An Intelligent Sampling Framework for Controlled Experimentation and QoE Modeling

Muhammad Jawad Khokhar, Nawfal Abbassi Saber, Thierry Spetebroot, Chadi Barakat

Université Côte d'Azur, Inria, France

Abstract

For internet applications, measuring, modeling and predicting the quality experienced by end users as a function of network conditions is challenging. A common approach for building application specific Quality of Experience (QoE) models is to rely on controlled experimentation. For accurate QoE modeling, this approach can result in a large number of experiments to carry out because of the multiplicity of the network features, their large span (e.g., bandwidth, delay) and the time needed to setup the experiments themselves. However, most often, the space of network features in which experimentations are carried out shows a high degree of similarity in the training labels of QoE. This similarity, difficult to predict beforehand, amplifies the training cost with little or no improvement in QoE modeling accuracy. So, in this paper, we aim to exploit this similarity, and propose a methodology based on active learning, to sample the experimental space intelligently, so that the training cost of experimentation is reduced. We validate our approach for the case of YouTube video streaming QoE modeling from out-of-band network performance measurements, and perform a rigorous analysis of our approach to quantify the gain of active sampling over uniform sampling.

Keywords: Quality of Experience, Active Learning, Controlled

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

Email addresses: muhammad-jawad.khokhar@inria.fr (Muhammad Jawad Khokhar), nawfal.abbassi.saber@gmail.com (Nawfal Abbassi Saber), thierryspetebroot@gmail.com (Thierry Spetebroot), chadi.barakat@inria.fr (Chadi Barakat)

1. 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, as in predicting the QoE of a certain application for a given network QoS (e.g., our previous work ACQUA [1, 2]), or optimizing network algorithms and protocols while accounting for end users QoE [3], [4].

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 [5],[6],[7],[8]. 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 their own requires experiments to be carried out in a controlled environment where the input QoS features, e.g., bandwidth, delay etc., are varied artificially (using network emulators such as *tc* [1] or *mininet* [2]) and the target application is run under the enforced network conditions. The result is a training set whose individual entries are mappings of the enforced QoS to the output QoE. These training sets are then used to train the supervised ML algorithms to build the QoS-QoE models.

It is to be noted that the space of experimentation can be huge as every QoS feature corresponds to one dimension with a potential wide range (e.g., from a few bits per second to gigabits per second for the bandwidth feature). Also, and more important, the complexity of the experimental space

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

increases by the power of the number of QoS features. As each experiment requires a non-negligible time to be executed (e.g., order of minutes for video streaming), the overall time required to build the models can then be huge. For example, to obtain a dataset of 10^N samples, in a scenario of N QoS features, with roughly 10 unique values per QoS feature placed on a grid, and if each experiment requires X minutes to complete, then the total time required to build such a dataset would equal to $X.10^N$ minutes. For N equal to 4 features and X equal to two minutes, the required experimentation time is 14 days! This is a significant hindrance as today's Internet applications are rapidly evolving and their implementations quickly changing, which urges for the models to be re-built on a regular basis. So, reducing the training cost becomes an absolute necessity. This reduction is even more needed if one considers the constant increase in the number of applications and services and the large diversity of content they provide.

In order to reduce this training cost, we observe that the space in which the experiments are carried out can show a high degree of similarity in the output labels of QoE. By similarity, we mean how many samples in a small region of the feature space have same output labels. 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, in this paper, we propose a sampling methdology for QoE modeling that is based on *active learning* to reduce this training cost without compromising accuracy. Our methodology *intelligently samples* the network QoS space by only experimenting with the most relevant configurations without impacting modeling accuracy, hence reducing cost of experimentation and considerably improving the time required for building the QoE models.

Active learning is a semi-supervised ML approach where the learner is intelligent enough to select which samples it wants to label and learn from this labeling as part of an iterative process. It is mostly used in scenarios where there is a large *pool* of unlabeled data and the cost of labeling is high [9]. Here, at each iteration, the learner poses queries on the large pool of unlabeled data and selects the most relevant sample for labeling in terms of the gain in accuracy of the model under construction; the most rewarding instance is considered to be the one for which the model has maximum uncertainty. This approach is known as *pool based uncertainty sampling* in literature [9].

In this paper, we apply the aforementioned approach for QoE modeling by controlled experimentation where we present a first validation for a case

of YouTube video streaming with the help of a dataset collected and labeled with uniform sampling, thus forming our ground truth. In this case, we validate the gain of *pool based uncertainty sampling* and show its capacity to reduce the training cost by an order of magnitude. We then observe 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 uncertainty computation. Moreover, as active sampling picks samples with the maximum uncertainty, the learner 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 this problem, we then reframe 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 consider YouTube in our paper as it is the most widely used video streaming service and it provides an API that can be used to capture the video QoE features such as join time, stallings etc, in a web browser. 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. Overall, the goal of the paper is to present an intelligent sampling methodology with the aim of reducing the training cost of QoS-QoE modeling in a controlled experimentation scenario. In summary, the contributions of this paper are:

1. We present an application of pool based uncertainty sampling for QoS-QoE modeling by controlled experimentation for a case of YouTube video streaming showcasing significant gain over uniform sampling.

2. We highlight the blocking issue with pool-based sampling and devise our online active learning approach that minimizes the computation cost of uncertainty and implements randomness in the selection to avoid being trapped in noisy parts of the space.

3. Finally, we analyze our approaches with both real and synthetic feature spaces to conclude that the gain of active active sampling over uniform sampling is dependent on the complexity of the experimental space considered for a given experimentation scenario.

The rest of the paper is organized as follows. In the next section, we give a brief overview of active learning and explain how it is relevant for QoE modeling using controlled experimentation. We then discuss the methodology, implementation and analysis for pool based sampling in Section3 and our online sampling methodology in Section 4. In Section 5, we discuss the application of active sampling in real production networks and present a validation of our sampling approach on an open dataset of network measurements. The related work is discussed in Section 6 and the paper is concluded in Section 7. The work in this paper is a consolidation of our previous works in [11] and [12] with additional analysis on synthetic datasets.

2. Active Learning for QoE Modeling

Conventional supervised machine learning builds on a training dataset, which consists of pairs of input features and output labels, to infer a function or a model that is then used to predict the label of new input features. Traditionally, the learning algorithms rely on whatever training data is available to the learner for building the model. Yet, there are scenarios like ours where there is an abundant availability of unlabeled data, and the effort involved in labeling these data can be complex, time consuming or simply requires too many resources. For our case for example, a label of a network QoS instance is the QoE associated to this instance, which requires an experimentation to be known. In such scenarios, building a training data by labeling each unlabeled instance can become a tedious task that can seriously consume significant amount of resources just to build the fully labeled training data. At the same time, and more often, the training data built can contain redundancy in the sense that most of the labels come out to be similar for big parts of the unlabeled dataset. So, if the learner can become "intelligent" enough in choosing which unlabeled instances it wants to label and learn from, the task of building the training dataset can be greatly improved. This improvement should come by reducing the size of the training dataset without impacting the accuracy of the learner. In fact, an *Active Learning* system updates itself as part of a continuing interactive learning process whereby it

develops a line of inquiry on the unlabeled data and draws conclusions on the data much more efficiently [9].

The literature on active learning suggests three fundamental modes of performing the inquiry on the available data, which are Query synthesis, Stream-based selective sampling and Pool-based sampling. In Query syntesis, the queries to label new features are synthesized ex novo [13], whereas in stream based and pool based sampling, the queries are made based on an *informativeness measure* (we discuss the informativeness measure in the upcoming Section 2.1). The difference between the latter two is in the way the unlabeled data is presented to the learner, i.e., as a *stream* of data (where the learner either selects or rejects the instance for labeling) or a *pool* of data (where the *most rewarding* sample is selected from a *pool* of unlabeled data). For many real world problems including ours, large amount of unlabeled data can be collected quickly and easily, which motivates the use of *pool* based sampling, making it the most common approach for Active learning scenarios [9]. The other two modes (query synthesis and stream based) are omitted because of their incompatibility with our case study. We present in the next section our framework that relies on pool based sampling, where a *pool* is a set of network QoS features to experiment with (uniformly sampled in space), and where the objective is to find the most rewarding QoS instances in terms of the gain in the accuracy of the QoE model under construction.

2.1. The Informativeness Measure

The informativeness measure used for active learning is devised by several strategies in the literature as the ones based on uncertainty, query by committee, and error/variance reduction. Among them, the most popular strategy as per literature is *uncertainty sampling*, which is fast, easy to implement and usable with any probabilistic model. Due to its relevance to implementation and its interesting features, we only discuss about uncertainty sampling in this paper and leave the study of the other strategies for future research; detail on rest of the strategies can be found in [9].

To describe uncertainty sampling, we first notice that for any given QoS instance whose label has to be predicted by any machine learning model, an *uncertainty* measure is associated with this prediction. This refers to the certainty or confidence of the model to predict (or classify) the QoS instance belonging to a certain label (or class). The higher the uncertainty measure, the less confident the learner is in its prediction. This uncertainty measure is the *informativeness measure* that is used in *uncertainty sampling*.

Generally, uncertainty of a machine learning model is higher for instances near the decision boundaries compared to the instances away from them. These unlabeled instances near the current decision boundaries are usually hardly classified by the model, and therefore if labeled and used for training, would have greater chance of making the model converge towards a better learning accuracy as compared to other instances which are far from the decision boundaries. So, if such instances of high uncertainty are selected for labeling and training, the model would alter and assume a final shape much more quickly.

The literature on active learning suggests three main strategies of uncertainty sampling for picking the most uncertain instances in a given pool. Each strategy has a different utility measure, but all of them are based on the model's prediction probabilities. Let x denote the set of instances in the pool still candidates for being labeled. Based on this notation, a brief description of the various uncertainty sampling strategies is given below:

1. **Least Confident**. This is a basic strategy to query the instance for which the model is least *confident*, i.e., $x^*_{LC} = \arg\min_x P(\hat{y})$, where $\hat{y} = \arg\max_y P(y)$ is the most probable label for instance x. This means picking an unlabeled instance from the pool for which the best model's prediction probability is the lowest compared to other instances in the pool.

2. **Minimal Margin**. In this method, the instance selected is the one for which the difference (*margin*) in the probabilities of its first and second most likely predicted labels is minimal. Here $x^*_{MARGIN} = \arg\min_x [P(\hat{y_1}) - P(\hat{y_2})]$, where $\hat{y_1}$ and $\hat{y_2}$ are the first and second most likely predicted labels of an instance x. The lower this margin is, the more the model is ambiguous in its prediction.

3. **Maximum Entropy** is the method that relies on calculating the entropy of the given unlabeled instance, where, $x^*_{ENTROPY} = \arg\max_x -\sum_y P(y) \log P(y)$, with $P(y)$ being the probability of the instance being labeled y by the given model. The higher the value of the entropy is, the more the model is uncertain.

So, the best instance for selection becomes the one that has the maximum uncertainty which can be quantified using one of the above mentioned strategies.

3. Pool based sampling for QoE Modeling

Given a large pool of unlabeled network QoS instances, we first select an instance from the pool which has the maximum uncertainty (as previously discussed) and then experiment with this QoS instance to find its QoE label. We then update the QoE model with the obtained label, and recalculate the uncertainty of the remaining instances in the pool using the newly learned QoE model. Then again we select the most rewarding instance, and so on. This defines our iterative methodology for sampling the experimental space.

3.1. Methodology

Our methodology begins by defining the pool, $\mathcal{P}$ and initializing the training set, $\mathcal{T}$ with few labeled instances. A first QoE vs. QoS model is built using a machine learning algorithm. Decision trees are typical models, but other models are also possible as Bayesian or Neural Networks. The idea is to refine this model in an iterative way using the labels of most rewarding instances. At each iteration, we compute the uncertainty of all instances in the pool w.r.t the given QoE model and select the unlabeled instance with the highest uncertainty (based on the utility measures). The selected instance is removed from the pool, instantiated in the experimental platform, labeled with the corresponding QoE, then added to the training set. The updated training set is then used to train the model, Θ in the next iteration. The whole process is repeated until adequate number of samples are collected, denoted by the experimental budget N available to the experimenter. Note that experiments can be stopped before this budget limit if the model being built converges and stops improving further. In active learning literature, the criterion to stop further experiments is refered to as the *stopping criterion*, we will discuss about it later in Section 4.1. For evaluating our approach, we set the budget equal to the pool size and experiment until the pool is exhausted. To validate our approach, we also define a *Validation* set, $\mathcal{V}$, over which we validate the model Θ. The overall algorithmic summary of our methodology is given below:

1: $\mathcal{P}$ = Pool of unlabeled instances $\{x^{(p)}\}_{p=1}^{P}$
2: $\mathcal{T}$ = Training set of labeled instances $\{\langle x, y\rangle^{(t)}\}_{t=1}^{T}$
3: Θ = QoE Model e.g., a Decision Tree
4: Φ = Utility measure of Uncertainty e.g., Max Entropy
5: N = Experimental Budget
6: Initialize $\mathcal{T}$

for $i = 1, 2, ... N$ **do**
    $\Theta = \textbf{train}(\mathcal{T})$
    select $x^* \in \mathcal{P}$, as per Φ
    experiment using x^* to obtain label y^*
    add $\langle x^*, y^* \rangle$ to $\mathcal{T}$
    remove x^* from $\mathcal{P}$
end for

3.2. Implementation

To implement machine learning we use the Python SciKit-Learn library [14]. The choice of the learner in the context of uncertainty sampling is dependent on the intrinsic ability of the learner to produce probability estimates for its predictions, which ML algorithms such as Decision Trees and Gaussian Naive Bayes can provide inherently. However, other popular machine learning algorithms such as Support Vector Machines (SVM) do not directly provide the probability estimates of their predictions but make it possible to calculate them using an expensive k-fold cross-validation [15]. With Decision Trees, the probability estimates can be directly obtained from the distribution of the samples in the leafs of a Decision Tree making uncertainty calculation convenient. They also provide an intuitive understanding of the relationship between the QoS features and the corresponding QoE which learners such Neural Networks do not provide. So, due to the aforementioned reasons we use them as our choice of ML algorithm. Having said that, our active sampling approach is general and can be used with other ML learners including Neural Networks and SVM as well.

Note here that if the minimum number of samples per leaf of the Decision Tree is set to 1, leafs will be homogeneous in terms of the labeled instances per leaf, and the QoE model will remain *certain*, thus resulting in zero or one probability values which would not allow any uncertainty measure to be extracted from the model. So, the minimum samples per leaf in the Decision Tree should be set to a value larger than 1; we set it to 10 for binary classification and to 25 for multiclass classification (a QoE on five labels and a leaf size of 5 times the number of labels), to ensure that the QoE model allows to produce probablities between 0 and 1. We discuss later our validation of the approach using the YouTube use case, and the QoE labels used for classification (Section 3.5).

3.3. Definition of Accuracy

The accuracy of the QoE vs. QoS model (or classifier) is calculated over a *Validation* set and is defined as the ratio of correct classifications to the total number of classifications. It is to be noted that this global accuracy is relevant for the case of binary classification. But for multiclass classification, we also use a measure of the absolute prediction error, which we call the *Mean Deviation*, given by $E[|\hat{y} - y|]$, where y is the true label of an instance and $\hat{y}$ is the classifier's prediction. It is to be noted here that the *Mean Deviation* is only calculated over the *mis-classified* instances, so it caters for not just only the mis-classifications but also for the range of error.

3.4. Building the Pool and the Validation set

To assess the benefit of active learning for QoE experimentation and modeling, our study is based on a YouTube video streaming dataset. This case study is meant to validate the interest from active sampling, we leave the thorough analysis of YouTube QoE to future research. Our overall data collection was based on a sample 2 min YouTube 720p video (video ID: Ue4PCI0NamI) with a time out of 5 mins for each experiment. It took our setup almost one month to collect this data using automated YouTube video playout, network QoS configuration and data collection. Our dataset was built by playing a video using YouTube API in JavaScript, in a controlled environment, in which the network QoS features comprising *throughput*, *delay* and *packet loss rate*, were varied in the *download direction* using DummyNet [16]. The overall labeling procedure for each experiment is divided into three main steps:

1. For each QoS instance in the pool, we enforce the QoS configuration (throughput, delay and packet loss rate) using DummyNet in the downlink direction.
2. Then we perform the experiment with the enforced network conditions, i.e., we run the YouTube video and we collect from within the browser the application-level metrics of initial join time, number of stalling events and the duration of stalls;
3. Finally, we use an objective mapping function for mapping these application-level measurements to the final application QoE value (to be explained later, but for now we use 0 to 1 for binary and 1 to 5 for multiclass mapping).

We define two experimental spaces, one for the instances pool, $\mathcal{P}$, and the other for the validation set, $\mathcal{V}$. The experimental space for $\mathcal{P}$ ranges from 0 to 10 Mbps for throughput, 0 to 5000 ms for Round-Trip Time (set to two-way delay) and 0 to 25% for packet loss rate. For the validation set, $\mathcal{V}$, we use a range from 0 to 10 Mbps for throughput, 0 to 1000 ms for RTT and 0 to 10% for packet loss rate. This reduction is meant to lower the number of instances that can be easily classified, while concentrating validation on challenged instances around the borders between classes.

Within the experimental space $\mathcal{P}$, we first generate a uniformly distributed pool of around 10,000 unlabeled network QoS instances. We obtain the corresponding labels using the above mentioned procedure of controlled experimentation. Similar procedure is adopted to generate 800 samples within the experimental space of $\mathcal{V}$ to obtain independent validation sets for each classification scenario (binary/multiclass), equally distributed among the classes.

While performing the experiments, we make sure that the impairments in the network QoS are only due to the configured parameters on DummyNet. We confirm this by verifying before every experiment that the path to YouTube servers has zero loss, low RTT (less than 10 ms) and an available bandwidth larger than the configured one on DummyNet.

3.5. Mapping Function

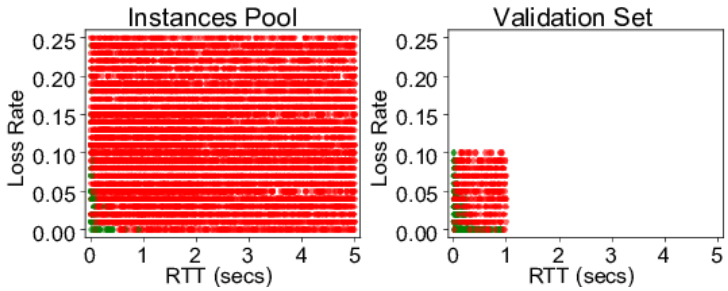
We label our datasets using controlled experiments for both the binary and the multiclass classification scenarios. For the former case, QoE of YouTube is classified into two labels; *Good*, if there are no interruptions and *Bad* if there are interruptions of the playout. This mapping relationship is used to translate the application level measurements in the pool to binary QoE labels. Owing to the large sample space for the pool, the resulting dataset was highly unbalanced with less than 1% of *Good* samples.

For the multiclass scenario, we rely on integer labels ranging from 1 to 5, based on the ITU-T Recommendation P.911 [17]. The QoE labels quantifying the user experience are directly related to the application level metrics of overall stalling time and initial join time [18]. The larger these times, the worse the QoE. Based on this information, we define the multiclass QoE label as a function of the total buffering time calculated as the sum of the initial join time and the overall stalling time. Prior work on subjective YouTube QoE has suggested an exponential relationship between buffering time and QoE [18]. Hence we define $QoE_{multi} = \alpha e^{-\beta t} + 1$, where t is the total buffering

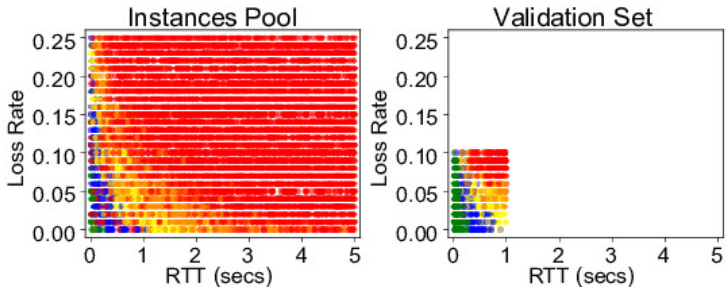
time (in seconds) for each experiment. The values of the factors α and β are calculated based on some assumptions of the best and worst scenarios for QoE. For example, for the best scenario, the QoE is maximum of 5 for zero buffering time which leads to $\alpha = 4$. And for the worst scenario, we consider a threshold for the total buffering time, beyond which the QoE label would be less than 1.5. We define this threshold to be 60 seconds which is equal to half of the total duration of the video used in our case. The value of this threshold is based on the results of a user engagement study (based on 23 million views) [19] which illustrates the video abandonment rate going upto more than 80% for a join time of 60 seconds. With this threshold, we get $\beta = 0.0347$. We would like to emphasize here that this mapping function and its given parameter values are not the final model for YouTube, rather these values are used as an example to get multiple output classes, yet well capturing the dependency between QoS and QoE, over which we can test active learning. Of course, with different mapping relationships, the models built will also be different, however the underlying active sampling methodology would remain the same.

The visual representation in Figure 1 shows a projection of the experimental space over the two features Round-Trip Time (RTT) and packet loss. For both the binary and the multiclass datasets, the relationship of the QoE labels with network QoS is evident – we can visualize a decision boundary over this projection. Here, samples having higher QoE are clustered in the lower regions of the RTT and packet loss space, with an apparent monotonicity in the variation of the QoE labels in particular for the multiclass set. Note here that the projection over the throughput feature, that we do not include here for lack of space, did not show such a clear clustering.

Figure 2 shows the CDF of the features in the pool, $\mathcal{P}$, where the relationship of the QoE w.r.t the individual network QoS features is highlighted. It can be seen that the effect of RTT and packet loss on the QoE is more pronounced as compared to throughput. In fact, for the case of throughput, the CDF is linear for all classes and shows a threshold effect in case of binary classification, i.e., for all videos that do not stall (*Good* samples in binary classification), the network has a throughput higher than around 1.6 Mbps which can be understood as the bit rate of the given YouTube 720p video. For such visualizations, if we look at the other two QoS features, we can see that the RTT is always lower than 1 second and packet loss remains below 10% for 90% of the*Good* cases. This gives a brief idea on how the network should be provisioned for the given YouTube 720p video to play out without

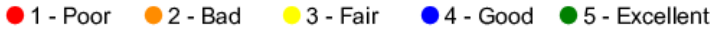


(a) Binary classification



(b) Multiclass classification

(c) Binary QoE labels



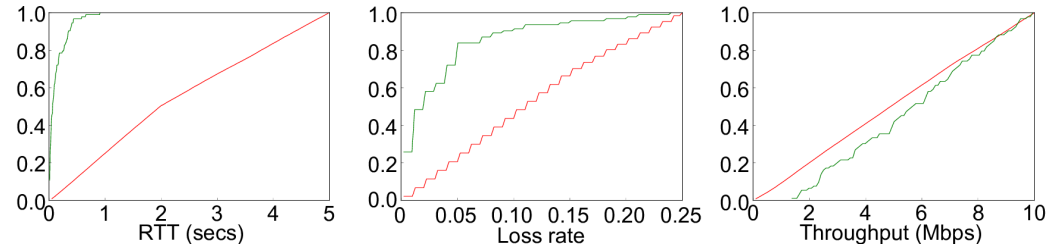
(d) Multiclass QoE labels

Figure 1: Visual representation of the datasets

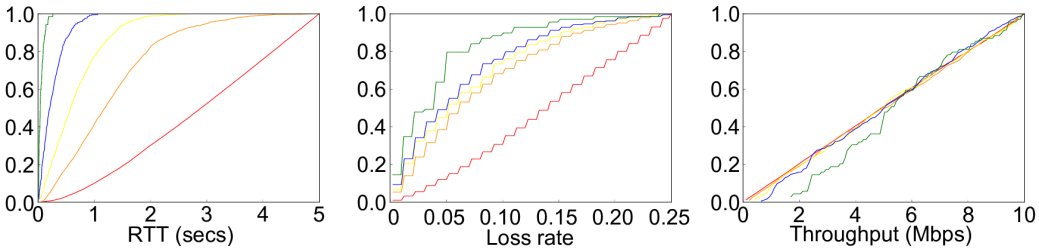
any buffering event.

3.6. Performance Analysis

We compare the performance of the active learning algorithms with uniform sampling where the selection of the instances from the pool $\mathcal{P}$ is random. For all scenarios, the same initial training set is used, which comprises of just two instances from different classes. Along algorithm execution, each iteration corresponds to picking an instance from the pool and adding it to the training set. After each iteration, the size of the training set increases by one, so the number of iterations also corresponds to the size of the built



(a) Binary classification



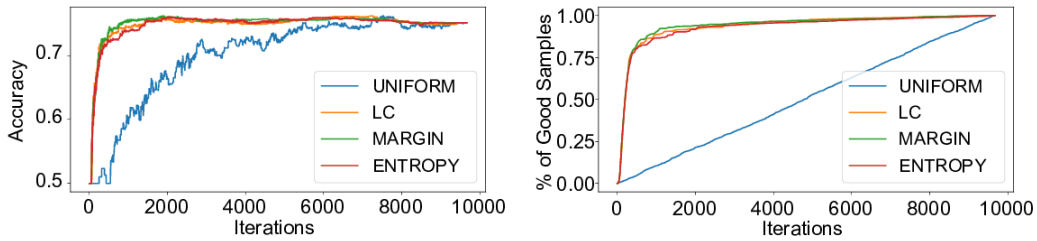
(b) Multiclass classification

Figure 2: CDF of the samples in the pool

training set, $\mathcal{T}$. The entire algorithm is run for all the samples in the pool and the performance results shown are the average of 20 independent runs. For ease of understanding, the naming convention for the sampling scenarios are mentioned below:

1. UNIFORM: sampling is random.
2. LC: sampling is done based on the utility measure of Least Confidence.
3. MARGIN: sampling is done based on the utility measure of Minimal Margin.
4. EYNTROPY: sampling is done based on the utility measure of Maximum Entropy.

The overall accuracy of the Decision Tree QoE model using the different sampling techniques applied to the binary classification case is shown in Figure 3a. It can be seen that all the uncertainty sampling techniques have a close performance and they all outperform uniform sampling by achieving a higher accuracy with a much lower number of samples in the training phase. This ratifies our earlier description of active learning, that training



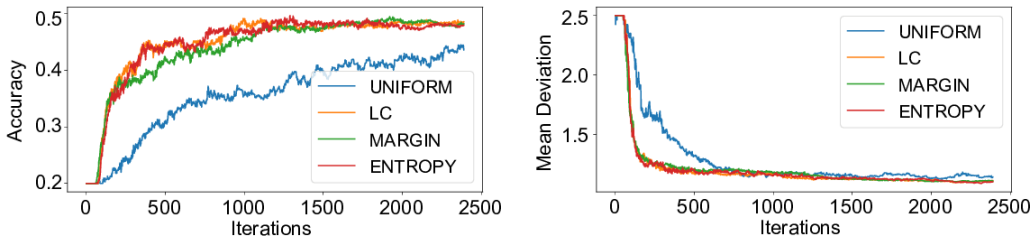
(a) Learner accuracy (b) Cumulative ratio of *Good* samples

Figure 3: Binary classification results

with samples closer to the decision boundaries is much more beneficial as compared to samples away from them. The accuracy starts from its initial value of 0.5 (learner predicts initially everything as *Bad*) and then quickly achieves the optimum accuracy of around 75%, whereas the accuracy using uniform sampling becomes comparable much later on. The performance gain of our methodology can be gauged by the fact that we are now able to build a QoE model much faster and by using almost one order of magnitude fewer experiments compared to randomly choosing the network QoS configurations for experimentation (uniform sampling). The reduction in the number of experiments means that we gain in the overall cost involved in building the model.

It has to be noted that this observed value of model prediction accuracy is dependent on the used pool and the validation sets in our experimentation. For different validation sets, the learner accuracy might be slightly different. But still, one can question why the accuracy of the learner is only up to 75%, and does not reach higher values as in prior works in [20], [7] and [21] where similar ML models have accuracy ranging from 84% to 93%. The reason for this difference lies in the type of features used. The prior works rely on features obtained directly from traffic traces (*in-band* features), whereas our work considers features that are *out-of-band* w.r.t the traffic itself. As a result the obtained accuracy for such models is higher. This will also be verified in Section 4.6, where we will show that *in-band* features indeed give better accuracy compared with *out-of-band* network features.

From Figure 1a, we can see that the underlying distribution of the binary labels in the pool is highly unbalanced with all samples from the minority class (i.e., *Good* class in our case) located in a single small region in the



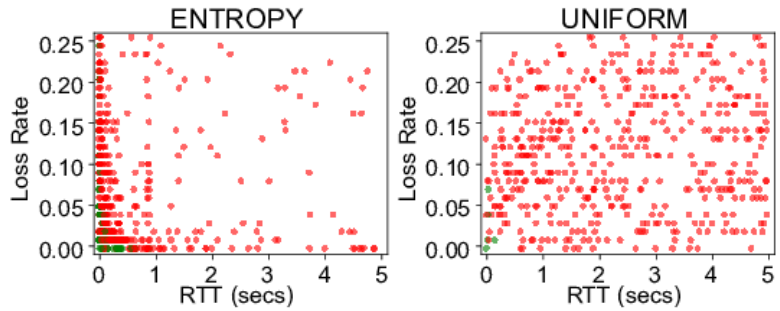
(a) Learner accuracy (b) Mean deviation

Figure 4: Multiclass classification results

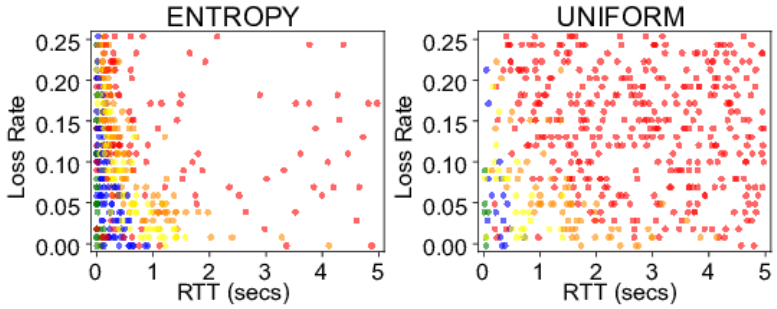
corner of the experimental space. Due to this reason it can be expected that the active learner tends to focus on this region in picking its points for labeling which would result in picking the instances from the minority class much faster as compared to uniform sampling. To verify this, we plot the cumulative ratio of the selected instances from the minority class (*Good* class in our case) in Figure 3b, where we can indeed see that the minor points are picked much faster by active sampling as compared to uniform sampling. This results in having adequate number of samples from each class to get a better accuracy.

The results for the multiclass classification case are shown in Figure 4a for 25% of the pool size to focus on the initial part, where the gain of active sampling over uniform sampling is more visible. A noticeable observation is that the accuracy of the model is much lower than in the binary classification case. The reason being that the number of borders between the multiclass labels is larger as compared to binary classification, so the number of ambiguous regions (the challenged regions where the validation is performed) is also larger, thus the accuracy of the multiclass QoE model is decreased. Having said that, the converged value of the *Mean Deviation* shown in Figure 4b is close to 1, which means that most of the mis-classifications are only to the adjacent classes. As for the active learner's convergence rate, we can see that it again performs better than uniform sampling as it achieves a lower mean deviation with fewer training samples.

Finally, the difference between the selected instances using active learning (ENTROPY) and uniform sampling is visualized in Figure 5, at 5% of the pool size. It is evident that in case of uniform sampling, the training set comprises of points scattered throughout the sample space whereas for



(a) Binary classification



(b) Multiclass classification

Figure 5: Training set samples at 5% of the pool size

active learning, points are mostly selected from a region near the decision boundaries. More importantly, unrealistic extreme points of very high RTT/loss rate are seldom sampled by the active learner compared to uniform sampling. Therefore, active learning allows experimentation to be carried out with more relevant network configurations compared to uniform sampling of the space. Hence, we are able to build accurate models quickly, without the need to know any prior information on what a relevant experimental space should be for the given target application.

3.7. The effect of pool size on model accuracy

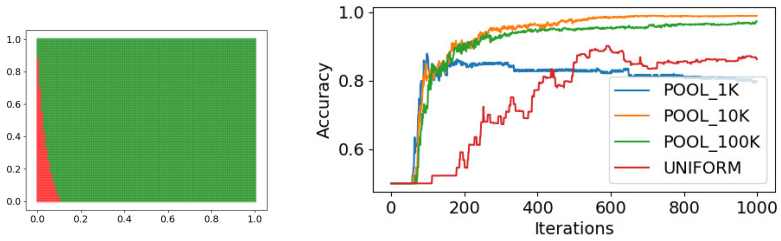
In our validation of pool based sampling, we used an arbitrary pool size of 10k samples. A question arises on the appropriate pool size to be used for a given experimentation scenario without having any a priori knowledge of the feature space. Ideally the pool used should be as large as possible. However, as we will show subsequently, uncertainty sampling with very large pools can cause the learner to block for long time in uncertain regions of space resulting

in an inaccurate ML model. To verify this problem, we perform an analysis of pool based sampling on a *synthetic* dataset with different pool sizes (1k, 10k and 100k samples). We consider a binary classification scenario with two input features and use our pool based uncertainty sampling methodology (Section 3.1) to build the training set. The ground truth (the labeler) is represented by a function whose decision boundary has a non-linear shape with a monotonic transition in label classes, similar to what we observe in the training sets of Figure 1. The function we use to represent such a boundary is given by $y = 0.1 + (1 - x)^{20}$ (Figure 6a) assuming that the features take values of 0 or 1. Furthermore, to mimic real systems, we add random noise in the region near the decision boundary; the noise is added within 0.1 units from the decision boundary.

Using our methodology, we obtain the performance curves shown in Figure 6b, which show the accuracy of the ML model (a Decision Tree model with 20 samples per leaf) obtained for 20 independent runs using the utility measure of ENTROPY. We observe that the performance of the model is indeed dependent on the size of the pool. For a pool of 1k instances, the performance is poor due to the limited number of instances. If we increase the pool to a reasonably large value of 10k samples, we observe a significant gain over uniform sampling. However, if we further increase the pool size to 100k samples, the performance deteriorates, instead of improving further. This is because of the blocking issue as mentioned before. Thus, for any controlled experimentation environment, the experimenter either has to first define a pool of a given size and then apply active learning on it. A very large pool may result in the blocking issue resulting in lower modeling accuracy while a small pool may also lead to low accuracy. The issue here is that we cannot ascertain the right pool size a priori without performing the experiments. So in order to avoid this problem of choosing the right pool size, we re-formulate our active learning approach in the next section, where instead of defining any pool, we sample the space directly by cutting the space into regions and use a probabilistic approach of choosing the experimentation scenarios to avoid the learner to get blocked in the noisy parts of the feature space.

4. Relaxing the Pool Assumption and Solving the Blocking Issue: An Online Sampling Approach

Our approach to overcome the aforementioned problem regarding the pool size selection and sampler blocking can be summarized in Figure 7. Instead



(a) Ground Truth (b) Accuracy

Figure 6: Comparison of accuracy with different pool sizes

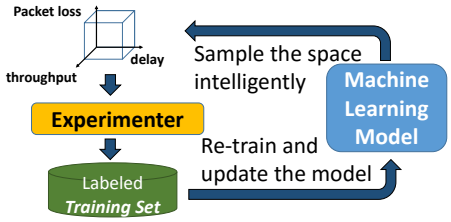


Figure 7: Active sampling

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. And as already stated before, 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 here. A DT learner splits the space using a set of internal nodes and edge leafs disposed in a tree structure and 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

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

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 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 new 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

randomly select $\mathbf{x}^* \in l^*$
experiment using $\mathbf{x}^*$ to obtain label y^*
add $\langle \mathbf{x}^*, y^* \rangle$ to $\mathcal{T}$
$\Theta = \textbf{train}(\mathcal{T})$
end for

4.1. Stopping Criterion

In active learning, 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 [22]. For the accuracy measure, in general, a separate validation set that is needed to represent the ground truth and 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 can demonstrate model convergence by observing the change in the uncertainty (Eq. 1) and the confidence measures (Section 4.2) over the course of the iterations. As we will show in Section 4.5, 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.

4.2. Gauging Model Quality using Model Confidence

Here, 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 $c_i = \max \left\langle p_i^{(1)}, \ldots p_i^{(M)} \right\rangle$, 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

$$w_i = \frac{1}{X_1 X_2 ... X_N} \int \cdots \int_{x_n^{LOW_i}}^{x_n^{HIGH_i}} dx_1 \dots dx_N, \tag{2}$$

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.

4.3. Evaluation

We implement our HYBRID sampling methodology shown in Figure 7, in a controlled experimentation framework for modeling YouTube video QoE based on the same methdology of Section 3.4. The framework consists of a *client* node that performs the experiments, and a *controller* node that 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 video ID: oFkulzWMotY and obtains its corresponding QoE, 3) sends back the labeled tuple of the enforced QoS and the output QoE to the controller, which then updates the DT model locally. For the QoE, we consider the binary case to ease the illustration of results, i.e., *Bad* if the video suffers from stalling (stalling duration of more than 1000 ms) 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 major stalls (stalling duration of less than 1000 ms). In the subsequent analysis, the *green* color refers to the *Good* class and the *red* color refers to the *Bad* class.

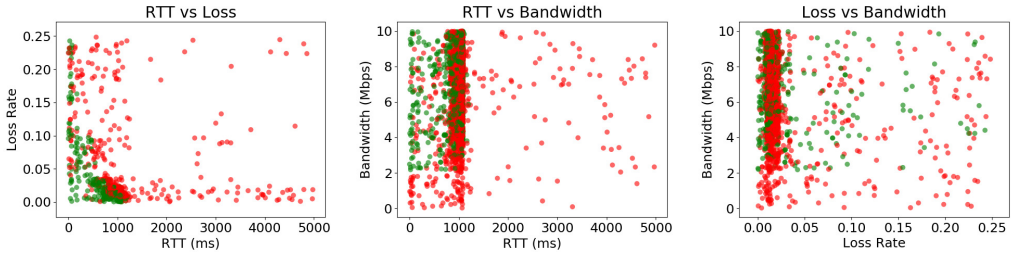
To evaluate our methodology, we collect two datasets, $\mathcal{D}_{MAXENTROPY}$ and $\mathcal{D}_{HYBRID}$ based on two different sampling approaches, MAXENTROPY and HYBRID. In MAXENTROPY, the region selected for labeling is the one which has maximum entropy according to the ML model under construction whereas in HYBRID, the criterion for selecting a region is based

on the entropy-based probabilistic measure. Each dataset is comprised of 4000 YouTube video playouts in our 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 and the previously described pool based sampling. We then illustrate how the model built with $\mathcal{D}_{HYBRID}$ changes 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.

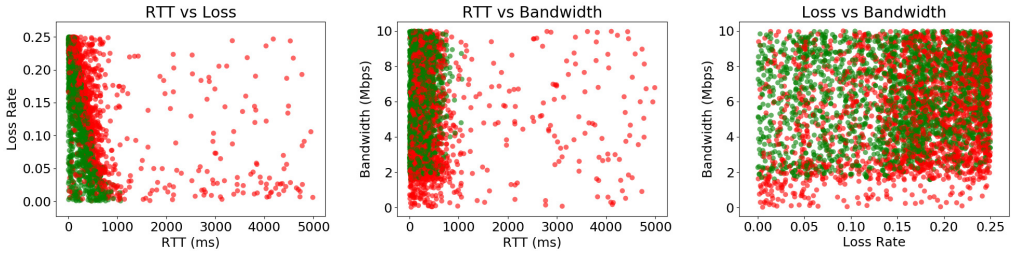
The visual representation of the collected datasets is shown in Figures 8a and 8b. In these figures, sampled network scenarios are projected over different pairs of QoS axes. We can clearly see a significant difference between the two approaches in the regional distribution of the sampled scenarios for 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 models trained with these datasets are also different. This is highlighted in Figure 9a 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.

4.4. 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



(a) MAXENTROPY



(b) HYBRID

Figure 8: Visual representation of the collected datasets

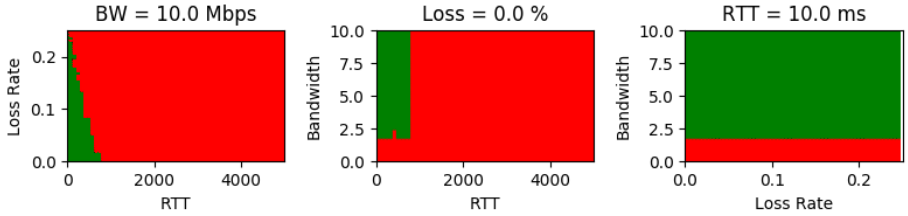
accuracy score is then the average of these k scores. We plot the F1-Score[4] based on cross validation ($k = 3$ with a data split ratio of 80:20 for training and validation) that is computed at each iteration of our experimentation in Figure 10, where we can see that HYBRID achieves a better accuracy for both classes compared to MAXENTROPY[5]. However, an important observation here is that as we increase the number of experiments, the cross validation F1-Score begins to decrease. For MAXENTROPY, that drop is significantly more important. The reason for it is that as we keep on experimenting, more and more scenarios will be 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 Sec-

[4]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.

[5]The results shown in Figures 10 to 13 are based on moving average with a window size of 100 iterations to smooth out the curves for better presentation.

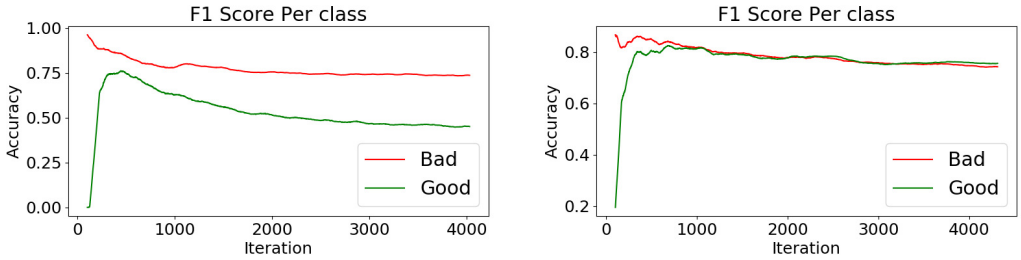
(a) MAXENTROPY



(b) HYBRID

Figure 9: Visual depiction of the DT ML Model

tion 4.1 – to better gauge the quality of the models. Figure 11 shows this comparison for DT models built with HYBRID and MAXENTROPY. We also add to this figure the result of pool based sampling of Section 3. 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 1 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 Figure 8). In fact, in our case, the performance of the model w.r.t 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.



(a) MAXENTROPY (b) HYBRID

Figure 10: Comparison of the F1-Scores per class b/w HYBRID and MAXENTROPY

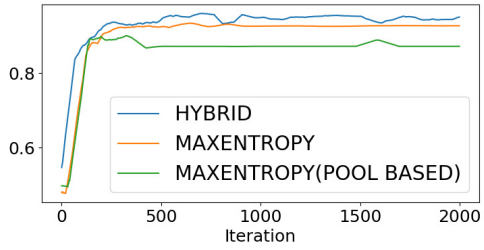


Figure 11: The weighted confidence measure

4.5. 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 Figure 12. We can observe that 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 Figure 12 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 Figure 12 is a weighted sum of the entropies of all leafs with each leaf assigned a weight according to the leaf's geometric volume. 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.

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

Table 1: Model confidence per class after 3000 iterations

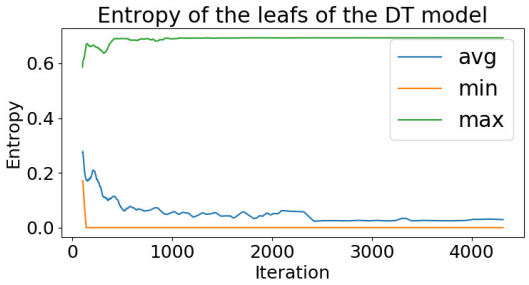


Figure 12: Variation of Entropy

We now study convergence from another perspective. Figure 13 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 Figure 12 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 the above metrics is an indication of 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 Figure 11. Experiments beyond this point have almost no impact on the model performance.

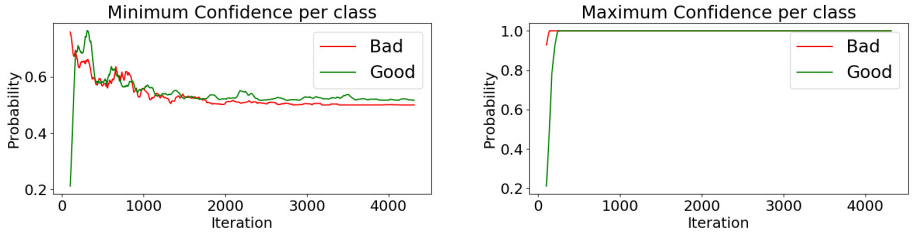


Figure 13: Classification probabilites (confidence) of the DT leafs per class

4.6. Inband and Outband QoS feature performance for different ML algorithms

An important question is why the F1-Scores obtained in Figure 10 are only up to around 75% whereas similar QoE models for YouTube [7], [23] 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 [1], 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, the maximum, the standard deviation, the 20th, the 50th and the 90th percentiles. Overall, 6 statistical measures for the 4 network measurements, which amount to a total of 24 features are given as input to a new ML QoS-QoE model. The improvement in the accuracy by modeling with these inband features can be confirmed by comparing results in Table 2 where a gain of 15-20% can be noticed. We consider this difference as the cost of QoE prediction and we will be looking in the future into ways to reduce it further. Note here that in this table, different learners are used to confirm the insensitivity of our

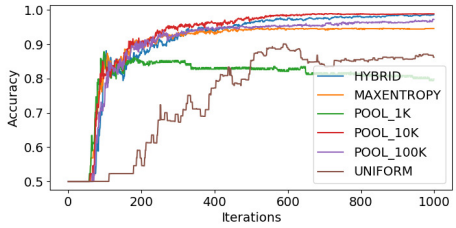


Figure 14: Comparison of accuracy for synthetic dataset (non-linear function)

results to the choice of Decision Trees as a machine learning technique.

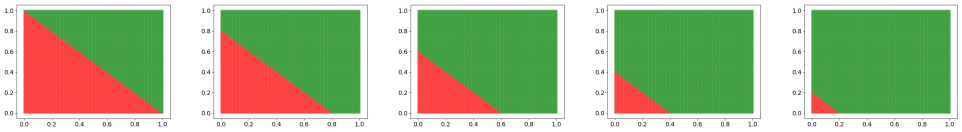
Classifier	Class	Outband QoS			Inband QoS		
		Prec	Recall	F1	Prec	Recall	F1
Nearest Neighbors	Good	57%	55%	56%	89%	90%	90%
	Bad	58%	60%	59%	90%	89%	90%
Decision Tree	Good	74%	73%	73%	96%	93%	94%
	Bad	74%	75%	74%	93%	96%	95%
Random Forest	Good	76%	78%	77%	96%	96%	96%
	Bad	78%	76%	77%	96%	96%	96%
AdaBoost	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 2: Performance of popular ML algorithms with *Inband* and *Outband* QoS features

4.7. Quantifying the Gain of Active Sampling with Synthetic Datasets

We further validate the HYBRID approach over the synthetic non-linear function described in Section 3.7 to obtain the performance curves of Figure 14. This figure shows the performance comparison of both the Pool based and the HYBRID approaches for modeling a non-linear decision boundary. It can be seen that HYBRID and POOL_10K perform similarly while MAXENTROPY performs poorly. Note that MAXENTROPY is equivalent to pool based sampling with an infinite pool size, thus has lower accuracy than POOL_10K.

From Figure 14, it is also evident that active sampling techniques clearly outperform uniform sampling. However, the overall gain of active sampling



(a) C=0 (b) C=0.2 (c) C=0.4 (d) C=0.6 (e) C=0.8

Figure 15: The synthetic linear functions

C	Class 0	Class 1
0	50%	50%
0.2	32%	68%
0.4	18%	82%
0.6	8%	92%
0.8	2%	98%

Table 3: Covered area for each class according to a tunable linear function

for any new given experimental scenario depends on the distribution of the ground truth labels and the complexity of the decision boundaries in the underlying feature space. In order to verify this claim, we analyze the performance of our techniques over different synthetic feature spaces (the methodology of Section 3.7 for synthetic datasets is again used here but with a linear decision function) to see the difference in the performance gain of active sampling over uniform sampling. Here, we vary the distribution of the ground truth labels in the feature space by changing the position of the decision boundary which is represented by the linear function given by $y = 1 - x - C$. The constant C is varied in steps of 0.2 from 0 to 1 to obtain five different spaces. The resulting distribution of the output labels is visualized in Figure 15 while the area covered by each class is given in Table. 3. Considering such scenarios, and supposing we have a budget of 500 or 1000 experiments, we showcase next the performance gain of active sampling in terms of modeling accuracy. The results are given in Figure 16 where we can see that the performance gain increases as the distribution of the labels gets more and more unbalanced in space. Furthermore, for increased budget, the performance gain decreases. Note here that these results are for a simple linear decision boundary; for complex decision boundaries, the gain would be even more pronounced. From these results it may be concluded that the actual gain of active sampling over uniform sampling is not fixed, rather it is depen-

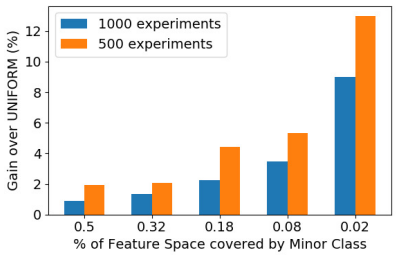


Figure 16: Gain over Uniform Sampling for different feature spaces

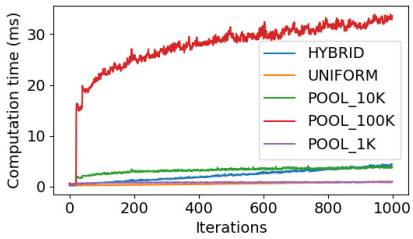


Figure 17: Computation complexity in milliseconds of each sampling iteration. Results obtained on a PC using Intel Core i7-6600U (2.60 GHz) CPU with 15 GB of RAM and using Python 2.7 on Fedora release 24.

dent on the complexity of the feature space and the available experimentation budget.

4.7.1. Comparing the computational overhead of active sampling

Since active sampling involves computing and ranking the model uncertainties at each iteration, it is expected that these additional computations will incur a computational overhead. To evaluate this overhead, we plot the computational time observed at each iteration in modeling the synthetic linear function (Figure 15b) in Figure 17. It can be seen that the overhead due to active sampling is of the order of few milliseconds only. For pool based sampling, the computational time increases as we increase the size of the pool, while for HYBRID, it increases as the number of leafs in the DT model grows. Considering that experiments in QoE modeling scenarios consume time in the order of minutes, a computational overhead of few hundreds of milliseconds is negligible, especially if compared to the time saved in carrying out the experiments. Also, the overhead is dependent on the type of ML algorithm used, in our work we use Decision Trees which are faster to train

compared to other learners such as Neural Networks or SVM.

5. Implementation of active sampling in real production networks

While our active sampling approach targets controlled experimentation environments, it can also be used in real-world production networks (e.g., YouTube, Netflix etc.) as well. Here, we can use pool based sampling where the pool of unlabeled data can be the set of network measurements observed for each of the end users' web/video/audio streaming sessions. Then, the task of labeling this entire measurement dataset with its corresponding QoE label can be optimized using active learning by choosing the samples for labeling (by user feedback) for which the underlying ML model (at the backend server) is uncertain. Only for such measurements, the server should ask the end user to feedback the corresponding video session in terms of the MoS observed (the labeling procedure in active learning). This labeled sample is then added to the training set at the backend to update and build the model iteratively and efficiently. This schema is relevant to any service provider, Youtube, Netflix, Whatsapp, Skype etc., where end users are solicited for feedback at the right moment for samples for which the model is uncertain. On one hand active sampling will reduce the number of solicitations (user feedback), and so will bother less the end users, and on the other hand it will produce efficiently a model that can be used later by these players to model and estimate Quality of Experience of their customers, and operate over the content and the network to optimize Quality of Experience.

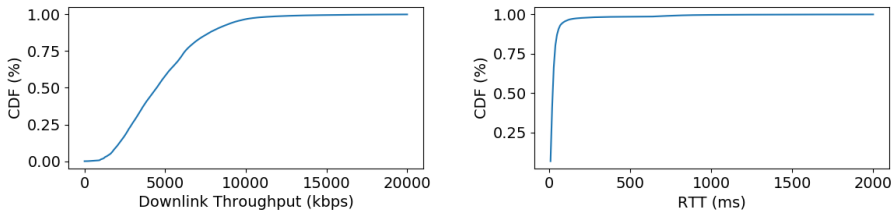
5.1. Application of pool based sampling on an open network measurement dataset

To illustrate the performance of active sampling in a scenario with real networks, we apply pool based sampling on a real network measurements dataset used as our pool of data. Specifically, we use the dataset provided by FCC [24] which has more than 3 million measurements of downlink bandwidth and latency for broadband video streaming tests performed with several ISPs in August 2016. The CDF of this dataset is shown in Figure 18 where the median of the downlink throughput occurs at around 5 Mbps while the distribution for RTT is highly skewed with 95% of samples having RTT values less than 92 ms.

We use this dataset to build both our pool and the validation set with a data split ratio of 95:5 resulting in a validation set with over 20k samples and

a pool of more than 3 million samples. We then repeat our sampling methodology described in Section 3.1 and obtain at each iteration the accuracy of the model being built using active sampling compared to uniform sampling of the FCC pool. The labeling is done using the ML model of Figure 9b with loss rate set to zero. The results are shown in Figure 19 where we can indeed see that pool based sampling achieves higher accuracy faster than uniform sampling, ratifying the feasibility of using active sampling techniques on realistic data pools as well.

Note that the gain in terms of the experiments saved in this case is in the order of hundreds of experiments which might seem low. This is due to the fact that in this particular scenario, the decision boundary is a horizontal or a vertical line (Figure 9b for bandwidth vs. RTT). Learning such boundaries is easier and would require fewer experiments compared to learning more complex boundaries such as the one in Figure 9b (loss rate vs. RTT).



(a) Downlink Throughput (b) RTT

Figure 18: CDF of the FCC dataset

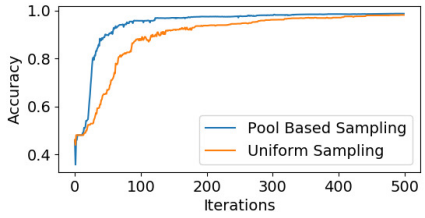


Figure 19: Application of pool based active sampling over FCC dataset. The measure of accuracy is the average of the F1-scores per class and the scores at each iteration correspond to the average of 20 independent runs.

6. Related Work and Discussion

A detailed survey of the active learning techniques can be found in [9]. To the best of our knowledge, the work presented in this paper is the first attempt to apply active learning techniques in the context of QoE modeling versus network QoS conditions based on controlled experimentation. Prior work on the application of active learning in the domain of QoE is discussed under different contexts other than controlled experimentation. For example, the authors in [25] propose the application of active learning to tackle the scale of subjective experimentation in addressing the problem of biases and variability in subjective QoE assessments for video. Other works such as in [26] use active sampling in choosing the type of subjective experiments for image quality assessments.

Similar experimental selection problems have also been considered in other contexts than QoE and ML. For example authors in [27] propose a technique to maximize the information gain while minimizing the duration of the bench-marking/experimentation process. Their technique gives higher importance to placing experiments in regions where larger changes in the response variable exist. The regions have fixed volume in the experimental space. Their method is similar to uncertainty sampling in terms of targeting uncertain regions of space but differs w.r.t the output variable; in our work we consider discrete integral output labels in a supervised ML classification scenario while in their case they consider a continuous output variable. Also, they consider fixed size regions while we use variable sized geometric regions of space represented by the leafs of a DT.

Our online active sampling methdology relies on using a probablistic variable uncertainty criterion to pick the most suitable region for experimentation. A similar criterion is used in [28] in a stream based selective sampling scenario with a variable threshold of uncertainty to target a problem of *dynamic* decision boundaries (referred to as concept drift in active learning literature). This problem is different than ours in that we consider *noisy* decision boundaries. Also, in our work, we do not receive any unlabeled data, rather we *synthesize* the data for labeling.

In [29], authors use active learning to target crowdsourced QoE modeling scenarios whereas we consider controlled experimentation. They highlight the problem of subjectivity in the QoE scores which leads to degraded performance with active learning. A similar problem can occur in our controlled experimentation approach where a network fluctuation can deviate the QoS

observed from to the enforced QoS, thus creating noise in the training set which would lower the performance of the model being built. But if these fluctuations remain minimal, their impact on the performance of our solution will also be minimal. As a solution to this problem of dynamic network conditions, one can imagine redoing our work but with as input features not only the mean network QoS values, but also their variability, and make sure to measure these means and variability while using the model (training and QoE estimation).

Supervised ML has been used extensively in literature for QoS-QoE modeling; a detailed survey of such models is provided in [30]. While we also use supervised ML in our work, our focus is not on the final QoE models, but on reducing the cost of building such models by active sampling. Furthermore, the conventional approach of using supervised ML in QoS-QoE modeling is to use the training data that is either collected in the wild [31], [7] or built by controlled experimentation [32]. For data collected in the wild, the model suffers from the biasness of the underlying data source, while for the case of controlled experimentation, the similarity in the experimental space is not fully exploited. Our paper fills these gaps by proposing intelligent sampling that allows building models quickly in a general way not biased to any particular data source.

The work in this paper presents a validation of active sampling for a video-on-demand (VOD) QoE modeling scenario where the same content(s) can be played in multiple experiments over time. However, For live video streaming, the content of the video cannot be stored and can change over time. If only the network QoS features are used to model live video QoE, the underlying decision boundaries are expected to be noisier with larger regions of uncertainty in the network feature space compared to modeling for VOD. Active sampling in such a scenario will still be useful but may converge to a lower accuracy compared to a modeling for a stored video content.

7. Conclusion and Future Work

In this paper, we showcased the benefit of using active learning to speed up the process of building QoE models using controlled experimentation. Our work validated active learning over a sample YouTube QoE modeling use case and our results showed improvement over conventional uniform sampling of the experimental space. In the future, we plan to incorporate active learning in practice such that the datasets of the QoS-QoE mappings are

directly obtained using active learning for a variety of applications such as VoIP (Skype, Viber, Whatsapp etc.) and Web Browsing. Each application scenario requires its own QoE definition which may be subjective (with real users) or based an objective functions. Objective functions can be e.g., PESQ for VoIP or Google's SpeedIndex for Web QoE. So, for different applications, the QoS-QoE modeling setup for data collection will be different, however the underlying active sampling framework would remain same.

References

[1] ACQUA: Application for prediCting Quality of User experience at Internet Access, `http://project.inria.fr/acqua/` (2017).

[2] T. Spetebroot, S. Afra, N. Aguilera, D. Saucez, 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, 2015, pp. 1–6.

[3] X. Yin, A. Jindal, V. Sekar, 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] H. Mao, R. Netravali, M. Alizadeh, Neural adaptive video streaming with pensieve, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '17, ACM, New York, NY, USA, 2017, pp. 197–210.

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

[6] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, 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, ACM, NY, USA, 2014, pp. 367–379.

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

[8] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, H. Zhang, CFA: A practical prediction system for video qoe optimization, in: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), USENIX Association, Santa Clara, CA, 2016, pp. 137–150.

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

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

[11] M. J. Khokhar, N. A. Saber, T. Spetebroot, 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, Internet QoE '17, ACM, New York, NY, USA, 2017, pp. 31–36.

[12] M. J. Khokhar, T. Spetebroot, C. Barakat, An online sampling approach for controlled experimentation and qoe modeling, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

[13] D. Angluin, Queries and concept learning, Machine Learning 2 (4) (1988) 319–342.

[14] F. Pedregosa, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[15] T.-F. Wu, C.-J. Lin, R. C. Weng, Probability estimates for multi-class classification by pairwise coupling, J. Mach. Learn. Res. 5 (2004) 975–1005.

[16] L. Rizzo, Dummynet: A simple approach to the evaluation of network protocols, SIGCOMM Comput. Commun. Rev. 27 (1) (1997) 31–41.

[17] ITU, Subjective audiovisual quality assessment methods for multimedia applications, ITU-T Recommendation P.911.

[18] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, R. Schatz, Quantification of youtube qoe via crowdsourcing, in: Multimedia (ISM), 2011 IEEE International Symposium on, 2011, pp. 494–499.

[19] S. S. Krishnan, R. K. Sitaraman, Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs, in: Proceedings of the 2012 Internet Measurement Conference, IMC '12, ACM, New York, NY, USA, 2012, pp. 211–224.

[20] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, H. Yan, Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements, in: Proceedings of the 15th Workshop on Mobile Computing Systems and Applications, HotMobile '14, ACM, New York, NY, USA, 2014, pp. 18:1–18:6.

[21] I. Orsolic, D. Pevec, M. Suznjevic, L. Skorin-Kapov, A machine learning approach to classifying youtube qoe based on encrypted network traffic, Multimedia Tools and Applications 76 (21) (2017) 22267–22301.

[22] J. Zhu, H. Wang, E. Hovy, M. Ma, Confidence-based stopping criteria for active learning for data annotation, ACM Trans. Speech Lang. Process. 6 (3) (2010) 3:1–3:24.

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

[24] Federal Communications Commission. 2016. Raw Data - Measuring Broadband America., `https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016` (2016).

[25] V. Menkovski, G. Exarchakos, A. Liotta, Tackling the sheer scale of subjective qoe, in: L. Atzori, J. Delgado, D. Giusto (Eds.), Mobile Multimedia Communications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 1–15.

[26] P. Ye, D. Doermann, Active sampling for subjective image quality assessment, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 4249–4256. doi:10.1109/CVPR.2014.541.

[27] R. Hashemian, N. Carlsson, D. Krishnamurthy, M. Arlitt, Iris: Iterative and intelligent experiment selection, in: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ICPE '17, ACM, New York, NY, USA, 2017, pp. 143–154.

[28] I. liobait, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 27–39.

[29] H. Chang, C. Hsu, T. Hobfeld, K. Chen, Active learning for crowdsourced qoe modeling, IEEE Transactions on Multimedia (2018) 1–1.

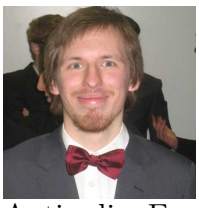
[30] S. Aroussi, A. Mellouk, Survey on machine learning-based qoe-qos correlation models, in: 2014 International Conference on Computing, Management and Telecommunications (ComManTel), 2014, pp. 200–204.

[31] P. Casas, M. Seufert, N. Wehner, A. Schwind, F. Wamser, Enhancing machine learning based qoe prediction by ensemble models, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018, pp. 1642–1647.

[32] V. Vasilev, J. Leguay, S. Paris, L. Maggi, M. Debbah, Predicting qoe factors with machine learning, in: 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1–6.

Muhammad Jawad Khokhar is a Ph.D. student in the DIANA team at INRIA, Sophia-Antipolis, France. He received his Bachelors degree in Electrical (Telecommunication) engineering and his Masters degree in Electrical Engineering from National University of Sciences and Technology (NUST), Islamabad, Pakistan, in 2010 and 2012 respectively. His main research interests are in Internet measurements and traffic analysis, user quality of experience and machine learning.

Nawfal Abbassi Saber was a Research Intern in the DIANA team at INRIA, Sophia-Antipolis, France. He received his Masters degree in Computer Science from University of Nice Sophia Antipolis, France, in 2016. Currently he works as a Consultant at Amayas Consulting, France.



Theirry Spetebroot was a Research Engineer in the DIANA team at INRIA, Sophia-Antipolis, France. He received his Bachelors degree in Computer Software Engineering from University of Bologna, Italy, in 2013 and his Masters degree in Ubiquitous Networking from University of Nice Sophia Antipolis, France, in 2015. His main research interests are in Internet measurements and traffic analysis, machine learning and android development.

Chadi Barakat is a permanent research scientist in the DIANA team at INRIA, Sophia Antipolis, France, since 2002. He got his Electrical and Electronics engineering degree from the Lebanese University of Beirut in 1997, and his master, Ph.D., and H.D.R. degrees in Networking from the University of Nice, Sophia-Antipolis, France, in 1998, 2001 and 2009, respectively. He is senior member of the IEEE and the ACM. His main research interests are in Internet measurements and traffic analysis, user quality of experience and network transparency, new network architectures (ICN and SDN), and performance evaluation of computer networks.