidelity and fully controllable simulation environment at both network and streaming levels. The simulation platform is based on the Adaptive Multimedia Streaming Simulator Framework (AMust) [17] in ns-3 which implements an HTTP client and server for LibDASH, one of the reference software of ISO/IEC MPEG-DASH standard.

As streaming content, we have chosen 3 representative open movies[1] commonly used for testing video codecs and streaming protocols: Big Buck Bunny (BBB), a cartoon with a mix of low and high motion scenes, Swiss Account (TSA), a sport documentary with regular motion scenes and Red Bull Play Street (RBPS), a sport show with high motion scenes.

We have considered a star network with a bottleneck link as shown in Fig. 1, on top of which we have simulated a large number scenarios varying the number of nodes (from 1 to 100), the bottleneck capacity (from 500 kbps to 10Mbps per stream), the bottleneck delay (from 10ms to 100ms), the bottleneck packet loss (from 0% to 3%), screen resolutions and DASH policies (RB, BB and hybrid). After a month of simulations, we have obtained statistics for more than 69,000 video sessions with 50 associated variables from 4 categories: *Context* information on network congestion and stream characteristics, *QoS metrics*, *Target* QoE factors and *Hidden* QoE variables (see Tab. I for a complete list). The dataset is meant to become public.

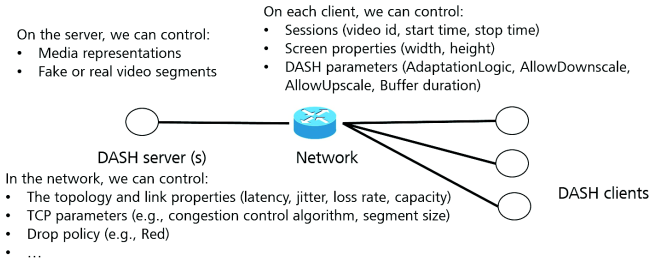


Fig. 1: Simulation environment with AMust in ns-3.

Arguably, out of the 3 target variables we want to predict RebufferingRatio is the most difficult, especially in its raw continuous form. To simplify our task we take a similar

[1] http://concert.itec.aau.at/SVCDataset/

Index	Name	Type	Description
1	RequestID	Context	Streaming session identifier
2	NbClients	Context	Maximum number of streams competing on the bottleneck
3	BottleneckBW	Context	Capacity of the bottleneck
4	BottleneckDelay	Context	Network delay on the bottleneck
5	BottleneckLoss	Context	Packet loss on the bottleneck
6	DASHPolicy	Context	DASH policy (e.g, or name of content provider)
7	ClientResolution	Context	Client screen resolution or device type (e.g., smartphone)
8	RequestDuration	QoS metric	Duration of the stream
[9, 13]	TCPOut/InPacket	QoS metric	Number of TCP packets (In and Out)
[10, 14]	TCPOut/InDelay	QoS metric	Average delay experienced by TCP packets (In and Out)
[11, 15]	TCPOut/InJitter	QoS metric	Average jitter experienced by TCP packets (In and Out)
[12, 16]	TCPOut/InPloss	QoS metric	Packet loss rate experienced by TCP packets (In and Out)
17	TCPInputRetrans	QoS metric	Packet retransmissions experienced by TCP
18	StdNetworkRate	QoS metric	Standard deviation of the network rate
[19:27]	[0,5,10,25,50,75,90,95,100] NetworkRate	QoS metric	x^{th} quantile for the network rate (measured in intervals of 2s)
28	StdInterATimesReq	QoS metric	Std. dev. of inter-arrival times of segment requests
[29:37]	[0,5,10,25,50,75,90,95,100] InterATimesReq	QoS metric	x^{th} quantile for the inter-arrival times of segment requests
38	StartUpDelay	Hidden	Initial time at the client to start playing the video
39	AvgVideoDownloadRate	Hidden	Average downloading rate for video segments
40	StdVideoDownloadRate	Hidden	Std. dev. of downloading rate for video segments
41	AvgVideoBufferLevel	Hidden	Average video buffer length.
42	StdVideoBufferLevel	Hidden	Std. dev. of video buffer length
43	StallEvents	Hidden	Number of stall events
44	**RebufferingRatio**	**Target**	Portion of time spent in stall events
45	**StallLabel**	**Target**	Discretization of RebufferingRatio variable
46	TotalStallingTime	Hidden	Total duration of stall events
47	AvgTimeStallingEvents	Hidden	Average duration of stall events
48	AvgQualityIndex	Hidden	Avg. normalized index of downloaded representations
49	**AvgVideoBitRate**	**Target**	Average video bitrate consumed by the player
50	**AvgVideoQualityVariation**	**Target**	Average variation of the video bitrate
51	AvgDownloadBitRate	Hidden	Average download rate of video segments

TABLE I: Context information, QoS metrics, hidden variables. Target QoE factors, which we want to predict from all other variables, are highlighted in bold.

approach as in [8]. The RebufferingRatio values are aggregated into 3 discrete values in a new variable *StallLabel*. Firstly, RebufferingRatio equals to 0 means that no stalling has occurred, hence we set StallLabel=*NoStall*. If it is between $(0, 0.1)$ then StallLabel=*MildStall*. Finally, if RebufferingRatio is above 0.1 then StallLabel is given the value *SevereStall*.

Fig. 2 shows the histogram of the 3 target variables. Similarly to the target variables, all other variables' distribution follow an exponential pattern. For this reason we initially apply *on the input* data a logarithmic transformation.

In the next sections, we use machine learning techniques to derive accurate QoS-QoE mappings given the available data.

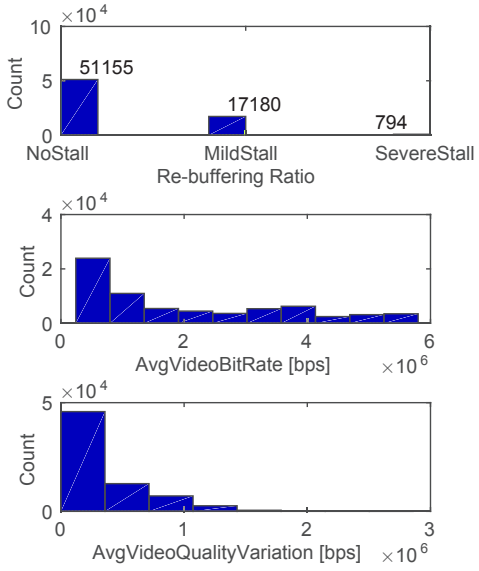


Fig. 2: Histograms for our 3 target variables.

Class	Training Accuracy	Validation Accuracy
NoStall	0.96178	0.95525
MildStall	0.7585	0.73587
SevereStall	0.43874	0.34211

TABLE II: RF class accuracies. Training to validation ratio is 4:1.

IV. Stalling Prediction

In this section we use a Bayesian Network (BN) [18] model to accurately predict the StallLabel variable from the QoS metrics listed in Tab. I. We then show that StallLabel is formed by a mixture of 2 distributions and that if there is a model that predicts accurately the true distribution of a data point we can get around 97% performance with the proposed BN model. Finally, we conjecture through a custom *novel* neural network search method that there is no such model, hence achieving higher performance with our dataset is unlikely.

As a benchmark model we use Random Forest (RF) as done in [8]. A RF is a bagging of Decision Tree (DT) models, see [19]. At each leaf node, DT greedily selects and splits an input variable into non-overlapping regions, so that the resulting new leafs gain predictive power. The bagging procedure essentially tries to minimize the effect of local optimality that stems from the greedy split procedure. Table II shows the performance of a RF classifier on the StallLabel variable pruned with minimum leaf size of 50 to prevent over-fitting and training to validation size ration of 4:1 (the same ratio is used in all the paper).

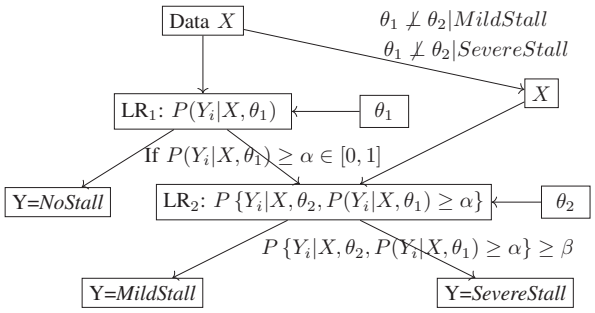
The results show that the RF is making an accurate prediction on the NoStall class of StallLabel while it has worse predictions on the MildStall. The performance on the SevereStall class is practically unacceptable. There are 2 most commonly occurring problems with RF, i.e., 1) the RF *greedy split* procedure result in low quality local optimum; 2) the RF's rectangular decision regions have boundaries parallel to the basis of the dimensions, which could fail to capture *some* dependencies among features.

For these reasons, we turn our attention to Bayesian Networks based on Logistic Regression (LR) predictors. LR is a binary classification model that maximizes the likelihood $L(\theta)$ of a target vector of binary values Y given the input data X for a given prior distribution $P(\theta)$ (assumed to be uniform in our experiments) of the parameter set θ, see Eq. (1). The a-posteriory probability model is assumed to be $P(Y_i|X,\theta) = (\sigma[\theta, X_i])^{Y_i}(1-\sigma[\theta, X_i])^{1-Y_i}$ where σ is the sigmoid function $\sigma[\theta, x] = \frac{1}{1+e^{-\theta \cdot x}}$. In Eq. (2) we report the gradient of the log-likelihood.

$$\max_{\theta} L(\theta) = \left\{ \prod_{i=1}^{n} P(Y_i|X,\theta) \right\} P(\theta) \quad (1)$$

$$\frac{\partial \log(L(\theta))}{\partial \theta} = \left\{ \sum_i X_i(\sigma[\theta \mathbf{X}_i] - Y_i) \right\} + \frac{\partial \log(P(\theta))}{\partial \theta} \quad (2)$$

Using the LR probabilistic model we next define the BN we used in Fig. 3 to predict the StallLabel variable.



$$P(Y,X,\theta_1,\theta_2) = \\ = [P(Y_i|X,\theta_1)P(\theta_1)] \cdot [P\{Y_i|X,\theta_2,P(Y_i|X,\theta_1) > \alpha\}P(\theta_2)]\,P(X)$$

Fig. 3: Bayesian Network using LR models.

Class	Training Accuracy	Validation Accuracy
NoStall	0.8681	0.8684
MildStall	0.7929	0.8048
SevereStall	0.9338	0.9368

TABLE III: BN class accuracy. Training vs. validation size ratio 4:1.

In the prediction step each data point is first classified by the 1st LR. If the 1st LR predicts the data to be in the SomeStall class then the data is further classed by the 2nd LR. Observe that the 2nd LR is dependent on the prediction of the 1st LR.

In the training phase, we first optimize the θ_1 parameters by standard gradient descent, hence obtaining the 1st LR and a prediction for the NoStall class. Then, we select from Y only the data that was predicted to have SomeStall by the 1st LR and we optimize for θ_2 to train the 2nd LR and gain the prediction for mild and severe stall classes. The pseudo-code of the training phase of our BN is shown in Fig. 4. We report in Table III the performance of the proposed BN model, that outperforms the SoA Random Forest approach (cfr. Tab. II).

We point out that we optimized the decision threshold of each LR, usually set to 0.5, so that the True Positive Rate (TPR) and the True Negative Rate (TNR) are equal. We do this in order to achieve similar final Class Accuracies (CA).

After deriving our BN model that makes a better trade-off between the class accuracies than the RF we would like to know if the data classes are linearly separable.

A. Is our data linearly separable?

In this section we seek to find if there is a more accurate, but also more complex mapping between the input data and the target variable StallLabel. We discovered through the experiment shown in Fig. 5 (top) that if we select *only the misclassified data by the 1st LR in Fig. 3* to train another (2nd) LR then we can reach a 97% CA. Thus, if we find a switching model as in Fig. 5 (bottom) that assigns the correct LR to each data point, then we will improve the overall CA of our BN.

In the next section we train the switching model by deriving a Neural Network (NN). Being able to extract the relevant features and make accurate predictions, NN became the state of the art technique to map and search highly complex relations [20].

Input: Data X, initial parameters $\theta_1 = 0$, $\theta_2 = 0$;
Output: Prediction for Y=StallLabel

Set $X := \log(X)$; Optimize the parameter θ_1 of the 1st LR by maximizing the log-likelihood via gradient descent method, i.e.:

$$\theta_1^t(i) = \theta_1^{t-1}(i) - \frac{\partial \ln(L(\theta_1^{t-1}))}{\partial \theta_1^{t-1}(i)}$$

Optimize the decision boundary α of the 1st LR such that:

$$\alpha = \text{argmin}_\alpha \, |TPR(LR_1) - TNR(LR_1)|$$

if $(P(Y_i|X,\theta_1) \geq \alpha)$ **then**
  $Y_i = \textit{NoStall}$
else
  $Y_i = \textit{SomeStall}$
end if

Select $X := X(Y_i = SomeStall)$ and $Y := Y(Y_i = SomeStall)$

Optimize θ_2 via gradient descent:

$$\theta_2^t(i) = \theta_2^{t-1}(i) - \frac{\partial \ln(L(\theta_2^{t-1}))}{\partial \theta_2^{t-1}(i)}$$

Optimize the decision boundary β of the 2nd LR:

$$\beta = \text{argmin}_\beta \, |TPR(LR_2) - TNR(LR_2)|$$

if $(P\{Y_i|X,\theta_2,P(Y_i|X,\theta_1) > \alpha\} \geq \beta)$ **then**
  $Y_i = \textit{SevereStall}$
else
  $Y_i = \textit{MildStall}$
end if

Fig. 4: Training algorithm for our Bayesian Network model.

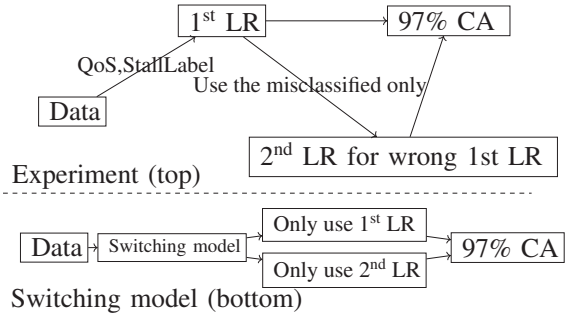


Fig. 5: Exploring how to improved the 1st LR of the BN: the experiment and the model that should yield the improvement.

B. Neural Network Search Using an Index Invariant Graph

Generally, when applying a NN it is best to derive its setting based on the underlying characteristics of the problem because this reduces the parameter set that needs to be hand-picked.

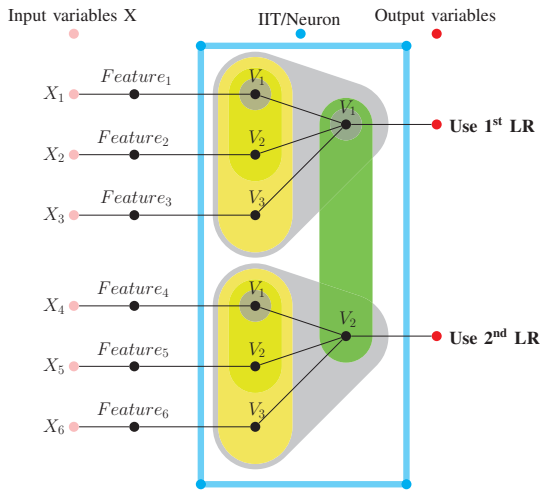


Fig. 6: The Index Invariant Tree, see [21], [22].

In order to derive the appropriate NN search method let us consider the example in Fig. 6. There we map between the input and output variables using a feature set and a single neuron. Notice that we do not use the classical NN model with sigmoid activation function. Instead, we resort to the tree structure of a neuron in order to reformulate it using an Index Invariant Tree (IIT). For the details of ITT, we refer to [21], [22]. Intuitively, the IIT states that each feature should map only to a single output and all pairs of features should not overlap. This enables to gain good prediction power while encoding large part of the data characteristics.

Following this analysis the training of the NN reduces to a stable set search on an appropriate graph whose vertices are features and each edge specifies if two features overlap. Here, a feature is any subset of the whole domain of the data, i.e., $Feature \subset \mathbb{R}^{\#variables}$. As a result, the training of the NN becomes a combination of simulated annealing [18] and graph clustering algorithms as in [22]. It is worth noting at this point that other clustering methods could be used based on available time and complexity constraints.

C. Using Hidden and Simulation Variables

It is worth examining another possibility to improve the prediction performance, namely the use of additional variables H. The additional variables are given in Tab. I as context and hidden variables. It is possible to train a separate model $H = F_H(X)$ for each of the simulation and hidden variables and then use their prediction in the final model $Y = F(X, H_{predicted})$. In this way the final model still uses only the input variables, but it makes an intermediate prediction on the additional variables which are then used along with the input to get the final target prediction. This process is shown in Fig. 7.

Bagged [19] Regression Tree and Bagged Random Forest are used for the additional variable prediction, depending on whether they are continuous or discrete.

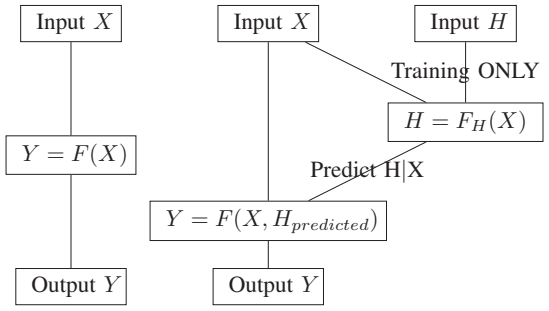


Fig. 7: Our approach using hidden variables. (Left) the standard prediction task. (Right) prediction procedure that includes intermediate hidden variables prediction.

D. Results from the IIT and the StallLabel variable

Tab. IV presents the results obtained using the BN predictor for five in different cases: when only QoS metrics are available, when additional context information is predicted or known to the network controller, and when hidden variables are obtained from prediction or perfectly known to the network controller. The table shows results only with four different validation splits, since the analysis of the standard deviation (STD) suggests that we cannot gain in accuracy.

We observe that the usage of the hidden variable procedure always increases accuracy and that if good model can be available for them the expected gain is high. Indeed, accuracies in the case where actual hidden variables are used (perfect prediction) plots a significant improvement compared to the one with QoS metric only . Also, the addition of context variables, which can be retrieved in practice from the network controller, improves prediction accuracy.

In term of the search for improvements, the IIT method did not find any areas where there are consistent misclassification between the two LRs in Fig. 5 for all classes (*NoStall,SomeStall,MildStall,SevereStall*). This is true regardless of whether the context parameters and the hidden variables are added as predicted or actual values. This finding suggests that the proposed BN based on LR is likely the best performing model on the considered dataset.

Case		Tr. NS	Tr. MS	Tr. SS	Val. NS	Val. MS	Val. SS
QoS metric only	Mean	0.8685	0.7946	0.9376	0.8665	0.7982	0.9267
	STD	0.0004	0.0014	0.0031	0.0018	0.0045	0.0122
Added predicted context	Mean	0.8691	0.7963	0.9409	0.8691	0.7996	0.9316
	STD	0.0004	0.0008	0.0031	0.0021	0.0036	0.0017
Added actual context	Mean	0.8689	0.7975	0.9396	0.8676	0.8023	0.9304
	STD	0.0001	0.0010	0.0027	0.0023	0.0033	0.0018
Added predicted hidden	Mean	0.8758	0.8012	0.9434	0.8735	0.7983	0.9366
	STD	0.0018	0.0033	0.0019	0.0034	0.0101	0.0079
Added actual hidden	Mean	0.9000	0.8399	0.9543	0.8990	0.8466	0.9530
	STD	0.0012	0.0023	0.0026	0.0020	0.0091	0.0113

TABLE IV: BN StallLabel training (Tr.) and validation (Val.) class accuracies for NoStall (NS), MildStall (MS) and SevereStall (SS) using a 4:1 training to validation size ratio and using hidden variables [38:42,48:51] (from Tab. I).

V. Video Bit-Rate and Quality Variation

We finally focus on the prediction of the two remaining QoE factors, namely average video bitrate (*AvgVideoBitRate*)

and its variation (*AvgVideoQualityVariation*). We remark that this turned out to be a much easier task, as those factors are linearly dependent on quantitative QoS metrics which are already observable. For this reason, a classic Regression Tree (RT) model was found to be suitable for both QoE factors.

AvgVideoBitRate can be easily predicted with high accuracy through the use of a bagged RT with minimum leaf size of 10. Tab. V shows the RT results. Similarly to AvgVideoBitRate, AvgVideoQualityVariation is efficiently predicted with a bagged RT with minimum leaf size of 10 as shown in Tab. VI.

We observe that the prediction of both the video bitrate and the quality variation is very accurate. Indeed, as results show, the mean error for both variables is only of a few kbps while the video bitrate can be up to 8 Mbps in the scenarios we considered. As for the case of StallLabel prediction discussed in the previous Section, these results also demonstrate that the accuracy can further improves when context information is know to the network controller and good prediction models are available for hidden variables. Our simple predictors for hidden variables help reducing a little bit the error but higher gains can be obtained with better models.

Case	Training	Validation
QoS metric only	58.13 (0.59)	68.78 (0.17)
Context Pred.	55.99 (0.52)	66.19 (0.10)
Context Actual	47.18 (0.75)	63.47 (0.23)
Hidden + Context Pred.	53.01 (0.75)	64.34 (0.16)
Hidden + Context. Actual	34.87 (0.60)	46.00 (0.12)

TABLE V: Mean and standard deviation (in parenthesis) for the prediction error of the average video quality (in Kbps). We used a 4:1 training to validation size ratio and hidden variables [38:42,44,45,51] (from Tab. I).

VI. Conclusion

In this paper we utilize machine learning techniques to demonstrate how QoS metrics can be exploited to accurately estimate and predict key QoE factors. We mostly focus on the StallLabel QoE factor as it is the hardest to predict. We improve on the RF performance [8] by building a more balanced model in term of class accuracies. We discovered simple but important patterns in the StallLabel variable and defined the BN model to envelop them. Using a custom NN based search method we showed that any other StallLabel model is unlikely to outperform our proposed Bayesian approach.

For all the crucial QoE factors we considered, we show that making intermediate predictions for hidden variables can boost the predictive performance of our approach, compared to the case where only observable variables are used. We also show that context information on network congestion and basic characteristics on video streams further improves predictions.

In the future, we plan to design new features, specific to video profiling (such as in [23]), that can be measured by QoS monitoring systems and improve QoE predictions.

Case	Training	Validation
QoS metric only	43.33 (0.23)	51.73 (0.69)
Context Pred.	37.51 (0.25)	46.41 (0.59)
Context Actual	28.88 (0.17)	41.24 (0.56)
Hidden + Context Pred.	36.78 (0.26)	46.09 (0.61)
Hidden + Context Actual	26.23 (0.13)	37.78 (0.38)

TABLE VI: Same as Tab. V but for the average video quality variation (in Kbps). Same training to validation ratio and same variables.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 20162021," June 2017.

[2] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," in *Proc. ACM SIGCOMM*, 2012.

[3] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[4] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *Proc. ACM IMC*, 2016.

[5] H. Nam, K.-H. Kim, and H. Schulzrinne, "Qoe matters more than qos: Why people stop watching cat videos," in *Proc. IEEE INFOCOM*, 2016.

[6] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proc. ACM MMSys*, 2011.

[7] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *ACM SIGCOMM*, 2015, pp. 325–338.

[8] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proc. ACM IMC*, 2016.

[9] F. Zhang, W. Lin, Z. Chen, and K. N. Ngan, "Additive log-logistic model for networked video quality assessment," *IEEE Trans. on Image Proc.*, vol. 22, no. 4, pp. 1536–1547, April 2013.

[10] ITU-T, "Parametric non-intrusive bitstream assessment of video media streaming quality - Higher resolution application area ," 2013.

[11] W. Song and D. W. Tjondronegoro, "Acceptability-based qoe models for mobile video," *IEEE Transactions on Multimedia*, vol. 16, no. 3, pp. 738–750, 2014.

[12] Z. Chen, N. Liao, X. Gu, F. Wu, and G. Shi, "Hybrid distortion ranking tuned bitstream-layer video quality assessment," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1029–1043, 2016.

[13] T. Hoßfeld, S. Biedermann, R. Schatz, A. Platzer, S. Egger, and M. Fiedler, "The memory effect and its implications on web qoe modeling," in *Proc. IEEE ITC*, 2011.

[14] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM CCR*, vol. 44, no. 4, 2015.

[15] P. T. A. Quang, K. Piamrat, K. D. Singh, and C. Viho, "Video streaming over ad hoc networks: A qoe-based optimal routing solution," *IEEE Tran. on Veh. Tech.*, vol. 66, no. 2, pp. 1533–1546, Feb 2017.

[16] Y.-T. Lin, E. M. R. Oliveira, S. B. Jemaa, and S. E. Elayoubi, "Machine learning for predicting qoe of video streaming in mobile networks," in *Proc. IEEE ICC*, 2017.

[17] C. Kreuzberger, D. Posch, and H. Hellwagner, "AMuSt Framework - Adaptive Multimedia Streaming Simulation Framework for ns-3 and ndnSIM," 2016.

[18] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[19] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[20] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. ACM ICML*, 2009.

[21] V. Vasilev, "Chromatic polynomial heuristics for connectivity prediction in wireless sensor networks," in *ICEST 2016*, Ohrid, Macedonia, 28-30 June 2016.

[22] V. G. Vasilev, *Algorithms and Heuristics for Data Mining in Sensor Networks*. LAP LAMBERT Academic Publishing, December 2016.

[23] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, and S. Valentin, "Traffic profiling for mobile video streaming," in *Proc. IEEE ICC*, 2017.