An Online Sampling Approach for Controlled Experimentation and QoE Modeling

Muhammad Jawad Khokhar, Thierry Spetebroot, Chadi Barakat

An Online Sampling Approach for Controlled Experimentation and QoE Modeling

Muhammad Jawad Khokhar, Thierry Spetebroot, Chadi Barakat
muhammad-jawad.khokhar@inria.fr, thierry.spetebroot@inria.fr, chadi.barakat@inria.fr
Université Côte d'Azur, Inria, France

***Abstract*—Predicting Quality of Experience (QoE) of end users from available network Quality of Service (QoS) measurements is of significant importance for today's network and content providers. This can be achieved by using application-specific QoE models that map the network QoS to the output QoE. QoS-QoE models can be built by training supervised Machine Learning (ML) algorithms with training data consisting of the mappings of the input network QoS to the output QoE. In most ML works on QoE modeling, the training data is usually gathered in the wild inside the core of the service or the content provider networks. However, such data is not easily accessible to the general research community. Consequently, the training data if not available before hand, needs to be built up by controlled experimentation. Here, the fundamental challenge is the sheer amount of time consumed in collecting the datasets needed to model the QoE. Considering this problem, we present here a framework of controlled experimentation based on active learning, that allows collecting rich datasets covering the experimental space intelligently. We perform a rigorous analysis of our approach and demonstrate the performance improvement over conventional pool based uncertainty sampling for the particular use case of YouTube video streaming.**

***Index Terms*—Active Learning, Quality of Experience, Controlled Experimentation**

I. Introduction

Machine learning (ML) is gathering a huge amount of interest within the networking community. A typical example of this interest can be found in the domain of Quality of Experience (QoE) modeling of the Internet applications where supervised ML classification algorithms are used. QoE modeling refers to building a model that maps the network or application level Quality of Service (QoS) measurements of a given application to the application specific Quality of Experience (QoE), e.g., Mean Opinion Scores (MOS), video abandonment rates. Such models can have diverse applications ranging from predicting the QoE from network measurements [1],[2], to optimizing the application performance itself [3].

Supervised ML classification techniques require the availability of some training data that is used to infer a model or a function linking the given input features to the output labels. Usually, ML works in the domain of QoE modeling use large training datasets that are mostly generated in the wild by a large population of real users. These datasets usually come from measurement probes within the network of the service/content providers [4],[5],[6],[7]. Such *internal* datasets are usually not made public to the research community, pushing most researchers to rely on building their own datasets. Building datasets on your own requires experiments to be carried out in a controlled environment where the input QoS features, e.g., bandwidth, delay, are varied artificially (using network emulators such as *tc* [1] and *mininet* [2]) and the target application is run under the enforced network conditions. The result is a training set whose individual samples are mappings of the enforced network QoS to the output QoE. These training sets are then used to train a supervised ML algorithm to build the QoS-QoE model.

For accurate QoE modeling with ML, we need to have large training datasets. The conventional approach of building such datasets is to experiment over a large set of unlabeled network QoS configurations *uniformly sampled* over the entire experimental space, i.e, the range within which the individual network QoS features are varied. The challenge here is in the large space to cover (power of the number of QoS features) and the non-negligible time required by each experiment to complete. For example, to obtain a dataset of 10^N samples, N being a positive integer number, in a scenario of N QoS features, with roughly 10 unique values per QoS feature if samples are placed on a grid, and if each experiment requires X minutes to complete, then the total time consumed in building such a dataset would be equal to $X.10^N$ minutes. If N equals 4 features and X equals two minutes, this is roughly 14 days! This experimentation cost is exacerbated by the fact that some QoS features span different orders of magnitude (as the bandwidth), and also by the fact that Internet applications and protocols are diverse and rapidly evolving, thus requiring the QoE models to be re-built on a regular basis.

In order to reduce this training cost, we observe that the space in which the experimentations are carried out can show a high degree of similarity in the output labels of QoE. Experimenting with this similarity provides little improvement in modeling accuracy in the case of uniform sampling of the experimental space. We aim to exploit this similarity to reduce the training cost while building the QoE models. In light of this observation, we advocate the use of active learning to reduce this training cost without compromising accuracy. Active learning is a semi-supervised ML approach where the learner is intelligent enough to select which samples it

This work is supported by the French National Research Agency under grant BottleNet no. ANR-15-CE25-0013 and Inria Project Lab BetterNet.

[1] http://lartc.org/
[2] http://mininet.org/

wants to label and learn from as part of an iterative process. The most popular technique of active learning is called the pool based *uncertainty sampling* in which at each iteration, the learner poses queries on a large set of unlabeled data often called the *pool*, to select the most rewarding sample for labeling in terms of the gain accuracy in the model under construction; the most rewarding instance is the one for which the model has maximum uncertainty [8]. In QoE modeling by controlled experimentation, the unlabeled data pool is the set of network QoS configurations over which the experiments can be performed. And labeling can be understood as the process of experimentation to get the QoE label, e.g. Good or Bad, for the corresponding network QoS.

We have applied pool based uncertainty sampling in a previous work [9] where we presented a first validation with a trace collected and labeled with uniform sampling, thus forming our ground truth. With the help of a video streaming QoE case, we validated the gain of uncertainty sampling and its capacity to reduce the training cost by an order of magnitude. This validation was done offline on a trace that we fully control. We observed that the performance of the ML model built is dependent on the size of the pool which is required to be predefined before starting the experiments. The size of the pool can be as large as possible but having pools of the order of millions can increase the cost of computations per iteration manifold because we need to compute at each iteration, the uncertainties of all samples in the pool according to the model under construction. Moreover, as uncerainty sampling picks samples with the maximum uncertainty, the learner in [9] can stick to some parts of the space showing high noise in the data, thus causing useless experiments until that part of the pool is fully explored. This phenomenon is referred to as *hasty generalization* in literature [10] where the learner tends to quickly converge to a final model based on an underlying *inaccurate* decision boundary.

In order to avoid these problems, in this paper, we depart from this previous work and re-frame the whole framework of active learning for controlled experimentation. We present a version that can work online without relying on having any predefined pool. Rather, we directly select a network configuration from a *region* of high dissimilarity in the given experimental space and we randomize the selection so that the aforementioned blockage problem is solved. We perform a rigorous analysis of our active learning approach and show that it can be used to build rich datasets intelligently. We demonstrate our approach with the case of YouTube QoS-QoE modeling with ML where the input controllable features are bandwidth, delay and packet loss, and the output QoE labels are binary depending on whether the video playout stalls or not. Apart from YouTube, our approach is general and can work in any scenario where the input features are controllable, real valued, continuous and bounded by a given range. In summary, the contributions of this paper are: 1) we devise a novel active learning approach to intelligently vary the network conditions and build a rich dataset of QoS-QoE mappings, and, 2) we minimize the computation cost of uncertainty and implement randomness in the selection to avoid being trapped in noisy parts of the space. The rest of the paper is organized as follows. In Section II, we discuss about active learning and explain our overall methodology. The evaluation of our active learning approach is done in Section III followed by our conclusion in Section IV.

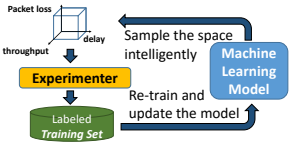


Fig. 1: Active Sampling

II. The Proposed Active Learning Approach

Our proposed approach can be summarized in Fig. 1, where instead of selecting samples from a pool of scenarios, the experimental space itself is directly sampled to obtain a point [3] from the *regions of high uncertainty* in the QoS feature space. This sampled point is then labeled by controlled experimentation and added to the training set to complete the iteration of the active learning process.

The regions of high uncertainty in the feature space are those where the ML model under construction has low confidence in its prediction (or classification). Some ML algorithms such as K-means and Decision Trees (DT) have this intrinsic capability of cutting the space into regions and assigning confidence levels to each region. Other ML algorithms can be complemented by an auxiliary solution to cut the space into such regions. For ease of presentation and without losing generality, we consider Decision Trees in this work. A DT learner splits the space using a set of internal nodes and edge *leafs* in a tree structure, that is learnt from the studied trace. Each node in the tree is assigned a feature threshold and the arcs generated from each node downwards indicate the value of the feature meeting the criterion at the node. The last node in this tree structure, called a *leaf*, represents a unique region in the feature space. Each leaf has a certain number of samples from each class and is labeled with the class having the maximum number of samples in it. Labeled leafs come with some uncertainty, which can be quantified by the measure of *Entropy*. So, in a given DT model with L leafs, each leaf i can have an entropy e_i given by

$$e_i = -\sum_{m=1}^{M} p_i^{(m)} \log p_i^{(m)}, \tag{1}$$

where $p_i^{(m)}$ is the classification probability of class m in leaf i, i.e., equal to the number of samples of class m divided by the total number of labeled samples in the leaf, with M being the total number of output classes.

At each iteration, we propose to select a leaf for experimentation from the set of leafs of the DT learner. As the model has same certainty about all points in a leaf, the

[3] A sampled point refers to a specific experimentation scenario.

experimentation scenario (e.g., a tuple of round-trip time, loss rate and bandwidth) is then picked randomly from the region covered by the selected leaf. The criterion for selecting the best leaf for experimentation could be based on the same criterion of pool based uncertainty sampling which is to select the *leaf* for which the entropy of the model is the highest. However, as already stated, if the regions of each of the QoE classes, e.g., Good or Bad, are not highly separable in space, i.e., if there is noise in the given regions, then experimenting with network scenarios in the regions of highest uncertainty may cause the learner to get stuck in these regions. To avoid this situation, we modify this criterion such that instead of choosing the region with the highest entropy, we propose to select the regions for labeling with a probability that follows the distribution of the entropies of the leafs in the given experimental space. As a consequence, regions in space with high uncertainty are *more likely* to be used for experimentation compared to regions with low uncertainty. We call our approach HYBRID where an ith leaf is selected for experimentation from a set of leafs with a probability given by $e_i / \sum_j e_j$, e_i is given in (1).

The overall summary of our methodology is given below:

1: Θ = Decision Tree ML Model
2: $\mathcal{L}$ = Leafs of a Decision Tree $\{l^{(i)}\}_{i=1}^{L}$
3: $\mathcal{T}$ = Training set of labeled instances $\{\langle \mathbf{x}, y \rangle^{(i)}\}_{i=1}^{T}$
4: Φ = Utility measure based on Entropy
5: **for** each experiment **do**
6: select $l^* \in \mathcal{L}$, as per Φ
7: randomly select $\mathbf{x}^* \in l^*$
8: experiment using $\mathbf{x}^*$ to obtain label y^*
9: add $\langle \mathbf{x}^*, y^* \rangle$ to $\mathcal{T}$
10: $\Theta = \textbf{train}(\mathcal{T})$
11: **end for**

As with all ML algorithms, there are certain parameters for DTs that need to be configured beforehand. For active learning, the most important one is the *minimum number of samples per leaf*. This parameter specifies the minimum count of samples that are allowed in each region of the feature space modeled as a leaf. Note here that if this value is set too small, then the DT leafs will always be homogeneous in terms of the labels they contain, thus preventing any reliable uncertainty measure to be associated with them.

A. Stopping Criterion

With our proposed HYBRID approach, we can stop the experimentations when the model stops improving while new samples are getting labeled. The convergence of the model under contruction can be gauged by several measures including model accuracy, uncertainty and model confidence [11]. For the accuracy measure, a separate validation set that is assumed to represent the ground truth has to be available over which the model can be tested at every iteration of active learning. However, when the training set is built on the fly as in our case, we do not have any information on the ground truth beforehand. So, in this case, we cannot use an independent accuracy measure to observe change in model performance over time. Instead, we will demonstrate model convergence by observing the change in the uncertainty (Equation 1) and the confidence measures (Section II-B) over the course of the iterations. As we will show in Section III-D, the uncertainty of the model decreases while the confidence increases with increasing number of experiments, eventually attaining a plateau indicating model convergence after which no major subsequent change in the model occurs even with more experiments. Thus, at this stage, further experiments can be stopped.

B. Gauging Model Quality using Model Confidence

We explain how the confidence measure, which reflects the *quality* of the model under construction, can be calculated for Decision Trees. For leaf i, let's define the confidence of the model in the label it assigns to the leaf as $c_i = \max \langle p_i^{(1)}, \ldots p_i^{(M)} \rangle$, where $p_i^{(m)}$ is the classification probability per class m in the given leaf i. With this definition, we can define a set $\mathcal{C} = \{c_i\}_{i=1}^{L}$ that denotes the confidence measures for each leaf i in the entire feature space, L being the number of leafs. To have a unified measure for the entire space, we can simply use the average of $\mathcal{C}$ as a single measure representing the global confidence of the model over the entire feature space. But simple averaging means that we give equal weight to all leafs. A better approach is to have different weights associated to different leafs based on the volume of the regions they occupy in the feature space, thus bigger leafs can be assigned bigger weighting factors and vice versa. The weighting factor w_i can be computed as

$$w_i = \frac{1}{X_1 X_2 ... X_N} \int \cdots \int_{x_n^{LOW_i}}^{x_n^{HIGH_i}} dx_1 \ldots dx_N, \qquad (2)$$

where $x_n^{LOW_i}$ and $x_n^{HIGH_i}$ are the threshold values for feature x_n of leaf i. These thresholds define the limits of the region represented by each leaf in the feature space. We normalize with respect to the total volume of the feature space $X_1 X_2 ... X_N$ to get weights between 0 and 1.

With w_i computed, we can then define our unified *Weighted Confidence Measure* as $\sum_{i=1}^{L} c_i w_i$. By defining it this way, the confidence measure models the confidence, or certainty, of the constructed model in classifying a random sample picked with a uniform probability in the feature space into a QoE label. This measure should be the highest possible.

III. Evaluation

A. Implementation for YouTube QoE

We implement our proposed active sampling methodology shown in Fig. 1 in a controlled experimentation framework for modeling YouTube video QoE from network QoS features. The framework consists of a *client* node, that performs the experiments, and a *controller* node, which implements active learning using Python Scikit. At each experiment, the client node 1) obtains from the controller a QoS feature tuple which it enforces on its network interface using linux traffic control `tc`, 2) runs a sample YouTube 720p video[4] and obtains its corresponding QoE, 3) sends back the labeled tuple of the

[4]The YouTube Video ID: `oFkulzWMotY`

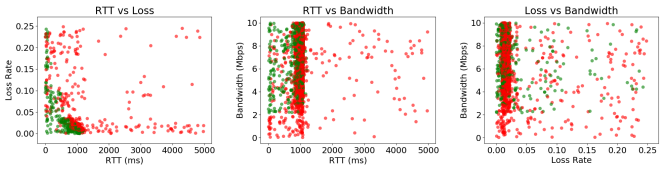
enforced QoS and the output QoE to the controller, which then updates the DT model locally. For the QoE, which is usually a subjective measure, we suppose it gets a binary value (*Bad/Good*) for YouTube video playout, i.e., *Bad* if it suffers from stalling or has a join time of more than 30 seconds, and *Good* if the video starts within 30 seconds and plays out smoothly without any stalls. Other finer definitions of QoE will be developed in a future research, the current one being solely focused on the validation of the online active learning approach. In the subsequent analysis, the *green* color refers to the *Good* class and the *red* color refers to the *Bad* class.

The features used to configure the network QoS with `tc` are Round-Trip Time (RTT), Download Loss Rate and Download Bandwidth. Given the asymmetric nature of video streaming traffic, we believe these features are most relevant from a QoE perspective. The space within which the features are varied is 0-5000 ms for RTT, 0-25% for Loss Rate and 0-10 Mbps for Bandwidth. Although this experimental space might seem impractically large, we will see how active sampling can enforce experimentation in practical regions.

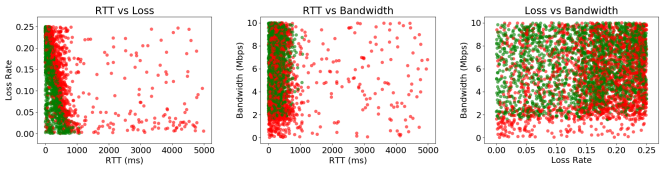
B. Datasets

To evaluate our methodology, we collect two datasets, $\mathcal{D}_{MAXENTROPY}$ and $\mathcal{D}_{HYBRID}$ based on two different sampling approaches, MAXENTROPY and HYBRID. In MAXENTROPY, the region selected for labeling is the one which has maximum entropy according to the ML model under construction whereas in HYBRID, the criterion for selecting a region is based on an entropy-based probabilistic measure as discussed in Section II. Each dataset is comprised of 4000 YouTube video playouts in the controlled network environment. We compare the DT models built by these two datasets and prove how our proposed methodology HYBRID can build better QoE models compared to MAXENTROPY. For the sake of comparison between the online and offline methods, we borrow the DT models built offline in our previous work [9] over a pool of 10K scenarios uniformly chosen in the experimental space. We then illustrate how the model built with $\mathcal{D}_{HYBRID}$ converges over the course of the iterations showcasing that further experiments can be stopped once stability is achieved in the model under construction. Finally, we observe the performance of some common ML algorithms trained with $\mathcal{D}_{HYBRID}$ and demonstrate that DTs perform well enough compared to other learners in our case.

The visual representation of the collected datasets is shown in Figs. 2a and 2b. In these figures, sampled network scenarios are projected over different pairs of QoS axes. We can clearly see a significant difference between the two approaches in the regional distribution of the sampled scenarios for both the Good and the Bad classes. For MAXENTROPY, the active learner is stuck in a small region of the experimental space which results in other useful regions being missed out. The HYBRID approach and thanks to its intrinsic random behavior, mitigates this problem and results in experiments being carried out in a larger region with a better capture of the decision boundary. As a result of this difference in the datasets, the DT

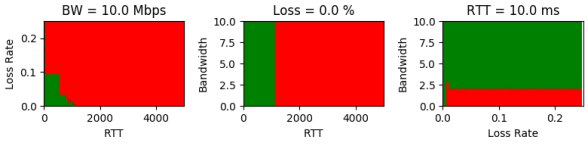


(a) MAXENTROPY

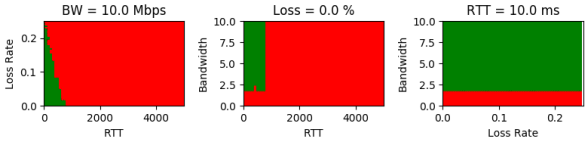


(b) HYBRID

Fig. 2: Visual representation of the collected datasets



(a) MAXENTROPY



(b) HYBRID

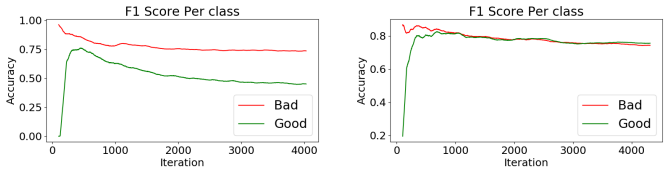
Fig. 3: Visual depiction of the DT ML Model

models trained with these datasets are also different. This is highlighted in Fig. 3a where we show the visual representation of what is predicted by the DT model over the same pairs of QoS axes. Clearly, the model built with the HYBRID approach captures better the distribution of QoE labels over the space. One still needs to validate the accuracy and confidence of the obtained models overall.

C. Accuracy and confidence

As we are in an online mode, we don't want to condition our validation by the presence of a separate dataset forming the ground truth. We want the validation to be carried out over the dataset itself produced on the fly by active learning. For this, we resort to a well known technique called *k-fold cross validation* to gauge the performance of the models. In k-fold cross validation, the target dataset is split into training and validation sets k times randomly. The model is then trained with the training set and tested with the validation set k times to get k accuracy scores. The final accuracy score is then the average of these k scores. We plot the F1-Score[5] based

[5]The F1-score is a measure to gauge the accuracy of the ML model by taking into account both the precision and recall. It is given by $2pr/(p+r)$, where p and r are the precision and recall of the ML model respectively. It takes its value between 0 and 1 with larger values corresponding to better classification accuracy of the model.



(a) MAXENTROPY (b) HYBRID

Fig. 4: Comparison of the F1-Scores per class b/w HYBRID and MAXENTROPY

on cross validation ($k = 3$ with a data split ratio of 80:20 for training and validation) that is computed at each iteration of our experimentation in Fig. 4, where we can see that HYBRID achieves a better accuracy for both classes compared to MAXENTROPY[6]. However, an important observation here is that as we increase the number of experiments, the cross validation F1-Score begins to decrease. For MAXENTROPY, this drop is significantly more important. The reason for it is that as we keep on experimenting, more and more scenarios are picked from the uncertain regions nearby the boundary between classes, thus making the resulting dataset more and more noisy and difficult to predict.

We further calculate the *Weighted Confidence Measure* – discussed in Section II-A– to better gauge the quality of the models. Fig. 5 shows this comparison for DT models built with HYBRID and MAXENTROPY. We also add to this figure the result from our previous work where DT models were calculated offline over a pool of 10K pre-labeled scenarios [9]. For this latter result, the performance of the learner is limited by the size of the pool used, which explains why it shows less confidence in its prediction than our two proposed online approaches. The figure shows that HYRBID can build QoE models having a better confidence than the other approaches, which added to the previous cross-validation result, confirms that the QoE models built with HYBRID are of better quality compared to those built with MAXENTROPY. Table I further highlights this result by showing the same measure of confidence but w.r.t each class after 3000 experiments were carried out. Not only HYBRID has better confidence on average, but it is also more confident in its prediction for the both classes. Interestingly, the confidence of the model for the *Bad* class is very high for all the approaches. As mentioned above, this is because of the unbalance of the space between the two classes (see Fig. 2). In fact, in our case, the performance of the model w.r.t to the minor class (*Good*) actually defines how good the model is. And indeed, HYBRID has the best confidence for the *Good* class as well among all the other sampling approaches.

D. Model convergence

To study the convergence of the model with the HYBRID approach, we start by showing the variation in the entropies of the leafs of the DT model in Fig. 6. We can observe that

[6]The results shown in Figures 4 to 7 are a based on moving average with a window size of 100 iterations to smooth out the curves for better presentation.

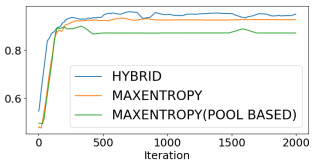


Fig. 5: The Weighted Confidence Measure

Strategy	*Bad*	*Good*
HYBRID	99%	93%
MAXENTROPY	99%	87%
MAXENTROPY(POOL)	100%	78%
UNIFORM	99%	71%

TABLE I: Model confidence per class after 3000 iterations

the minimum and maximum entropies of the DT leafs remain consistently at the respective values of 0 and 0.7. An entropy value of zero means that throughout the experiments, there remain regions in the experimental space which have clear uniformity, i.e., all samples in those regions belong only to one class thus the model would have 100% confidence in such regions. A second observation from Fig. 6 is that the maximum entropy goes up to 0.7 and remains at this value even if we add further experiments. This means that in this state, the ML model would have regions where it is still uncertain; but these regions are small in volume and cannot improve further. Finally, the average entropy shown in Fig. 6 is a weighted sum of the entropies of all leafs with each leaf assigned a weight according to the leaf's geometric size. We can see that the average entropy progressively goes down to a stable low value which is an indication of the model convergence in its leaf entropy distribution.

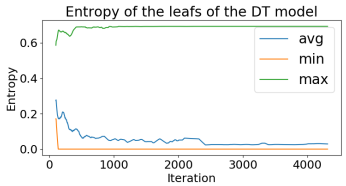


Fig. 6: Variation of Entropy

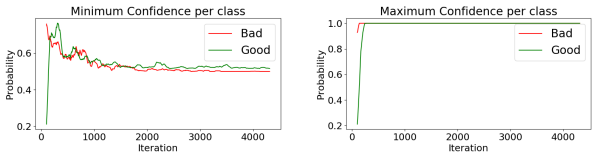


Fig. 7: Classification Probabilites (Confidence) of the DT Leafs per class

We now study convergence from another perspective. Fig. 7 shows the model's minimum and maximum classification probabilities per class over all leafs labelled with that class. The results here are complementary to the ones in Fig. 6 as maximum (resp. minimum) entropy corresponds to minimum (resp. maximum) confidence. As said above, the maximum

classification probability is 1.0 for both classes, which confirms the presence of regions in space where the model is 100% confident. As more and more leafs are added, and as leafs get smaller and smaller, the minimum probabilities converge to a value around 0.5, which in our binary case is the maximum uncertain scenario. This minimum is driven by highly uncertain small leafs.

The convergence of all these metrics is an indication of the model stability. Experiments can then be safely stopped as the model stops evolving. The stopping criterion can be the point where all these metrics converge, but one can anticipate and stop the experiments when the overall confidence measure stops increasing, as shown in Fig. 5. Experiments beyond this point have almost no impact on the model performance.

E. Discussion

An important question is why the F1-Scores obtained in Fig. 4 are only up to around 75% whereas similar QoE models for YouTube [6][12] obtain larger values. The reason is in the difference in the features used in our work compared to the ones used in the literature. The model we present here uses the *enforced* QoS on `tc` (Outband measurements) as input features whereas the models presented in the literature mostly use QoS features directly obtained from the application traffic itself (Inband measurements). Such features have better correlation with the output QoE, thus normally should result in better models. Their drawback is that they require access to the application traffic itself to predict the QoE, whereas with our QoS features, and this is the main motivation behind our project ACQUA [2], we can talk about QoE prediction without the need to run the application itself or have its traffic. To confirm this difference, we collect in parallel such *application traffic* features from the raw YouTube video traffic traces while building $\mathcal{D}_{HYBRID}$. The features we collect are based on the network measurements of Download throughput, Download packet inter-arrival time, Upload packet inter-arrival time and Download packet size. For each of these measurements, we calculate 6 statistical measures during each video streaming session. These statistical measures are the average, maximum, standard deviation, 20th, 50th and the 90th percentiles. Overall, 6 statistical measures for the 4 network measurements amount to a total of 24 features given as input to a new ML QoS-QoE model. The improvement in the accuracy by modeling with these features can be confirmed by comparing results in Table II where a gain of 15-20% can be noticed. We consider this difference as the cost of QoE prediction and we will be looking into ways to reduce it further. Note here that in this table different learners are used to confirm the insensitivity of our results to the choice of Decision Trees as a machine learning technique.

Classifier	Class	Outband QoS			Inband QoS		
		Prec	Recall	F1	Prec	Recall	F1
Nearest Neighbors	Good	57%	55%	56%	89%	90%	90%
	Bad	58%	60%	59%	90%	89%	90%
Decision Tree	Good	74%	73%	73%	96%	93%	94%
	Bad	74%	75%	74%	93%	96%	95%
Random Forest	Good	76%	78%	77%	96%	96%	96%
	Bad	78%	76%	77%	96%	96%	96%
AdaBoost	Good	79%	72%	76%	94%	95%	95%
	Bad	75%	81%	78%	96%	94%	95%
Naive Bayes	Good	75%	22%	34%	94%	66%	78%
	Bad	57%	93%	71%	74%	96%	83%

TABLE II: Performance of popular ML algorithms with *Inband* and *Outband* QoS features

IV. Conclusion and Future Work

In this paper, we demonstrated an active sampling approach based on decision trees to build rich and efficient QoS-QoE datasets in a controlled experimentation environment. We proved how our approach can ensure better performance and lower computational cost than the conventional approach based on maximum entropy and fixed size pool, and how, thanks to the randomness it introduces, can better cover the experimental space without being blocked in some of its noisy parts. In future, we will implement our approach on more applications such as Skype and Web Browsing to build QoE models that would be integrated into our mobile application platform ACQUA [2] for predicting end user's application specific QoE from network level measurements.

References

[1] T. Spetebroot, S. Afra, N. Aguilera, D. Saucez, and C. Barakat, "From network-level measurements to expected quality of experience: The skype use case," in *Measurements Networking (M & N), 2015 IEEE International Workshop on*, Oct 2015.

[2] "ACQUA: Application for prediCting Quality of User experience at Internet Access," 2017, http://project.inria.fr/acqua/.

[3] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of ACM SIGCOMM*, New York, 2015, pp. 325–338.

[4] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. New York, NY, USA: ACM, 2013, pp. 339–350.

[5] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Understanding the impact of network dynamics on mobile video user engagement," in *The 2014 ACM Int'l Conference on Measurement and Modeling of Computer Systems*. NY, USA: ACM, 2014, pp. 367–379.

[6] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*, ser. IMC '16. New York, NY, USA: ACM, 2016, pp. 513–526.

[7] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "CFA: A practical prediction system for video qoe optimization," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, 2016.

[8] B. Settles, "Active learning literature survey," University of Wisconsin Madison, Computer Sciences Technical Report 1648, 2010.

[9] M. J. Khokhar, N. A. Saber, T. Spetebroot, and C. Barakat, "On active sampling of controlled experiments for qoe modeling," in *Proceedings of the Workshop on QoE-based Analysis and Management of Data Communication Networks*, ser. Internet QoE '17. New York, NY, USA: ACM, 2017, pp. 31–36.

[10] B. C. Wallace, K. Small, C. E. Brodley, and T. A. Trikalinos, "Active learning for biomedical citation screening," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2010, pp. 173–182.

[11] J. Zhu, H. Wang, E. Hovy, and M. Ma, "Confidence-based stopping criteria for active learning for data annotation," *ACM Trans. Speech Lang. Process.*, vol. 6, no. 3, pp. 3:1–3:24, Apr. 2010.

[12] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, *Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience*. Springer Berlin Heidelberg, 2013, pp. 264–301.