Modeling Quality of Experience of Internet Video Streaming by Controlled Experimentation and Machine Learning

Muhammad Jawad Khokhar

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Modélisation de la qualité d'expérience de la vidéo streaming dans l'internet par expérimentation contrôlée et apprentissage machine

Muhammad Jawad KHOKHAR

Inria Sophia Antipolis

Présentée en vue de l'obtention
du grade de docteur en INFORMATIQUE
d'Université Côte d'Azur
Dirigée par : Chadi BARAKAT
Soutenue le : Octobre 15, 2019

Devant le jury, composé de :
Kavé SALAMATIAN, Professeur, University de Savoie
Jean-Louis ROUGIER, Professeur, TELECOM ParisTech
Isabelle CHRISMENT, Professeur, Université de Lorraine
Guillaume URVOY-KELLER, Professeur, UCA
Giovanni NEGLIA, Chercheur (HDR), Inria
Chadi BARAKAT, Directeur de Recherche (DR), Inria

Inria

Modélisation de la qualité d'expérience de la vidéo streaming dans l'internet par expérimentation contrôlée et apprentissage machine

Modeling Quality of Experience of Internet Video Streaming by Controlled Experimentation and Machine Learning

Jury:

Rapporteurs

Kavé Salamatian, Professeur, Université de Savoie
Jean-Louis Rougier, Professeur, TELECOM ParisTech

Examinateurs

Isabelle Chrisment, Professeur, Université de Lorraine
Guillaume Urvoy-Keller, Professeur, Université Côte d'Azur
Giovanni Neglia, Chercheur (HDR), Inria Sophia Antipolis

Directeur

Chadi Barakat, Directeur de Recherche (DR), Inria Sophia Antipolis

Abstract

Video streaming is the dominant contributor of today's Internet traffic. Consequently, estimating Quality of Experience (QoE) for video streaming is of paramount importance for network operators. The QoE of video streaming is directly dependent on the network conditions (e.g., bandwidth, delay, packet loss rate) referred to as the network Quality of Service (QoS). This inherent relationship between the QoS and the QoE motivates the use of supervised Machine Learning (ML) to build models that map the network QoS to the video QoE. In most ML works on QoE modeling, the training data is usually gathered in the wild by crowdsourcing or generated inside the service provider networks. However, such data is not easily accessible to the general research community. Consequently, the training data if not available beforehand, needs to be built up by controlled experimentation. Here, the target application is run under emulated network environments to build models that predict video QoE from network QoS. The network QoS can be actively measured outside the data plane of the application (*outband*), or measured passively from the video traffic (*inband*). These two distinct types of QoS correspond to the use cases of *QoE forecasting* (from end user devices) and *QoE monitoring* (from within the networks). In this thesis, we consider the challenges associated with network QoS-QoE modeling, which are 1) the large training cost of QoE modeling by controlled experimentation, and 2) the accurate prediction of QoE considering the large diversity of video contents and the encryption deployed by today's content providers.

Firstly, QoE modeling by controlled experimentation is challenging due to the high training cost involved as each experiment usually consumes some non-negligible time to complete and the experimental space to cover is large (power the number of QoS features). The conventional approach is to experiment with QoS samples uniformly sampled in the entire experimental space. However, uniform sampling can result in significant similarity in the output labels, which increases the training cost while not providing much gain in the model accuracy. To tackle this problem, we advocate the use of *active learning* to reduce the number of experiments while not impacting accuracy. We consider the case of YouTube QoE modeling and show that active sampling provides a significant gain over uniform sampling in terms of achieving higher modeling accuracy with fewer experiments. We further evaluate our approach with synthetic datasets and show that the gain is dependent on the complexity of the experimental space. Overall, we present a sampling approach that is general and can be used in any QoS-QoE modeling scenario provided that the input QoS features are fully controllable.

Secondly, accurate prediction of QoE of video streaming can be challenging as videos offered by today's content providers vary significantly from fast motion sports videos to static lectures. On top of that, today's video traffic is encrypted, which means that

network operators have little visibility into the video traffic making QoE monitoring difficult. Considering these challenges, we devise models that aim at accurate forecasting and monitoring of video QoE. For the scenario of QoE forecasting, we build a QoE indicator called *YouScore* that quantifies the percentage of videos in the catalog of a content provider that may play out smoothly (without interruptions) for a given out-band network QoS. For the QoE monitoring scenario, we estimate the QoE using the inband QoS features obtained from the encrypted video traffic. Overall, for both scenarios (forecasting and monitoring), we highlight the importance of using features that characterize the video content to improve the accuracy of QoE modeling.

Keywords: Quality of Service, Quality of Experience, Machine Learning, Active Learning, Controlled Experimentation, Internet Video, YouTube

Résumé

Le streaming vidéo est l'élément dominant au trafic Internet actuel. En conséquence, l'estimation de la qualité d'expérience (QoE) pour le streaming vidéo est de plus en plus importante pour les opérateurs réseau. La qualité d'expérience (QoE) de la diffusion vidéo sur Internet est directement liée aux conditions du réseau (par exemple, bande passante, délai) également appelée qualité de service (QoS). Cette relation entre QoS et QoE motive l'utilisation de l'apprentissage automatique supervisé pour établir des modèles reliant QoS à QoE. La QoS du réseau peut être mesurée activement en dehors du plan de données de l'application (outband) ou de manière passive à partir du trafic vidéo (inband). Ces deux types de qualité de service correspondent à deux scénarios d'utilisation différents: la prévision et la surveillance. Dans cette thèse, nous examinons les défis associés à la modélisation de la QoE à partir de la QoS réseau, à savoir 1) le coût élevé de la phase expérimentale, et 2) la considération de la grande diversité du contenu vidéo et du chiffrement déployé.

Premièrement, la modélisation de la QoE par expérimentation contrôlée constitue un défi, les dimensions d'espace d'expérimentations ainsi que le temps non négligeable de chaque expérience rend cette modélisation plus complexe. L'approche classique consiste à expérimenter avec des échantillons (de qualité de service), échantillonnés de manière uniforme dans tout l'espace expérimental. Cependant, un échantillonnage uniforme peut entraîner une similarité significative au niveau des labels, ce qui entraine une augmentation du coût sans gain en précision du modèle. Pour résoudre ce problème, nous recommandons d'utiliser *apprentissage actif* pour réduire le nombre d'expériences sans affecter la précision. Nous examinons le cas de la modélisation QoE sur YouTube et montrons que l'échantillonnage actif fournit un gain significatif par rapport à l'échantillonnage uniforme en termes d'augmentation de la précision de la modélisation en moins d'expériences. Nous évaluons ensuite notre approche avec des ensembles de données synthétiques et montrons que le gain dépend de la complexité de l'espace expérimental. Dans l'ensemble, nous présentons une approche générale d'échantillonnage qui peut être utilisée dans n'importe quel scénario de modélisation QoS-QoE, à condition que la fonctionnalité de QoS en entrée soit entièrement contrôlable.

Deuxièmement, prévoir la qualité de l'expérience de la vidéo avec précision s'avère difficile, d'une part les vidéos des fournisseurs de contenu actuels varient énormément, des vidéos sportives rapides aux vidéos éducatives statiques. De plus, le trafic vidéo actuel est crypté, ce qui signifie que les opérateurs de réseau ont une visibilité réduite sur le trafic vidéo, ce qui rend la surveillance de la QoE plus complexe. Face à ces défis, nous développons des modèles afin de prévoir ainsi que surveiller avec précision la qualité de l'expérience vidéo. Pour le scénario de prévision QoE, nous construisons

un indicateur QoE appelé *YouScore* qui prédit le pourcentage de vidéos pouvant être lues sans interruption en fonction de l'état du réseau sous-jacent. En ce qui concerne la surveillance QoE, nous estimons la QoE à l'aide des fonctionnalités de qualité de service inband obtenues à partir du trafic vidéo crypté. En conclusion, pour les deux scénarios (prévision et surveillance), nous soulignons l'importance d'utiliser des fonctionnalités qui caractérisent le contenu vidéo afin de pouvoir améliorer la précision des modèles.

Mots-clés: Qualité de Service, Qualité d'Expérience, Apprentissage Machine, Apprentissage Actif, Expérimentation Contrôlée, Streaming Vidéo, YouTube

Acknowledgements

Finally, after a journey of three years, I am writing this acknowledgment to recognize all those people who supported me during my Ph.D. I would like to first thank my supervisor Prof. Chadi Barakat for his continuous help, guidance, and moral support, without which, I wouldn't have been able to complete this thesis. Chadi was always there to motivate me in my periods of depression and most certainly helped me in brainstorming new ideas and formulating my research. In short, I found Chadi a thorough gentleman, both personally and professionally. I thank the reviewers for taking the time to review my thesis and thank the examiners for being part of the jury of my defense. I thank the DIANA team members for the memorable time that I spent at Inria Sophia Antipolis. In particular, I would like to thank my fellow Ph.D. students with whom I spent most of the time drinking coffee; I will most certainly miss those long coffee breaks.

I would also like to thank my family for the support they gave me throughout my Ph.D. I am going to mention each of them. First of all, I have to say that whatever I am today is because of my mother. It is her who inculcated in me the passion to achieve academic excellence and her prayers are the real source of my success. Doing a Ph.D. was a personal goal that I had set for myself at a very young age. But it was my late father who deeply wished that I do a Ph.D. Fulfilling his desire has been extremely satisfying for me and my family; had he been alive, he would have been the happiest person to see both his sons (my elder brother is also a Ph.D.) holding Ph.D. degrees. My brother also has been a great source of inspiration for me. It was him who kept me motivated to pursue a Ph.D. during my career as a Telecom engineer. I also thank my elder sisters for their consistent prayers for my success. Lastly, I thank my wife for being the most caring life partner and for her unwavering support and comfort during my stressful days and for taking care of our daughter.

Contents

List of Figures

List of Tables

Abbreviations

QoS	**Q**uality **o**f **S**ervice
QoE	**Q**uality **o**f **E**xperience
ML	**M**achine **L**earning
MOS	**M**ean **O**pinion **S**core
DT	**D**ecision **T**ree
RF	**R**andom **F**orest
RMSE	**R**oot **M**ean **S**quared **E**rror
HAS	**H**TTP **A**daptive **S**treaming
ABR	**A**daptive **B**it**R**ate
DASH	**D**ynamic **A**daptive **S**treaming over **H**TTP

Dedicated to my mother.

Chapter 1

Introduction

Video streaming is the most dominant contributor to global Internet traffic. By 2021 the global share of IP video traffic is expected to reach 82%, up from 73% in 2016 [2]. Similarly, by 2023, mobile video traffic is expected to increase from 56% in 2017 to 73% in 2023 [3]. The huge demand for Internet video pushes network operators to proactively estimate Quality of Experience (QoE) of video streaming users in their networks. The QoE of Internet video, as per prior subjective studies, is dependent on application Quality of Service (QoS) features such as initial loading time (startup delay or join time), frequency of re-buffering/stalling events, playout quality (spatial resolution) and its variations [4], [5], [6]. Network operators usually do not have such information about the video traffic generated in their networks as most of the traffic is getting encrypted. So the only possible solution for gauging the QoE of video streaming is to rely on network-level features obtained from the encrypted video traffic traces, or independent network measurement tools executed outside the video application data plane.

Prior works on video QoE estimation have shown that network-level performances (e.g., bandwidth, delay, packet loss rate) directly impact QoE [5]. This motivates the use of supervised machine learning (ML) to link the network level measurements (referred here as the network QoS) to QoE. Supervised ML based QoS-QoE modeling requires some ground truth dataset that maps the input QoS to the output QoE. Typically, such datasets are built either by controlled experimentation or by crowdsourcing. In controlled experimentation, the target application is run (on an end user device) in a controlled environment where the input QoS features are *fully controlled*, resulting in a QoS-QoE dataset that maps the enforced QoS to the observed QoE [7], [8], [9], [10]. On the other hand, in crowdsourcing the ground truth data is collected in the wild without any control over the QoS metrics, and typically by a large number of users

who are asked to rate the QoE of a given application over the Internet [11], [12], [13], [14]. Crowdsourcing suffers from a lack of control over the test conditions [15] and not everyone can have access to such data from a large set of real users, which pushes most researchers to rely on building their own datasets by controlled experimentation.

In controlled experimentation, the videos are played out in emulated network conditions, as shown in Figure 1.1, to build the dataset for training the ML algorithms to build the QoE prediction models. These models estimate video QoE by relying only on the network QoS features. Most works on network QoS-QoE modeling use QoS features that are collected from video traffic traces for *QoE monitoring*. However, in a recent work based on QoS-QoE modeling for Skype [16], the notion of *QoE forecasting* from end-user devices is presented. This consists of using network QoS measured outside the data plane of the application and without relying on the application traffic (Skype in this particular case). The benefit of this approach is that it allows the same measurements to be used to predict QoE for many different applications. In this thesis, we consider video streaming where we build prediction models (by controlled experimentation) that allow both the forecasting and the monitoring of video QoE using network QoS only. We discuss in greater detail each of the prediction scenarios and the associated network QoS measurement types next.

1. **QoE forecasting**. In QoE forecasting, the network QoS is based on measurements made by the clients towards the target measurement servers placed at different vantage points in the network. These measurements are *out-of-band* w.r.t to the video traffic, which means that no video traffic is required to measure them. The benefit of the QoE forecasting approach is that the QoE of any Internet application can be forecasted in advance, without the need to run the application itself. This method of QoE forecasting is used by mobile applications such as ACQUA [17] and Meteor [18] where QoS-QoE models (built offline) forecast the QoE of apps such as YouTube and Skype using the QoS measurements of bandwidth, delay, jitter, etc.

2. **QoE monitoring**. The QoE monitoring approach is used to monitor the QoE from within the core of the networks. This approach uses models that require network QoS to be measured directly from video traffic. These measurements are *inband* w.r.t to the video traffic and are obtained *passively* without disrupting the video traffic. Examples of such measurements can include the instantaneous throughput, the packet interarrival times and the packet sizes [5]. The QoE monitoring approach is deployed on network middleboxes that parse the network traffic to estimate the QoE of the videos played out [5], [19].

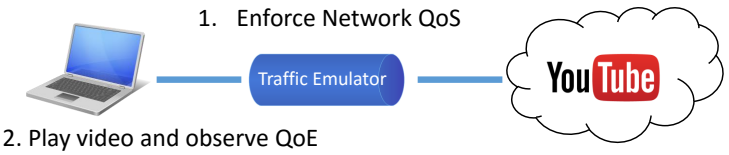


FIGURE 1.1: The controlled experimentation setup

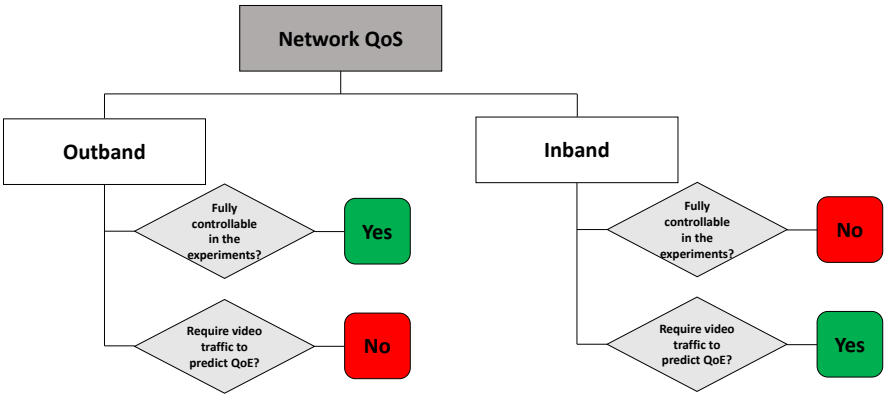


FIGURE 1.2: The network QoS

As already mentioned, the *outband* features include measurements such as bandwidth and packet loss rate. These measurements can be emulated in the lab by network emulators such as Linux traffic control (*tc*), and so are fully controllable. On the other hand, *inband* measurements such as packet sizes and their variation are hard to control as they depend on how the video application adapts to the network conditions. The controllability of the features is an important property that defines the sampling method we employ to vary the QoS in our experiments, we discuss these in the next section. Overall, a summary of the characteristics of the two types of network QoS measurements is shown in Figure 1.2, where we can see that outband features are fully controllable, whereas inband features are not fully controllable, and are based on passive measurements obtained from the video traffic.

Considering these scenarios, we tackle two challenges in this thesis that are related to 1) the large training cost associated with QoE modeling by controlled experimentation, and 2) the accurate prediction of video streaming QoE. These form the basis of the work presented in this thesis. In the next section, we individually discuss the challenges and the solutions proposed in the thesis.

1.1 Challenges tackled in this thesis

1.1.1 The large experimental space to cover in controlled experimentation

Typically in QoS-QoE modeling using ML, the training datasets are either collected in the wild or built by controlled environments. The conventional approach of building such datasets by controlled experimentation 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 are diverse and rapidly evolving, thus requiring the QoE models to be re-built regularly.

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. 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 wants to label and learn from as part of an iterative process. In active learning literature, *pool based uncertainty sampling* is the most widely used active learning strategy, where the ML model has a pool of unlabeled data and the objective is to intelligently label samples from this pool to build an accurate model quickly with fewer labeled samples.

In this thesis, we first apply the *pool based uncertainty sampling* in our scenario of QoS-QoE modeling for YouTube video streaming by controlled experimentation and show that pool based sampling provides a significant gain over uniform sampling. We then illustrate a potential issue with uncertainty sampling with very large pools and propose an active sampling approach that does not require any pool. Finally, we illustrate with synthetic datasets that the gain of active sampling depends on the complexity of the

feature space for any given experimentation scenario. To the best of our knowledge, this is a first attempt at applying active learning in the context of network QoS-QoE modeling by controlled experimentation.

1.1.2 Accurate prediction of video streaming QoE

1.1.2.1 The large diversity of contents of today's video content providers

Today's video content providers typically store a large number of videos that vary in content type ranging from fast motion sports videos to static educational lectures. Due to the difference in contents, the QoE can vary significantly from one video to the other for the same network QoS. Considering this variation in content, we propose to build global QoE indicators for video streaming that can accurately highlight the variation in the QoE of videos due to the difference in contents. Such indicators can allow network operators to measure the extent to which the QoE of video streaming users can degrade in poor network conditions considering the diversity of the contents offered by today's content providers.

Internet video is typically stored in a variety of video encoding formats including FLV, MP4, and VP9. Most commonly, the Variable BitRate (VBR) encoding techniques are used by modern video content providers where complex scenes of the video are allocated higher bitrates while less complex scenes are allocated lower bitrates. Owing to this, different videos can have different video bitrates. Fast-moving videos tend to have a higher video bitrate compared to slow-moving videos. The video bitrate, in turn, affects the network QoS required for smooth video playout.

Based on the inherent relationship between average bitrate, QoS and QoE, we build a global QoE indicator, which we call *YouScore*, that can predict the percentage of videos in the catalog of the content provider (e.g., YouTube) that may play out smoothly (without stalls) for a given network QoS. To devise *YouScore*, we first build a large catalog of 1 million YouTube videos consisting of metadata including the average video bitrate of each video. We then build a training QoS-QoE dataset by playing out a diverse set of YouTube videos (sampled from the catalog) under a wide range of network conditions where the sampling of the videos and the selection of the relevant conditions for network emulation is done using active sampling. The collected dataset is then used to train supervised ML algorithms to build a QoS-QoE model that takes as input the outband network QoS (downlink bandwidth and RTT) and the average video bitrate to predict whether the video plays out smoothly or not –by using the video bitrate, we show a 13% improvement in model accuracy highlighting the importance of using features that

characterize video content. Finally, we use this model to devise *YouScore*, which takes as input the outband network QoS only and predicts the percentage of videos (in the catalog) that may play out smoothly.

The benefit of YouScore is that it can be used by network operators to benchmark their performance w.r.t video streaming considering the large diversity of contents offered by today's content providers. We show this benefit by applying YouScore on a real dataset of network measurements to compare the performance of different mobile networks for video streaming.

1.1.2.2 The encryption of today's video traffic

Today's Internet video traffic is mostly served using end to end encryption. This means that network operators have no visibility into the video content traversing their networks and traditional methods of parsing HTTP packets to monitor QoE [20] are no longer applicable. So the only option left for network operators to monitor end user QoE is to rely on the *inband* network QoS measurements generated inside the core of their networks. Considering this scenario, we build ML models that use *inband* features obtained from the network traffic to estimate the video QoE. The characteristics of *inband* features are highly dependent on the internal dynamics of the video application generating the traffic and are not fully controllable. Therefore, we propose to use a trace based sampling approach where we vary the network QoS as it is observed in the wild. Here, we sample the space based on the measurement distributions of applications such as RTR-Netz [21] and MobiPerf [22]. Overall, we build QoE prediction models to estimate the objective QoE metrics of the startup delay, the quality switches and the resolution of playout, and the subjective MOS from the encrypted video traffic traces[1]. For the MOS, we rely on the standardized ITU-T P.1203 model that provides a MOS ranging from 1–5 taking into account the QoE metrics such as the resolution and bitrate of chunks, and, the temporal location and duration of the stalling events. To the best of our knowledge, this is the first attempt at linking network QoS to the ITU P.1203 model. Prior works [5, 23, 24] do not consider the subjective MOS, they consider objective QoE metrics only (e.g., average video quality). Finally, we compare the performance between the outband and the inband network QoS features and highlight that the inband features enriched with video chunk sizes (inferred directly from the encrypted traces) improve the accuracy of the models for all the QoE metrics. The models presented are based on a highly diverse dataset of around 100k unique video playouts from different geographical locations of France (Sophia Antipolis, Grenoble, Rennes, Nancy, Nantes).

[1] We also refer the objective metrics by the term application QoS in our work.

Chapter	Sampling method	Dataset size	Network QoS	QoE metrics
3	Uniform and active sampling	10k (uniform), 4k (active)	outband	Number and duration of stalls
4	Active sampling	2k	outband	Startup delay and stalls
5	Trace based sampling	100k	outband, inband	Startup delay, quality, stalls, quality switches, ITU-T P.1203

TABLE 1.1: Summary of the datasets collected in this thesis.

1.2 Thesis Outline

The thesis is organized as follows:

In Chapter 2, we review the methods used in literature for measuring and modeling video streaming QoE.

In Chapter 3, we present our intelligent sampling framework for QoE modeling where we validate our framework for the case of YouTube QoE modeling. The work presented in this chapter has been published in [25], [26] and [27].

In Chapter 4, we apply our active sampling methodology for modeling QoE for a large number of videos and develop the YouScore for performance benchmarking of network operators w.r.t Internet video. The work of this chapter is published in [28].

In Chapter 5, we devise and evaluate the performance of the ML models for QoE monitoring of video streaming from encrypted network traffic. This work has been published in [29].

Finally, in Chapter 6, we conclude the thesis with a discussion on the possible future research work. The datasets collected as part of this work are summarized in Table 1.1 where we highlight the sampling method (used to vary the network QoS), the number of samples, the type of the input network QoS features and the output QoE metrics used for each dataset. We make these datasets publicly available at [30] and [31].

Chapter 2

State of the Art on QoE Modeling for Internet Video Streaming

In this chapter, we describe the methods used in the literature for measuring and modeling Internet video QoE. We first discuss the technologies used in today's video streaming services in Section 2.1 followed by a discussion on the video QoE metrics in Section 2.2. Finally, in Section 2.3, we discuss the techniques used for modeling video QoE in literature.

2.1 Video streaming delivery techniques

Video streaming refers to the process where the video content is viewed by an end user while being delivered by a content provider concurrently. Typically, the videos start playing once the initial part of the video is downloaded and the remaining part continues to be fetched from the servers while the video is being viewed.

Traditionally, video streaming was based on protocols such as RTSP (Real Time Streaming Protocol) [32] and RTP (Real Time Protocol) [33] for real time delivery of video content. These protocols relied mostly on UDP to ensure minimal delay in video transmission. However, with UDP, the video playback can suffer from visual impairments, i.e., some video frames might get distorted or dropped due to packet losses [34], [35].

Modern video streaming systems of today use the HTTP protocol for delivering the video content. Using HTTP for video streaming is easy to deploy as most firewalls allow HTTP/HTTPS traffic, which means additional network configurations for handling the video traffic are not needed [36]. This ease of deployment propelled the widespread use of HTTP based video streaming [37]. HTTP runs over protocols such as TCP or QUIC,

which means that video content is reliably delivered to the client and there can be no visual distortions (frame drops) in video playout. However, it can suffer from additional delays due to the use of an application buffer at the client. We discuss this issue next.

2.1.1 Progressive download using HTTP

In HTTP based video streaming, the video file is divided into chunks of equal duration and the client downloads the chunks progressively and stores them in the application buffer. When the user requests to watch a video, the client starts to fill the buffer, once a sufficient amount of video is downloaded, the playback of the video is started. The video chunks are downloaded concurrently with the playout of the video, i.e., the buffer is filled with newly downloaded chunks, while it repletes as the downloaded chunks are played out. In this scenario, if the network conditions degrade due to issues such as packet losses or low throughput, larger delays can occur in the downloads, which can cause the buffer to fill up slowly compared to the playback time of the video. This can result in the buffer getting empty (buffer underrun) causing the playback of the video to interrupt. This interruption lasts until the next chunk has been downloaded to continue the playout. These interruptions are also referred to as stalling or rebuffering events.

2.1.2 HTTP based Adaptive video Streaming (HAS)

Prior subjective studies on video QoE have shown that stalling events in the video playout significantly degrade the video QoE [38], [39], [4]. In some cases, stalling is even considered worse than image quality [40], which means that users may prefer lower quality than having stalls in the playout. Considering these observations, the HTTP based Adaptive video Streaming (HAS) [41] was developed to allow adaptation of the quality of playout to prevent the occurrence of any stalls. In adaptive video streaming, the video chunks are stored in several quality representations (usually in different spatial resolutions) where the quality of the next chunk to download can change depending upon the state of the network. This means that in case of poor network conditions, the client can request a lower resolution for the next chunk to download, a lower resolution would require lower throughput, which means less time to download and thus, preventing any interruptions in the playout.

Adaptive video streaming is now widely deployed by most of the video content providers on the Internet [36]. Examples of different implementations include Microsoft HTTP Smooth Streaming (HSS) [42], Adobe HTTP Dynamic Streaming (HDS) [43], Apple HTTP Live Streaming (HLS) [44] and MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [41]. Among them, MPEG-DASH is the only solution that is open and

standardized [34]; the implementation includes the dash.js player managed by the DASH Industry Forum (DASH-IF) [45].

The method of choosing the best resolution to maximize QoE of a video playout for a given network condition is referred to as the ABR selection process. Devising algorithms for optimal ABR selection is an active area of research with a plethora of papers published in the literature [36]. We summarize the main algorithms in the next section.

2.1.2.1 Adaptive BitRate (ABR) selection algorithms

ABR selection algorithms can be rate based, buffer based or a hybrid of both the preceding methods. In traditional rate based methods, the ABR algorithm first estimates the network throughput and then selects the chunk whose video bitrate (among the available chunk representations) is the highest but lower than the throughput estimate. Estimating throughput in today's mobile networks is challenging and can be prone to errors. In order to avoid using any erroneous throughput estimates, buffer based approaches were developed that do not require any estimation of the throughput, rather they rely only on the buffer occupancy to choose the quality of the chunk to download [46], [47]. The third method, which is a hybrid method, considers both the throughput estimation and the buffer occupancy in the chunk selection process [48], [49], [50], and is most widely used technique in production environments [51]. Apart from these approaches, learning based methods have also been introduced that use reinforcement learning to learn ABR algorithms instead of using fixed heuristics [52], [53]. These techniques rely on trace based video streaming simulations and fundamentally learn the throughput dynamics of the trace they are trained on.

2.2 Quality of Experience metrics of video streaming

The end user QoE of video streaming is dependent on multiple factors. These factors can be categorized into the following four categories [20]:

1. *System Level* factors consider the technical aspect of the video streaming players and the ABR algorithms used. These factors include the network QoS (bandwidth, delay), the end user device types (computing power, screen size, mobile/PC, types of the browser), and the application layer QoS features (video adaptation strategies, rebufferings).

2. *Context Level* factors capture the surroundings and the context in which the user views the video content. These can include factors such as the user's location, the purpose of viewing the video (for educational purposes or entertainment), etc.

3. *User Level* considers the psychological factors of the end user such as the user expectations, the user's mood, the user's background, her/his browsing history, etc.

4. *Content Level* factors are related to the characteristics of the video content and are gauged by metrics such as the encoding rate, the encoding format, the resolution, the playback duration, the quality of the video, the popularity of the video, etc.

Most studies consider the system-level factors in the QoE modeling studies for video streaming as it is very difficult to accurately obtain other factors. In this thesis, we also consider the system-level factor where we focus on the network QoS and the application QoS features.

Fundamentally, there are two types of methods for gauging video QoE, 1) subjective, and 2) objective methods [54], [55], [56]. The subjective methods involve a set of real users who explicitly rate the QoE as good/bad or give a rating on a continuous scale, usually ranging from 1–5, where 1 is the lowest while 5 is highest perceived QoE [57]; the average of all the individual ratings represents the Mean Opinion Score (MOS), a QoE metric standardized by the ITU-T P.910 recommendation. The main drawbacks of the subjective approach are: it is costly, time consuming and lacks repeatability [58]. These limitations have motivated the development of objective methods that do not involve real users, rather they rely on using objective video metrics to estimate the QoE.

The QoE metrics for modern HTTP adaptive video streaming are summarized in Figure 2.1, which can be objective or subjective. The objective metrics include 1) the initial startup delay (join time), 2) the number and duration of stalls (rebuffering), 3) the quality of playout (usually quantified in terms of the spatial resolution of the video), and 4) the quality (resolution or bitrate) switches. The subjective metric is the Mean Opinion Score (MOS) that is the average of the ratings (usually on a scale of $1 - 5$) given by a set of real users.

Owing to their ease of measurement, objective QoE metrics are commonly used in video QoE modeling studies [5], [59], since these metrics accurately reflect the subjective QoE of real users. We discuss the individual metrics in detail below.

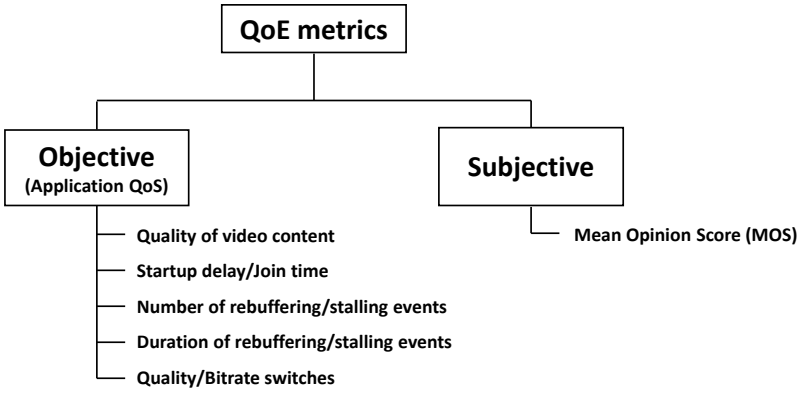


FIGURE 2.1: QoE metrics for modern Internet video streaming

2.2.1 Startup delay

The startup delay is the time taken by the client (measured from the instant the user requests for the playout) to start playing out the video. Normally, when the user requests the video playback, the client starts downloading the video chunks and begins playout when a sufficient number of chunks have been downloaded. Startup delay is an important metric for QoE and subjective studies have shown it to have an impact on user engagement as large startup delays can cause users to abandon video playout. According to a subjective study based on a dataset of 23 million views in [39], users start abandoning after 2 seconds with the abandonment rate going up to 80% after 60 seconds. This means that the startup delay has to be minimal to ensure good QoE.

2.2.2 Stalling events

Stalling events occur when the playout buffer gets empty. The impact of stalls on video QoE has been extensively studied in literature and is shown to depend on the number, the duration and the position of the stalling events in the playout. Authors in [60] find that a larger stalling duration leads to a degraded QoE. Furthermore, the authors also find that users prefer one long stalling event over frequent short ones. In [61], it is observed that the position of the stalling event also has an impact on QoE where stalls occurring at the beginning of playout have a lesser impact than stalls occurring later in the playout. Furthermore, in [62], authors show that stalls occurring at irregular intervals are worse than stalls occurring periodically. The authors of [4] find an exponential relationship between the stalling events and the MOS. Additionally, they find that users can tolerate at most one stall per video session as long as its duration is in the order of a few seconds, any additional stalls degrade the QoE severely. Overall, these studies show

that whenever possible, the videos should be played out smoothly without any stalls to ensure good QoE.

2.2.3 Video quality

Video quality is a characteristic of a video passed through a transmission system (the Internet) or a processing system (video codec) that measures the perceived degradation in the video content, generally, w.r.t the original raw video. The visual quality evaluation is performed using Visual Quality Assessment (VQA) [7] methods.

2.2.3.1 Visual Quality Assessment (VQA)

VQA methods can be subjective if real users rate the quality of the video, or objective if automatic algorithms rate the quality of the video instead of real users. The objective VQA methods can be categorized as Full Reference (FR), Reduced Reference (RR) and No Reference (NR) [7]. FR based metrics have both the original and the distorted video available, which are compared with each other to estimate how similar the two videos are frame by frame. Examples of such metrics include Peak Signal to Noise Ratio (PSNR) [63], Structural Similarity (SSIM) [64], Video Quality Metric (VQM) [63] and Video Multimethod Assessment Fusion (VMAF) [65]. In RR, only partial information of the original video is used to rate the distorted video [66], while NR methods do not have the original video and the quality assessment is done directly on the distorted video by relying on algorithms that identify different visual artifacts such as blurring, contrast variations or blockiness [67].

VQA has its relevance in evaluating the quality of a received video in real time video streaming solutions. Such streaming solutions are prone to visual distortions (the video frames getting distorted or dropped) in the playout due to packet losses as the solutions are based on UDP. However, today's Internet video is mostly based on HAS (runs on top of TCP/QUIC) and does not have any visual distortions during playout. Therefore, most QoE studies on HAS rely on using much simpler metrics for evaluating video quality. These metrics are the encoding bitrate and the spatial resolution of playout (the number of pixels used to represent the video frames). The encoding bitrate is the data required to play one second of video; it increases with the spatial resolution, which in turn generally increases video quality. Since these metrics have a direct impact on the video quality [68], these can be used to gauge the quality for HAS [7] without relying on a complex analysis of the video content. In our work, we consider the spatial video resolution as the metric of video quality.

2.2.4 Quality switches/adaptation

In adaptive video streaming, the quality of the video (spatial resolution) can change depending upon the network conditions; the client can decrease the quality in case of low network throughput and increase it when the network throughput improves. However, too many quality changes can also have a detrimental impact on QoE [69]. The amplitude of the quality switch is also important as authors in [70] find that a stepwise decrease in quality is better than one single big decrease in quality. Furthermore, in [71], it is shown that the QoE of a video played at constant low quality is better than the QoE of a video played at a higher quality with frequent quality changes. Overall, playing with constant quality is always better, however, if a quality downgrade is needed, it should be done gradually rather than abruptly.

2.2.5 MOS prediction models

Many subjective studies have shown a direct relationship between the application QoS features and the MOS. This led to the development of MOS prediction models that estimate the MOS using the application QoS. Such models suggest an exponential, logarithmic or a linear relationship between the QoS and the QoE. In [72], a general logarithmic relationship between QoS and QoE is proposed, whereas, in [73], the authors present an exponential relationship between the QoS and the QoE and refer to it as the IQX hypothesis. According to this hypothesis, the change in the QoE depends on the current level of the QoE, that is if the QoE is already very high, then even a small disturbance in QoS can significantly affect the QoE; however, if the QoE is low, further disturbances will not be perceived. This hypothesis demonstrated better approximation to the MOS compared to the earlier logarithmic relationship. A similar exponential relationship is also observed in [4] where the MOS follows an exponential relationship with the number and duration of stalls in YouTube video streaming. In another work [50], the authors propose a composite metric to gauge video QoE that is composed of a weighted linear combination of different QoE metrics including the quality, the stalls, and the quality switches.

Given the importance of video QoE estimation for network operators, the International Telecommunications Union (ITU) has developed a MOS prediction model, namely the ITU-T P.1203 model [1] for adaptive video streaming. This model estimates the MOS considering the application QoS features only and is developed based on subjective QoE ratings obtained from experiments with real users. We discuss this model in the next section.

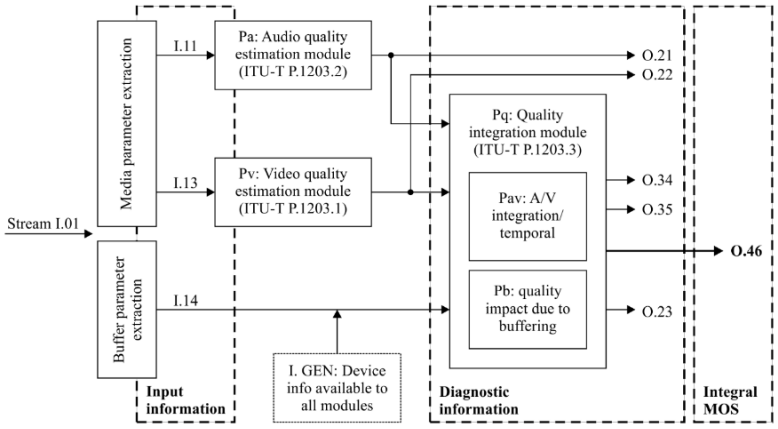


FIGURE 2.2: Building blocks of ITU-T P.1203 model. The figure is taken from [1].

Mode	Encryption	Input	Complexity
0	Encrypted media payload and media frame headers	Meta-data	Low
1	Encrypted media payload	Meta-data and frame size/type information	Low
2	No encryption	Meta-data and up-to 2% of the media stream	Medium
3	No encryption	Meta-data and any information from the video stream	Unlimited

FIGURE 2.3: Modes of operation of ITU-T P.1203 model. The figure is taken from [1].

2.2.5.1 The ITU-T P.1203 model

The ITU-T P.1203 model is a parametric model for audiovisual quality assessment of adaptive video streaming. It has three main building blocks that include the audio quality estimation module (Pa), the video quality estimation module (Pv) and the quality integration module (Pq) as shown in Figure 2.2. The quality integration module is further composed of modules for audio/video temporal integration and for estimating the impact due to stalling.

The model has four modes of operation depending upon the complexity of the input features as shown in Figure 2.3. As can be seen, mode 0 only requires the metadata on the chunk level, while for other modes, additional information on the frame level is required.

The model takes as input the following information:

1. I.GEN: Display resolution and device type (PC/TV or mobile)
2. I.11: Audio coding information.
3. I.13: Video coding information.

4. I.14: Video startup delay and stalling event information (timestamp and duration of each stall).

The I.11 and the I.13 can be on the chunk level or the frame level. Information for I.13 includes the bitrate, the frame rate, the chunk duration, the encoding resolution and the codec type for each video chunk.

The model outputs are as follows:

1. O.21: Audio coding quality per sampling interval.
2. O.22: Video coding quality per sampling interval.
3. O.23: Perceptual stall indication.
4. O.34: Audiovisual chunk coding quality per sampling interval
5. O.35: Final audiovisual coding quality score.
6. O.46: Final media session quality score. This score (ranging from 1 – 5) is the final subjective MOS for the given video session.

Model outputs O.21, O.22 and O.34 are used when frame level information of the video is provided to the model while the rest of the outputs require only the chunk level information.

We consider the ITU-T P.1203 model as the subjective MOS in our work in Chapter 5 where we try to estimate the ITU MOS (O.46) using the inband network QoS features. We use mode 0 of the model since we rely on the encrypted traffic for inferring the ITU MOS. Furthermore, the metadata needed to get the ITU MOS is obtained from the Google Chrome browser using the YouTube API [74] and the Chrome Webrequest API [75].

In the next section, we discuss the methodologies used in literature for modeling QoE of video streaming.

2.3 Methodologies for modeling video QoE

The QoE of video streaming is dependent on the application QoS metrics, which in turn are dependent on the network QoS metrics. QoE modeling studies aim at understanding and building the relationships between the application or the network QoS to the video QoE. These studies are mostly data-driven and the data required by the models is either collected in the wild or built by controlled experimentation.

2.3.1 QoE modeling using data collected in the wild

Here, the QoS-QoE models are based on data that is generated by real users using their end devices or by the collection of the measurement data observed within the networks of the service providers. A YouTube QoE model is developed in [4] that is based on data obtained through crowdsourcing. Specifically, the data is collected from 1,349 users who rate the QoE of 4,047 video streaming sessions in a Google Chrome browser using the Microworkers [76] crowdsourcing platform; the QoE metrics such as the stalling events are collected in the Chrome browser using JavaScript. The main finding of this work is that the MOS follows an exponential relationship with the number of stalls in the video playout.

In [77], an android mobile application called YoMoApp is developed to passively monitor the relevant objective metrics (i.e., player state/events, buffer, and video quality level) for YouTube QoE from end user smartphones. A field study of the dataset collected by this application is provided in [78] where the relevant application and network QoS metrics are used to characterize the performance of YouTube video streaming. Similarly, in [79], the authors develop a tool that actively crawls the YouTube website and downloads videos while collecting additional information about the network and the CDN characteristics. This tool allows the measurement of YouTube QoE from end user devices.

In another work [80], a real time QoE Detection method is presented for encrypted YouTube video streaming traffic. The dataset they build consists of 60 diverse YouTube video clips streamed over WiFi networks from three operators. Machine Learning (ML) is then used to predict objective QoE metrics such as buffer warning (low buffer, high buffer), video state (buffer increase, buffer decay, steady, stall), and video resolution from the inband network QoS features

Additional works such as [81] and [82] present large scale video QoE modeling studies from data that is collected using client-side instrumentation. The datasets considered are composed of 200 and 300 million video sessions respectively and are used to identify the reasons affecting QoE and user engagement. These works consider the objective QoE metrics including join time, stalling events, rebuffering ratio (total stalling time divided by the video session time), average video bitrate, rendering quality and video playback failure rate. The main findings of these works are related to the identification of issues that affect QoE and user engagement. Mainly three issues are identified as reasons for poor QoE, which are network throughput variations, CDN performance issues, and ISP overloading. Furthermore, the rebuffering ratio is found to have the greatest impact on user engagement.

2.3.2 QoE modeling by controlled experimentation

In controlled experimentation, the target video application is run in a controlled environment. Given the fact that large scale datasets collected in the wild are generally not made public, researchers working in the domain of QoE modeling have to rely on building their own datasets by controlled experimentation. Here the QoS metrics are accurately controlled to build the relationship between the QoS and the QoE. These studies can be subjective if real users rate the QoE in terms of the MOS or objective if only the application QoS metrics are used to gauge the QoE.

Authors in [6] create an experimental platform for HTTP streaming videos where the QoS metrics are artificially varied in the video playouts to study their effect on the corresponding MOS. The dataset used consists of experiments conducted with a set of 10 users and the QoS metrics used for experiments are predefined by uniform sampling of the experimental space.

In another work [83], 141 volunteers are asked to rate the QoE of adaptive video streaming sessions based on DASH under trace-driven network emulations. The result of their study is in line with previous studies, which states that the QoE of adaptive streaming is affected by factors such as fluency, bitrate distribution, startup bitrate level and bitrate switching frequency in the playout.

In [84], a QoS to QoE causality analysis is done with a set of sixteen test users who rate their perceived quality of YouTube videos under different emulated network conditions.

In [85], the effect of different transport protocols (TCP and UDP) on the QoE of YouTube video streaming on a bottleneck link is presented. The authors derive mapping functions that accurately describe the relationship between network-level QoS features and the QoE for both the protocols.

In another work [23], a real time QoE monitoring model is developed for estimating objective QoE metrics from encrypted video traffic of YouTube that is based on both the TCP and the QUIC transport protocols. The authors perform controlled experiments and build a dataset of 5488 and 5375 video sessions for QUIC and TCP respectively. They use Linux traffic control to emulate the network conditions with random sampling of the experimental space and use ML to predict the startup delay, the stalling events and the video resolution from inband network QoS measurements such as packet interarrival times, packet sizes and throughput.

2.3.3 QoE prediction from network QoS

The video QoE depends on the network QoS. This inherent relationship allows the development of QoE models using ML to predict video QoE using only the network QoS. Using network QoS for QoE estimation is useful as it allows estimation of video QoE from encrypted traffic as well; most of today's video traffic is encrypted and network operators do not have any visibility into the application level QoS metrics. Hence, the only possible option of network operators to estimate end user QoE is to rely on network-level measurements. These measurements can be made from end user devices or from within the networks. We refer to such measurements as *outband* and *inband* respectively (a comparison between the inband and the outband measurements is already given in Figure 1.2). We discuss each of the prediction scenarios for the respective type of QoS next.

2.3.3.1 From end user devices

Here, the network measurements consist of active packet probes sent by the client towards dedicated servers and are made outside the data plane (hence the name outband) of the video application. This approach is introduced in [16] for forecasting the QoE of Skype. Inline of this work, a mobile application called ACQUA [17] is also developed that allows prediction of QoE of apps such as Skype and YouTube (the model for YouTube is developed in this thesis) using the outband network measurements of throughput, delay, packet loss, etc. Other than ACQUA, another android application that uses outband QoS to forecast QoE is called Meteor [18]. This application uses the network QoS measurements such as TCP throughput and delay to predict the QoE for applications such as Facebook, Netflix, etc.

The QoE models presented in [6] and [86] are also based on network measurements that are outband i.e., these models are based on measurements of throughput, delay and packet loss, which can be obtained by network monitoring applications including ACQUA, Meteor, Ookla SpeedTest, RTR-NetTest, and MobiPerf.

2.3.3.2 From within the network

The conventional approach of predicting QoE from network QoS relies on using inband QoS measurements collected directly from the video traffic on network middleboxes. This approach allows network operators to monitor the QoE of their end users. In this approach, the network QoS consists of features such as throughput, packet interarrival times and packet sizes, which are extracted from the encrypted traces to estimate the

Ref.	Method	Sampling Method	Type of QoS	NW QoS	QoE metrics
[6], [86]	CE	Uniform	NW	Outband	Startup delay, number and duration of stalls, MOS
[83]	CE	Trace based	App	NA	Number and duration of stalls, video bitrate, MOS
[87]	CE	Trace based	NW	Inband	Rebuffering ratio, video bitrate
[85]	CE	Uniform	NW and App	Outband	Number and duration of stalls, video bitrate, MOS
[23]	CE	Uniform	NW	Inband	Startup delay, number of stalls, video resolution
[24]	CE	Uniform	NW	Inband	Startup delay, number of stalls, video resolution, quality switches
[80]	W	NA	NW	Inband	buffer state, video state, and video resolution
[81], [82]	W	NA	App	NA	Startup delay, stalling frequency, rebuffering ratio, video bitrate/quality and video abandonment rate
[77], [79]	W	NA	NW and App	Inband	Buffer state, number and duration of stalls, buffer duration, MOS

TABLE 2.1: Summary of methodologies used in literature for QoE modeling of video streaming. CE: Controlled Experimentation. W: Data collected in the Wild. NW: Network. App: Application level. NA: Not Applicable.

video QoE. In [5], ML models based on random forests are trained to classify a video playout into three categories considering the stalling events ("no stall", "0-1 stalls", or "2 or more stalls") and video resolution (low definition, standard definition, high definition). The data is collected from a web proxy of a mobile operator and consists of 390k video playouts in the wild. A similar approach is taken in [24] for building the models for QoE monitoring with three objective QoE classes ("low", "medium" or "high"). These approaches predict the QoE for the video session in its entirety. Other works such as [23] target real-time QoE monitoring. The authors present ML models that use the network and the transport level features of the encrypted YouTube video traffic (both TCP and QUIC based) to infer the stalls and the resolution of playout in

windows of 10 second duration; the approach is similar to the preceding works, the only difference is the shorter prediction window (10 seconds instead of the entire playout). Apart from ML, in [87], the authors devise a heuristic that analyses the network and transport level features of the encrypted video traffic to infer the video bitrate and the rebuffering ratio based on a study of around 2k video playouts from two video service providers.

Overall, a summary of the QoE modeling methodologies is given in Table 2.1 where we highlight the methodology used, the measurement source, the type of network features and the QoE metrics.

In this thesis, we devise methodologies and models using controlled experimentation and ML for forecasting and monitoring QoE of video streaming using the outband and the inband network QoS features respectively. For QoE modeling with outband QoS, we use active learning to sample the experimental space to reduce the training cost, while we use trace based sampling for the QoE models based on inband QoS. Furthermore, we consider a large set of videos in our QoE modeling scenarios where we build a catalog of 1 million YouTube videos (by crawling the YouTube website offline using the YouTube Data API) that is then used to build the *YouScore* in Chapter 4 and the QoE monitoring models in Chapter 5.

Overall, to the best of our knowledge, we make the following novel contributions:

1. We are the first to apply active learning in the context of network QoS-QoE modeling by controlled experimentation; prior works either use uniform sampling or trace based sampling as shown in Table 2.1.
2. The *YouScore* developed in Chapter 4 takes as input only the network QoS to predict QoE in terms of a novel indicator that considers the diversity of contents offered by a content provider such as YouTube. Prior works do not quantify this variability in QoE due to the difference in contents.
3. We develop an unsupervised ML based method to infer chunk sizes directly from the encrypted traffic traces and use them as features in the ML models to show significant improvement in ML modeling accuracy (Chapter 5). Furthermore, we are the first to build ML models that link the network QoS measurements to the ITU-T P.1203 MOS.

In the next chapter, we discuss our active sampling methodology for building the QoE forecasting models for YouTube.

Chapter 3

An Intelligent Sampling Framework for Controlled Experimentation and QoE Modeling

For accurate QoE modeling, the controlled experimentation 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 set up the experiments themselves. However, most often, the space of network features in which the 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 chapter, 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 outband network performance measurements and perform a rigorous analysis of our approach to quantify the gain of active sampling over uniform sampling.

3.1 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 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).

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 [59],[88],[5],[89]. 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 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 potentially wide range (e.g., from a few bits per second to gigabits per second for the bandwidth feature). Also, and more importantly, the complexity of the experimental space increases by the power of the number of QoS features. As each experiment requires a non-negligible time to be executed (e.g., the 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 regularly. 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.

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,

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

we mean how many samples in a small region of the feature space have the 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 chapter, we propose a sampling methodology 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 the 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 them is high [90]. 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 the 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 [90].

In this chapter, we apply the approach mentioned above for QoE modeling by controlled experimentation where we present the 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 explored. This phenomenon is referred to as *hasty generalization* in literature [91] where the learner tends to quickly converge to a final model based on an underlying *inaccurate* decision boundary. To avoid this problem, we then 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 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 work as it is the most widely used video streaming service and it provides an API that can be used to capture the video QoE metrics such as join time, stalls, 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 this chapter is to present an intelligent sampling methodology for reducing the training cost of QoS-QoE modeling in a controlled experimentation scenario. In summary, the contributions of this chapter 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 sampling over uniform sampling is dependent on the complexity of the experimental space considered for a given experimentation scenario.

The rest of the chapter 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 Section 3.3 and our online sampling methodology in Section 3.4. In Section 3.5, 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 3.6 and the chapter is concluded in Section 3.7.

3.2 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 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 with this instance, which requires experimentation of video streaming to be performed. In such scenarios, building training data by labeling each unlabeled instance can become a tedious task that can consume a significant amount of resources just to build the 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. 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 [90].

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 synthesis, the queries to label new features are synthesized ex novo [92], 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 3.2.1). 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, a 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 [90]. 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.

3.2.1 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 uncertainty sampling in this chapter and leave the study of the other strategies for future research; detail on the rest of the strategies can be found in [90].

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, the uncertainty of a machine learning model is higher for instances near the decision boundaries compared to 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 a 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 that needs to be 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 strategies mentioned above.

3.3 Pool based sampling for QoE modeling

Given a large pool of unlabeled network QoS instances, we first select an instance from the pool that 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.

3.3.1 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 an 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 if the model being built converges and stops improving further. In active learning literature, the criterion to stop further experiments is referred to as the *stopping criterion*, we will discuss it later in Section 3.4.1.

For evaluating our approach, we set the budget equal to the pool size and experiment until the pool is exhausted. 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 $\mathcal{T}$
7: **for** $i = 1, 2, ...\ N$ **do**
8: $\quad\Theta = \textbf{train}(\mathcal{T})$
9: $\quad$select $x^* \in \mathcal{P}$, as per Φ
10: $\quad$experiment using x^* to obtain label y^*
11: $\quad$add $\langle x^*, y^*\rangle$ to $\mathcal{T}$
12: $\quad$remove x^* from $\mathcal{P}$
13: **end for**

3.3.2 Implementation

To implement machine learning we use the Python SciKit-Learn library [93]. 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. 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 [94]. 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 that learners such as Neural Networks do not provide. So, due to the aforementioned reasons, we use Decision Trees 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 producing probabilities 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 3.3.5).

3.3.3 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 misclassifications but also for the range of error.

3.3.4 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 for active sampling, we perform a greater analysis on YouTube QoE in Chapters 4 and 5. Our overall data collection was based on a sample 2 min YouTube 720p video (video ID: Ue4PCI0NamI) with a timeout of 5 mins for each experiment. The dataset is 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 [95]. 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 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 objective QoE score (to be explained later, but for now we use 0 or 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.

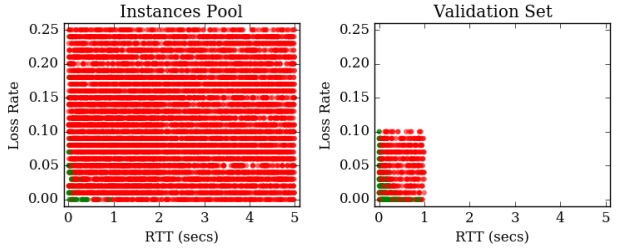
Within the experimental space $\mathcal{P}$, we first generate a uniformly distributed pool of around 10,000 unlabeled network QoS instances. We obtain the corresponding labels using the above-mentioned procedure of controlled experimentation. A similar procedure is adopted to generate 800 samples within 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.

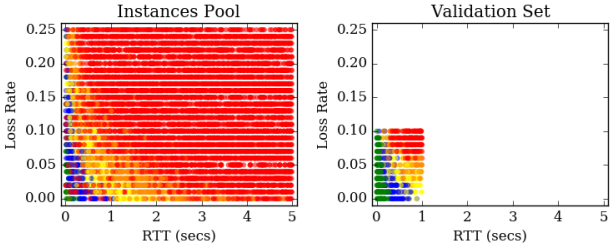
3.3.5 Mapping function

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 [96]. The QoE labels quantifying the user experience are directly related to the application level metrics of overall stalling duration and the initial join time [4]. The larger these durations, 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 [4]. 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

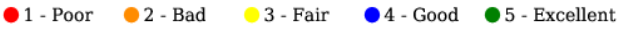


(A) Binary classification



(B) Multiclass classification

(C) Binary QoE labels

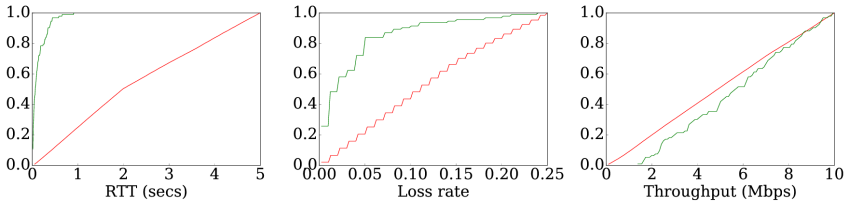


(D) Multiclass QoE labels

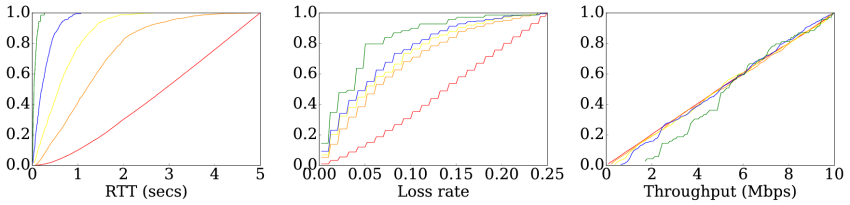
FIGURE 3.1: Visual representation of the datasets

scenario, the QoE is a 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) [39] which illustrates the video abandonment rate going up to 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 or its given parameter values do not represent 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 3.1 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



(A) Binary classification



(B) Multiclass classification

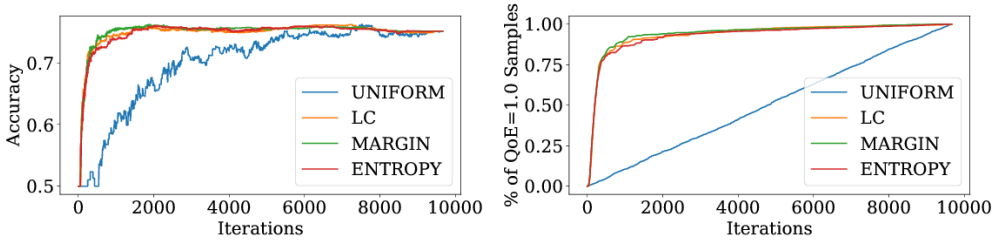
FIGURE 3.2: CDF of the samples in the pool

QoE are clustered in the lower regions of the RTT and packet loss space, with apparent monotonicity in the variation of the QoE labels, in particular for the multiclass set.

Figure 3.2 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 bitrate 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 of how the network should be provisioned for the given YouTube 720p video to play out without any buffering event.

3.3.6 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 the 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 training set, $\mathcal{T}$. The entire algorithm is run for all the samples in the pool



(A) Learner accuracy (B) Cumulative ratio of *Good* samples

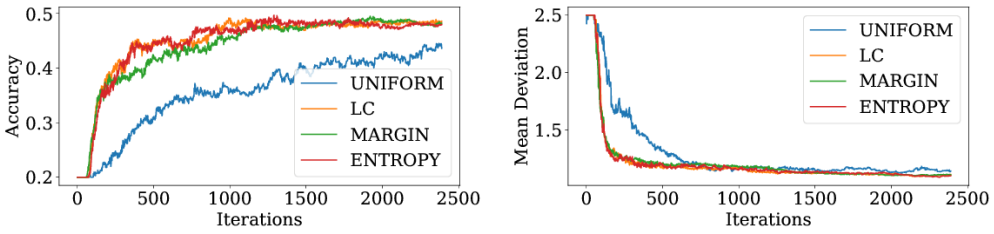
FIGURE 3.3: Binary classification results

and the performance results shown are the average of 20 independent runs. For ease of understanding, the naming convention for the sampling scenarios is 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 3.3a. 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 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 [97], [5] and [24] 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 (*inband* features), whereas the model



(A) Learner accuracy (B) Mean deviation

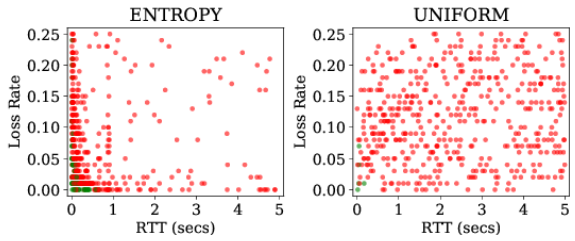
FIGURE 3.4: Multiclass classification results

presented here 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 Chapter 5, where we will show that *inband* features indeed give better accuracy compared with *outband* network features for a variety of video streaming QoE metrics.

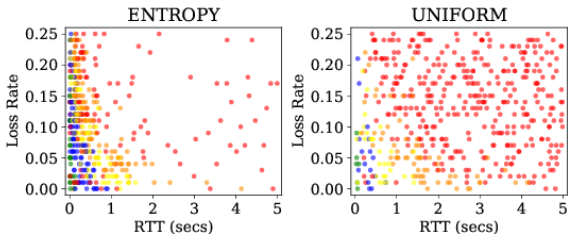
From Figure 3.1a, we can see that the underlying distribution of the binary labels in the pool is highly unbalanced with all samples from the minority class (i.e., *Good* class in our case) located in a single small region in the corner of the experimental space. Due to this reason, it can be expected 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 3.3b, 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 an adequate number of samples from each class to get a better accuracy.

The results for the multiclass classification case are shown in Figure 3.4a 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 3.4b is close to 1, which means that most of the misclassifications 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 3.5, at 5% of the pool size. It is evident that in the case of uniform sampling, the training set comprises of points scattered throughout



(A) Binary classification



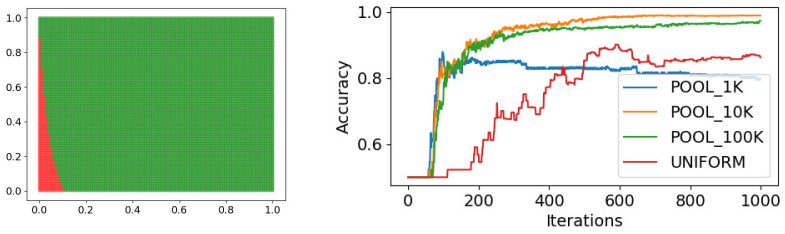
(B) Multiclass classification

FIGURE 3.5: Training set samples at 5% of the pool size

the sample space whereas, for 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 the uniform sampling of the space. Hence, we can 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.

3.3.7 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 a 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 3.3.1) 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 3.1. The function we use to represent such a boundary is given by $y = 0.1 + (1 - x)^{20}$ (Figure 3.6a) assuming that the features take values of 0 or 1. Furthermore, to mimic real systems, we add



(A) Ground Truth (B) Accuracy

FIGURE 3.6: Comparison of accuracy with different pool sizes

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 3.6b, which shows 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.

3.4 Relaxing the pool assumption and solving the blocking issue: an online sampling approach

Our approach to overcome the problem mentioned above regarding the pool size selection and sampler blocking can be summarized in Figure 3.7. Instead of selecting samples from a pool of scenarios, the experimental space itself is directly sampled to obtain a point [3]

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

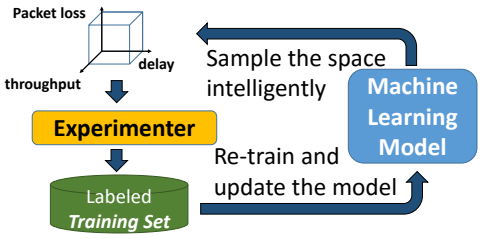


FIGURE 3.7: Active sampling

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 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 (DT) here. A DT learner splits the space using a set of internal nodes and edge leafs arranged in a tree structure and learned 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

$$e_i = -\sum_{m=1}^{M} p_i^{(m)} \log p_i^{(m)}, \tag{3.1}$$

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 the same certainty about all points in a leaf, the experimentation scenario (e.g., a tuple of the 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 (3.1).

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
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**

3.4.1 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 construction can be gauged by several measures including model accuracy, uncertainty and model confidence [98]. 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 the change in model performance over time. Instead, we can demonstrate model convergence by observing the change in the uncertainty (Equation 3.1) and the confidence measures (Section 3.4.2) over the course of the iterations. As we will show in Section 3.4.5, the uncertainty of the model decreases while the confidence increases with the 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.

3.4.2 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 $c_i = \max\langle p_i^{(1)}, \dots p_i^{(M)}\rangle$, 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 with 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

$$w_i = \frac{1}{X_1 X_2 ... X_N} \int \cdots \int_{x_n^{LOW_i}}^{x_n^{HIGH_i}} dx_1 \dots dx_N, \tag{3.2}$$

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.

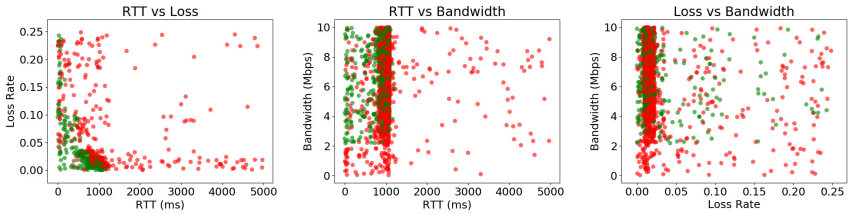
3.4.3 Evaluation

We implement our HYBRID sampling methodology shown in Figure 3.7, in a controlled experimentation framework for modeling YouTube video QoE based on the same methodology of Section 3.3.4. 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

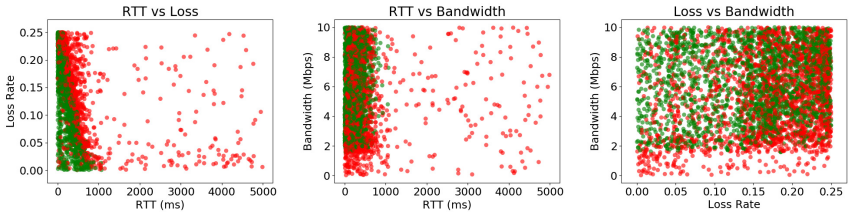
which it enforces on its network interface using Linux traffic control `tc`, 2) runs a sample YouTube 720p video video ID: oFkulzWMotY and obtains its corresponding QoE, 3) sends back the labeled tuple of the enforced QoS and the output QoE to the controller, which then updates the DT model locally. For the QoE, we consider the binary case to ease the illustration of results, i.e., *Bad* if the video suffers from stalling 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. 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 throughout the iterations showcasing that further experiments can be stopped once stability is achieved in the model under construction.

The visual representation of the collected datasets is shown in Figures 3.8a and 3.8b. In these figures, sampled network scenarios are projected over different pairs of QoS axes. We can 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 3.9a 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.

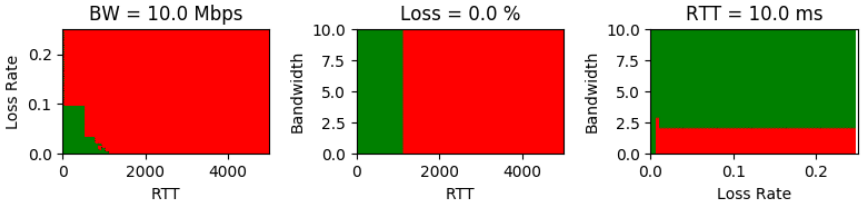


(A) MAXENTROPY

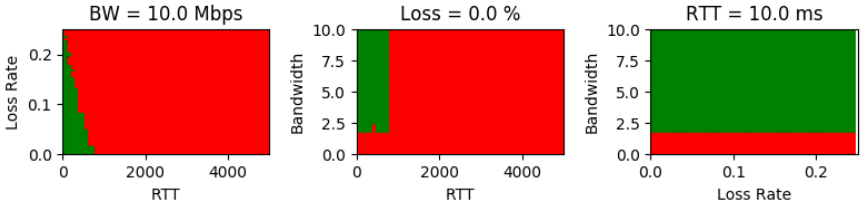


(B) HYBRID

FIGURE 3.8: Visual representation of the collected datasets



(A) MAXENTROPY



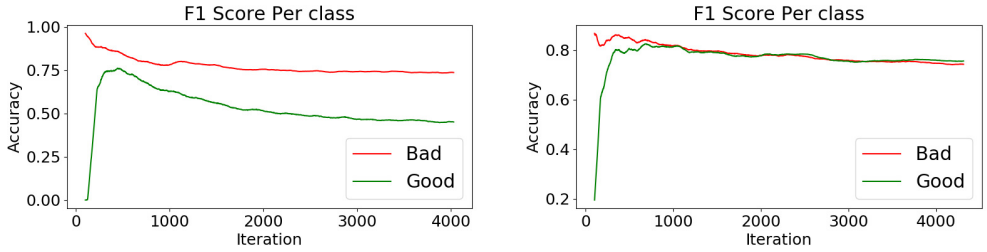
(B) HYBRID

FIGURE 3.9: Visual depiction of the DT ML model

3.4.4 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 the technique of *repeated cross validation* to gauge the performance of the models. In k-times repeated 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[4] based on cross-validation

[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



(A) MAXENTROPY (B) HYBRID

FIGURE 3.10: Comparison of the F1-scores per class b/w HYBRID and MAXENTROPY

($k = 3$ with a data split ratio of 80:20 for training and validation) that is computed at each iteration of our experimentation in Figure 3.10, 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 noisier and difficult to predict.

We further calculate the *Weighted Confidence Measure* –discussed in Section 3.4.1– to better gauge the quality of the models. Figure 3.11 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 3.3. 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 HYBRID can build QoE models having 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 3.1 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 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 3.8). In fact, in our case, the performance of the model w.r.t the minor class (*Good*) 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.

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 3.10 to 3.13 are based on moving average with a window size of 100 iterations to smooth out the curves for better presentation.

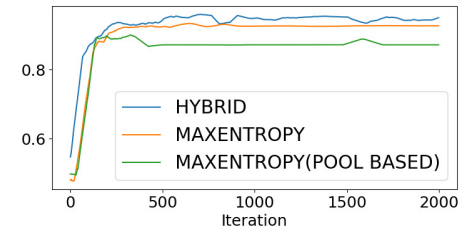


FIGURE 3.11: The weighted confidence measure

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

TABLE 3.1: Model confidence per class after 3000 iterations

3.4.5 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 3.12. 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 3.12 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 3.12 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.

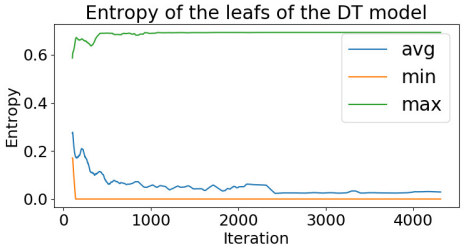


FIGURE 3.12: Variation of entropy

We now study convergence from another perspective. Figure 3.13 shows the model's minimum and maximum classification probabilities per class over all the leafs labeled

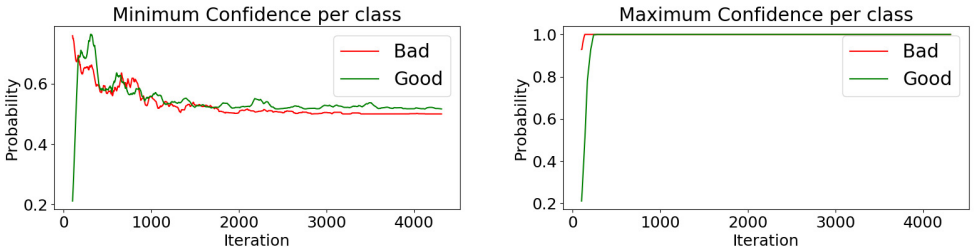


FIGURE 3.13: Classification probabilites (confidence) of the DT leafs per class

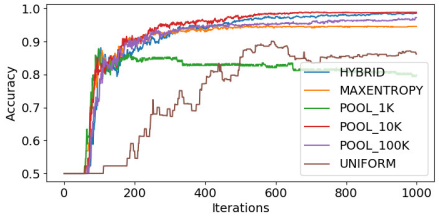


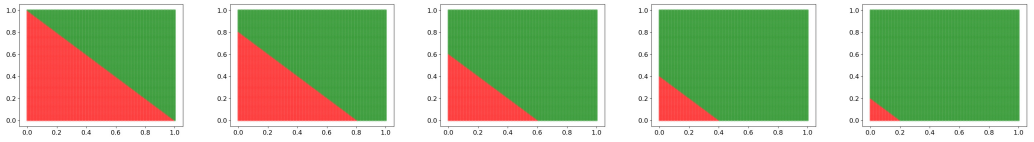
FIGURE 3.14: Comparison of accuracy for synthetic dataset (non-linear function)

with that class. The results here are complementary to the ones in Figure 3.12 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 3.11. Experiments beyond this point have almost no impact on the model performance.

3.4.6 Quantifying the gain of active sampling with synthetic datasets

We further validate the HYBRID approach over the synthetic non-linear function described in Section 3.3.7 to obtain the performance curves of Figure 3.14. 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.



(A) C=0 (B) C=0.2 (C) C=0.4 (D) C=0.6 (E) C=0.8

FIGURE 3.15: 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 3.2: Covered area for each class according to a tunable linear function

From Figure 3.14, it is also evident that active sampling techniques clearly outperform uniform sampling. However, the overall gain of active sampling 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. To verify this claim, we analyze the performance of our techniques over different synthetic feature spaces (the methodology of Section 3.3.7 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 3.15 while the area covered by each class is given in Table 3.2. 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 3.16 where we can see that the gain increases as the distribution of the labels gets more and more unbalanced in space. Furthermore, for an 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 dependent on the complexity of the feature space and the available experimentation budget.

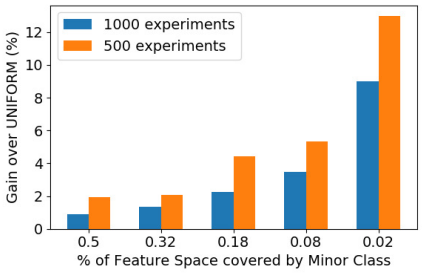


FIGURE 3.16: Gain of HYBRID over uniform sampling for different feature spaces

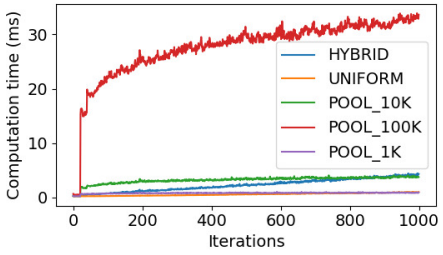


FIGURE 3.17: Computation complexity in milliseconds of each sampling iteration. Results obtained on a PC using Intel Core i7-6600U (2.60 GHz) CPU with 15 GB of RAM and using Python 2.7 on Fedora release 24.

3.4.6.1 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 3.15b) in Figure 3.17. 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 a 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.

3.5 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.

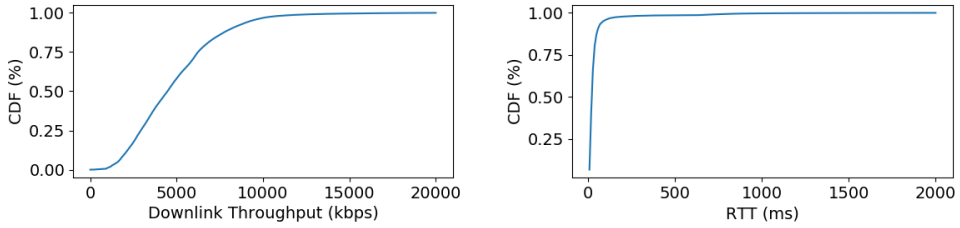
3.5.1 Application of pool-based sampling on an open network measurement dataset

To illustrate the performance of active sampling in a scenario with real networks, we apply pool based sampling on a real network measurement dataset used as our pool of data. Specifically, we use the dataset provided by FCC [99] 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 3.18 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 3.3.1 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 3.9b with loss rate set to zero. The results are shown in Figure 3.19 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 because, in this particular scenario, the decision boundary is a horizontal or a vertical line (Figure 3.9b 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 3.9b (loss rate vs. RTT).



(A) Downlink Throughput (B) RTT

FIGURE 3.18: CDF of the FCC dataset

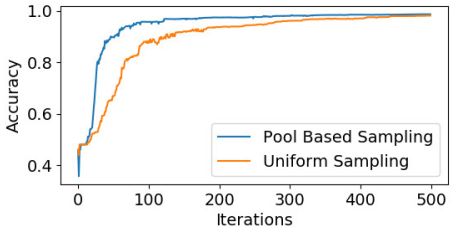


FIGURE 3.19: Application of pool based active sampling over FCC dataset. The measure of accuracy is the average of the F1-scores per class and the scores at each iteration correspond to the average of 20 independent runs.

3.6 Related work and discussion

A detailed survey of the active learning techniques can be found in [90]. To the best of our knowledge, the work presented in this chapter 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 [100] 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 [101] 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 [102] propose a technique to maximize the information gain while minimizing the duration of the benchmarking/experimentation process. Their technique gives higher importance to placing experiments in regions where larger changes in the response variable exist. The regions have a 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 methodology relies on using a probabilistic variable uncertainty criterion to pick the most suitable region for experimentation. A similar criterion is used in [103] 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 from 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 [104], 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 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 the literature for QoS-QoE modeling; a detailed survey of such models is provided in [58]. 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 [105], [5] or built by controlled experimentation [106]. For data collected in the wild, the model suffers from the bias of the underlying data source, while for the case of controlled experimentation, the similarity in the experimental space is not fully exploited. The work in this chapter 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 chapter 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 modeling for stored video content.

3.7 Summary

In this chapter, 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 with one video and our results showed improvement over the conventional uniform sampling of the experimental space. In the next chapter, we extend our sampling approach in building a QoE model for YouTube considering a large number of videos where we apply our sampling approach over a catalog of videos; we use the average video bitrate feature of each video to sample it from the catalog using the HYBRID active sampling approach.

Chapter 4

YouScore: A QoE indicator for Performance Benchmarking of Mobile Networks for Internet Video Streaming

Modern video streaming systems, e.g. YouTube, have huge catalogs of billions of different videos that vary significantly in the content type. Owing to this difference, the QoE of different videos as perceived by end users can vary for the same network Quality of Service (QoS). In this chapter, we present a methodology for benchmarking performance of mobile operators w.r.t Internet video that considers this variation in QoE. We take a data-driven approach to build a predictive model using supervised machine learning (ML) that takes into account a wide range of videos and network conditions. To that end, we first build and analyze a large catalog of YouTube videos. We then demonstrate a framework of controlled experimentation where we apply our active sampling approach (described in the previous chapter) to build the training data for the targeted ML model. Using this model, we then devise *YouScore*, an estimate of the percentage of YouTube videos that may play out smoothly under a given network condition. Finally, to demonstrate the benchmarking utility of *YouScore*, we apply it on an open dataset of real user mobile network measurements to compare the performance of mobile operators for video streaming.

4.1 Introduction

Today's content providers typically have billions of videos that vary significantly in content from fast motion sports videos to static video lectures. Building a model that represents all such contents is a challenge. Prior works on modeling video QoE either take a very small subset of videos [107], [108], or use datasets generated inside the core networks without elaborating on the kind of videos played out [5]. Such works miss out to highlight the variation in the QoE of videos due to the *difference* in contents and the span of network QoS. In this chapter, we propose to quantify this QoE variation and answer the following questions: for a catalog of given video content provider, how much would the QoE of videos of a given resolution (manually set by the end user) differ under the same network conditions? Differently speaking, what percentage of videos of a certain resolution play out smoothly under a given network QoS? Such questions are pertinent in today's 3G/4G mobile networks where capacity or coverage related issues can lower the bandwidth and inflate the delay and the loss rate of packets. Hence, it is important for operators to accurately gauge the extent to which the QoE of video streaming portals, e.g. YouTube, can degrade in congested/bad coverage scenarios considering the diversity and popularity in the contents offered by today's Internet.

Internet videos can vary significantly from high motion sports/music videos to static videos with very little motion. This difference in contents may be quantified by using some complex image/video processing techniques, however, in our work, we resort to using a much simpler metric of *average video encoding bitrate* to quantify the difference. It is calculated by dividing the total video size (in bytes or bits) by the duration of the video (in seconds). Since videos are compressed using video codecs such as H.264, which try to minimize the bitrate while not impacting the visual quality, the type of content affects the encoded bitrate of the video. For example, a high motion video is supposed to have a higher bitrate compared to a slow motion video for the same resolution. The video bitrate, in turn, affects the network QoS required for smooth playout. For example, for a high bitrate video, more bandwidth is required to ensure acceptable smooth playout compared to videos of lower bitrate for the same resolution.

Based on this basic relationship between video bitrate, QoS and QoE, we propose a methodology to devise global QoE indicators for Internet video content provider systems that would give estimates into the percentage of videos (in the targeted catalog) that may play out smoothly under a given network QoS. This QoE indicator which we call *YouScore* allows gauging network performance considering the intrinsic variability in the contents of the catalog of the target content provider. We consider YouTube in our work as it is the most widely used video streaming service of today's Internet. To devise *YouScore*, we first collect a large catalog of YouTube videos that contains video

bitrate and other meta-data information about the videos. We then build a training QoS-QoE dataset by playing out a diverse set of YouTube videos (sampled from the catalog) under a wide range of network conditions; the sampling of the videos and the selection of the relevant conditions for network emulation is done using *active learning*, an efficient sampling methodology that we developed in Chapter 3 for the YouTube experimentation case with network QoS only; in this chapter, we extend it to sample the content space (video bitrate) as well. The collected dataset is used to train supervised ML algorithms to build the predictive model that takes as input the video bitrate and network QoS to estimate the QoE. Using this ML model, we devise *YouScore* which quantifies the variation in video QoE for a case of YouTube. Finally, we demonstrate a performance analysis for different mobile operators by applying *YouScore* on a dataset of real user measurements obtained in the wild. Overall, we present a methodology for benchmarking mobile networks' performance w.r.t to Internet video streaming. Our approach is general and can be used to devise global QoE scores for any video content provider.

The rest of the chapter is organized as follows: in Section 4.2, we discuss our methodology for building the YouTube video catalog and present its statistical analysis. In Section 4.3, we discuss our framework for building the QoS-QoE dataset. We devise *YouScore* in Section 4.4 and apply it on the dataset of real network access measurements and demonstrate a comparative analysis between mobile operators in Section 4.5. The related work is discussed in Section 4.6 and the chapter is concluded in Section 4.8.

4.2 The video catalog

4.2.1 Methodology

To get a large and representative corpus of YouTube video catalog, we use the YouTube Data API and search YouTube with specific keywords obtained from Google Top Trends website. Specifically, we use around 4k keywords extracted from the top charts for each month from early 2015 to November 2017. We had to increase the number of keywords as the YouTube API restricts the number of videos returned for each keyword dynamically in the range of few hundreds. The search criterion was set to fetch short high definition (HD) videos (less than 4 minutes) for the region of USA. To obtain the bitrate information, we rely on Google's *get_video_info* API that returns the video meta-data for each video identifier. Overall and after some processing, we build a dataset of around 12 million entries for 1.2 Million unique video identifiers. Each entry in this dataset represents a unique video identified by a video ID and its available resolution

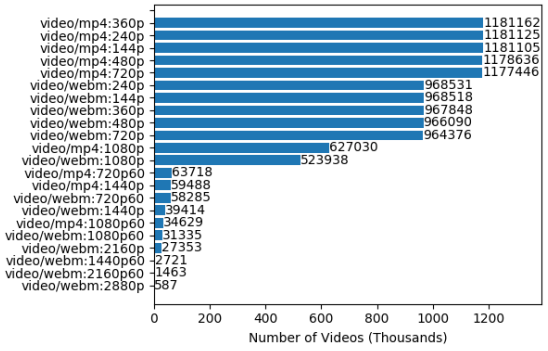


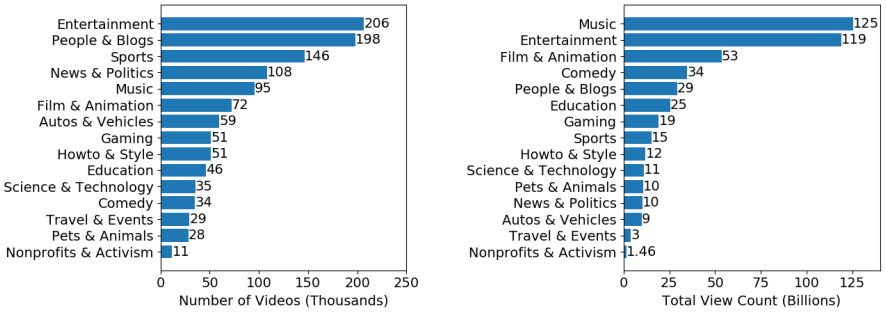
FIGURE 4.1: Video count per resolution

(identified by an *itag* value). Additional fields include the bitrate, codec, duration, category, and topic for each video.

From our analysis of the dataset, we observe that the YouTube content is available mostly in two major video types encoded by the H.264 codec and Google's VP9 codec [109]. The individual codec type of a given video encoding can be identified by the MIME (Multipurpose Internet Mail Extensions) type. For H.264 it is "video/mp4" and for VP9 it is "video/webm". In our dataset, 82% of the videos are encoded in both types with only 18% videos in only H.264 format. The overall distribution of the videos w.r.t the supported resolutions and MIME types is given in Figure 4.1. We can see that a significant portion of the obtained video IDs supports resolutions up to 1080p. We limit ourselves to this maximum resolution in our study. We make our catalog available at [30].

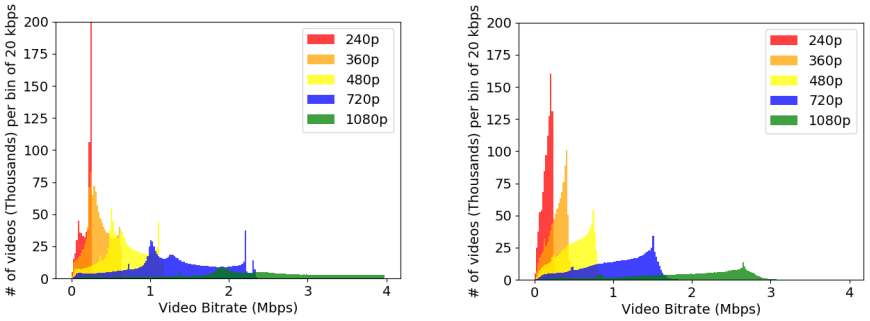
4.2.2 Video categories and popularity

Each video is assigned a *category* and a *topic* by the YouTube Data API. Based on our observation, a video can have only one category but it can have several topics. The distribution of the number of videos for each category along with the cumulative view count per category is given in Figure 4.2. The trend shows that the "Entertainment" category has the highest number of videos whereas the "Music" category has the highest number of views. Note that the "Music" category has fewer videos compared to "People & Blogs", "Sports" and "News & Politics" but has a higher view count. This indeed shows that YouTube is primarily an entertainment portal. A similar trend is observed for videos per topic (not shown here) where the highest viewed topic is "Music" while "Lifestyle" has the highest number of videos.



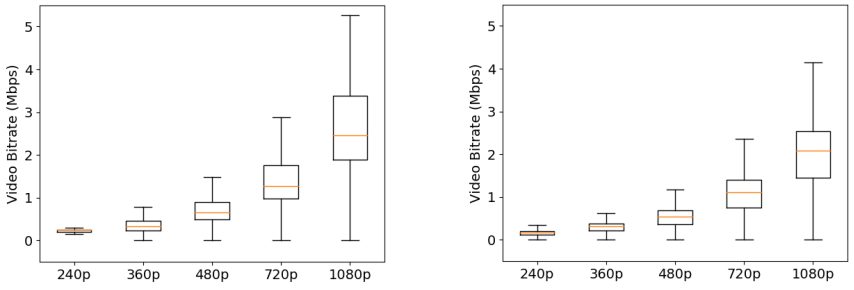
(A) Video count (B) Cumulative view counts

FIGURE 4.2: Distribution of videos per category



(A) mp4 (B) webm

FIGURE 4.3: Histogram of the bitrate of the YouTube videos



(A) mp4 (B) webm

FIGURE 4.4: Boxplot of the bitrates of the YouTube videos

4.2.3 Video bitrates

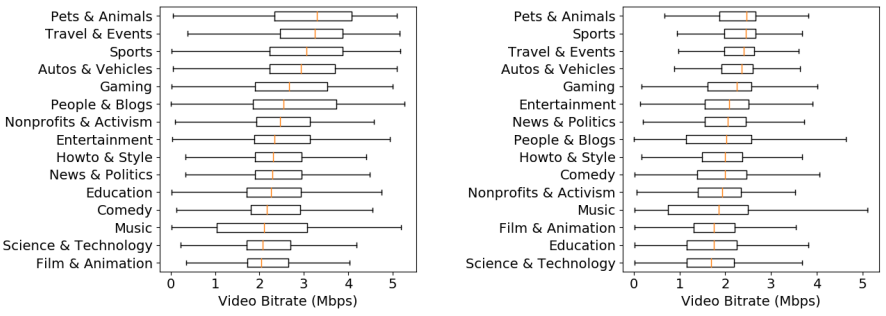
The overall distribution of the video bitrates is shown in Figure 4.3 for the two MIME types supported by YouTube (*mp4* and *webm*). We can notice how the spread of the distribution increases for higher resolution, which means that for HD videos, the bitrates can significantly vary between videos of the same resolution. This spread is also highlighted in the boxplot of Figure 4.4. An important observation is that the videos

encoded in *webm* format tend to have lower bitrate compared to *mp4*. The overall arithmetic mean of the bitrates for each resolution is shown in Table 4.1. We can notice that *webm* features on average a lower bitrate than *mp4* for all resolutions.

Resolution	mp4	webm
240p	0.215	0.154
360p	0.332	0.281
480p	0.648	0.488
720p	1.275	0.996
1080p	2.369	1.801

TABLE 4.1: Mean video bitrate (Mbps) per resolution

From this spread of the bitrate, we can infer that different contents can have different video bitrates for the same resolution. For example, fast motion content, intuitively, will have a higher bitrate compared to a slow motion content. To understand this span further, we compare the bitrates of the videos for each category as assigned by the YouTube data API. The spread of the bitrates for each category is shown in Figure 4.5 for both MIME types and for resolution *1080p*. Indeed, we can see a variation of bitrates not only in each category but across all categories. Consider for example Figure 4.5a, the median bitrate for "Film & Adaptation" is around 2 Mbps whereas it is around 3.3 Mbps for "Pets & Animals". This clearly indicates the relationship between content type and video bitrate. For *webm*, the variation across bitrates is lower but still evident; a median bitrate of 2.45 Mbps for "Pets & Animals" and 1.74 Mbps for "Film & Adaptation" can be seen.



(A) mp4 (B) webm

FIGURE 4.5: Boxplot of video bitrates (1080p) per category

4.3 The interplay between content, network QoS and QoE

The span of the bitrate across videos, even those of the same resolution, is an indication that a different QoE can be expected for the same network conditions based on the

content that is streamed. In order to study the interplay between the content itself, the network QoS and the QoE, we follow a data-driven approach supported by controlled experimentation and Machine Learning (ML). This approach consists of playing out videos of different bitrates under different network conditions to build a QoS-QoE dataset. This QoS-QoE dataset is then used to train supervised ML algorithms to produce a predictive QoE model able to capture the QoE of a video knowing its bitrate and the underlying network QoS. This model is the basis of our QoE indicator, *YouScore* to be presented in the next section.

4.3.1 Building the QoS-QoE training set for ML

We build a QoS-QoE dataset for training supervised ML classification algorithms by playing out different videos that are sampled from our catalog (Section 4.2). Videos are streamed from YouTube under different network conditions emulated locally using the Linux traffic control utility (*tc*) [110]. In this controlled experimentation approach, each experiment consists of enforcing the network QoS using *tc*, playing out a selected video under the enforced QoS and then observing the QoE (e.g. *acceptable/unacceptable*) of the playout. The videos are selected according to the *video bitrate* –to differentiate between video contents– while the network is controlled by varying the *downlink bandwidth* and the *Round-Trip Time (RTT)* using *tc*. Thus, the resulting dataset is a mapping of the tuple of three input features of *downlink bandwidth, RTT*, and *video bitrate*, to an output label consisting of the application-level QoE (*acceptable/unacceptable*).

4.3.1.1 Choice of network QoS features

The network QoS features of only *downlink bandwidth* and *RTT* are used because of the asymmetric nature of YouTube traffic where the download part is dominant. Other complex features can be taken into consideration e.g. loss rate, jitter but we limit our modeling with only two of the most relevant features for YouTube keeping in mind the unavailability of more complex features in large scale real user network measurement datasets, e.g. a popular crowd-sourced app *SpeedTest* provides measurements of only throughput and latency.

4.3.1.2 QoE definition

The QoE labels are defined as either *acceptable* if the video loads in less than 10 seconds and plays out smoothly or *unacceptable* if the loading time is more than 10 seconds or the playout suffers from stalls. Although more complex definitions of QoE are possible [1],

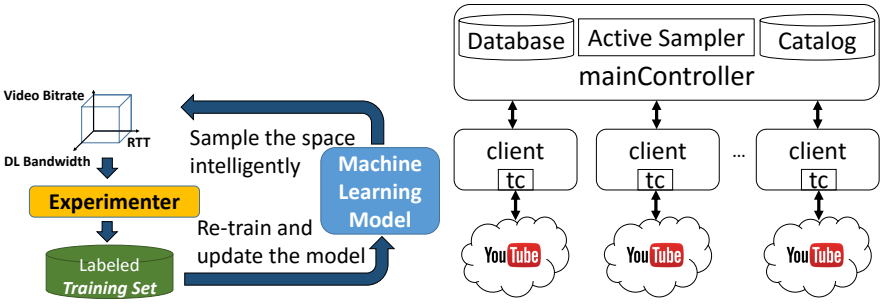
we use a simple definition that can be applied at scale so that the final *YouScore* obtained from this QoE definition has an inherent objective meaning; the value of YouScore would correspond to the percentage YouTube videos in the catalog that play out smoothly for a given network QoS. Nonetheless, the methodology that we present in this chapter can be extended to more complex QoE definitions as well.

Note that for any QoE model, the notion of smooth playout means no stalling and minimal join time. An acceptable join time for any video playout to be considered *smooth* can vary among different users, so a specific threshold for a minimal/acceptable join-time may not apply to every user. However, a user engagement study showed that users started to abandon videos after 2 seconds with the video abandonment rate going up to 20% after 10 seconds [39]. In another work based on a dataset of 300 million video views, almost 95% of the total video views had a join-time of less than 10 seconds [82]. Considering these observations, a threshold on join time of 10 seconds can be considered a reasonable value to assume a smooth playout for an *objective* QoE definition. Finally, we stream videos at fixed resolution by disabling the adaptive mode. Our aim is to quantify the resolutions that can be supported for a given network QoS.

4.3.2 The experimentation framework

In the experimental framework, we apply our active sampling method (HYBRID method in Chapter 3) based on Decision Trees (DT) to sample the experimental space intelligently. The experimental space we use in this chapter is as follows: $0 - 10$ Mbps for *bandwidth*, $0 - 1000$ ms for RTT and $0 - 3$ Mbps for the video bitrates taken from the *webm* video catalog (Figure 4.3b).

At each experiment, we rely on the DT ML model to select an unlabeled sample from the *uncertain* leaves, obtain its label by experimentation, update the training set with the labeled sample and then finally re-train the model to complete the experiment. An overall summary of our approach can be visualized in Figure 4.6a. We stop the experiments when the DT model converges to a stable state. The convergence of the model can be gauged by the *Weighted Confidence* measure that quantifies the quality of the model in terms of its classification probabilities. This measure is computed by taking a weighted sum of the entropies of all the leaves of a DT. The weights are assigned to each leaf according to the geometric volume it covers in the feature space. Thus, bigger regions have greater weights and vice versa. So, at each experiment, we compute the weighted confidence per class for the DT model and stop experimenting when the standard deviation of this measure over consecutive experiments is within 1% of the average for each class (two classes acceptable/unacceptable in our case).



(A) Active Sampling (B) Parallel Experimentation

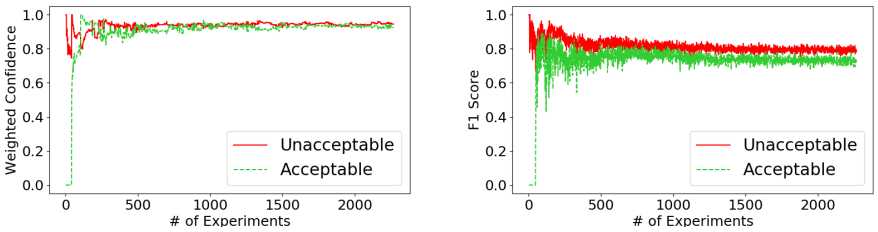
FIGURE 4.6: The sampling and experimentation framework

To further speed up our process of building our dataset, we rely on parallel experimentation. We propose a framework where the logic of choosing the features to experiment with (network QoS and video bitrate) is separated from the client that performs the experiments. With this functional segregation, we can have multiple *clients* running in parallel that ask the central node (*mainController*) for the network QoS and the video ID to experiment with. After completion of an experiment, the results obtained by each client are sent back to the *mainController* which updates the central *database* with the labeled sample and re-trains the active learner. The overall framework is given in Figure 4.6b. This setup can be realized in terms of separate virtual machines within the same network or separate physical machines. A benefit of our framework is that the clients do not need to be at the same physical location, they can be geographically separated provided that they can communicate with the *mainController* over the Internet.

We apply our proposed framework in a network of 11 physical machines in our experimental platform called R2lab [111] where the *mainController* is hosted on a unique machine while the experiments are performed in parallel on the rest of the 10 machines. Our active learning algorithm ended up converging after 2200 experiments. During all these experiments, we had ensured that a large bandwidth and low RTT (less than 10 ms) was available towards the YouTube cloud such that the network degradation was mainly caused by *tc*.

4.3.3 Learner convergence and accuracy

Figure 4.7 shows the DT model's convergence. As we can see in Figure 4.7a, the model achieves a stable confidence value of more than 90% for both classes. To validate the accuracy of the DT-based ML model trained with the QoS-QoE dataset, we rely on using repeated cross-validation. As already discussed before, in k-times repeated cross-validation, the target dataset is split into training and validation sets k times randomly.



(A) The Weighted Confidence (B) Cross Validated F1-Score

FIGURE 4.7: Model convergence for the DT Model per class

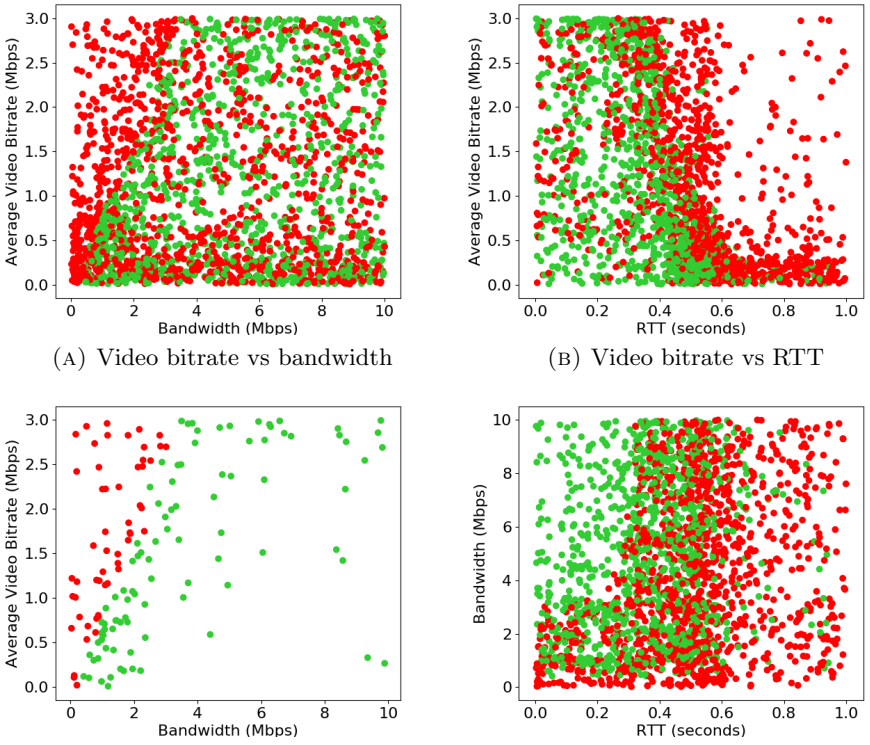
The model is then trained with the training set and tested with the validation set k times to get k accuracy scores. The final accuracy score is then the average of these k scores. We plot the F1-Score[1] based on cross-validation ($k = 3$ with a data split ratio of 80:20 for training and validation) and updated at each iteration in Figure 4.7b. Notice the slight decreasing trend in the cross-validation accuracy compared to the confidence that remains stable. The reason for it is that as we keep experimenting, more and more scenarios will be picked from the uncertain regions nearby the boundary between classes, thus making the resulting dataset noisier and difficult to capture the QoE.

4.3.4 The QoS-QoE dataset

The visualization of the obtained dataset is given in the form of a scatter plot over two dimensions in Figure 4.8. The respective colors represent the corresponding classes. The green points represent those experiments where the videos play out smoothly (*acceptable* QoE) while the red points correspond to *unacceptable* QoE. From Figure 4.8a, we can see a relationship between the video bitrate and the downlink bandwidth. As we increase the video bitrate, the required bandwidth to ensure smooth playout increases. Similarly, for latency, as we increase the RTT, more and more videos move from *acceptable* to *unacceptable* (Figure 4.8b). Notice that the distribution of the points is non-uniform as a consequence of active learning that causes experiments to be carried out with feature combinations near the *decision boundary* in the feature space. To better illustrate this decision boundary, Figure 4.8c is a scatter plot of the data filtered for RTT less than 100 ms from which we can observe a quasi-linear decision boundary between the two classes. Finally from Figure 4.8d, we can see that all video playouts are smooth for a bandwidth higher than 4 Mbps and RTT lower than 300 ms. This means that YouTube videos (having resolutions less than or equal to *1080p* according to our catalog) can play

[1]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 the better classification accuracy of the model.

out smoothly if these two conditions on network QoS are satisfied. The dataset is made available at [30].



(A) Video bitrate vs bandwidth

(B) Video bitrate vs RTT

(C) Video bitrate vs bandwidth (RTT <100 ms)

(D) Downlink bandwidth vs RTT

FIGURE 4.8: Projection of the QoS-QoE dataset. Red color: Unacceptable playout with stalling or long join time. Green color: Acceptable smooth playout.

4.3.5 Model validation

While we use a DT model in our active learning methodology to build the dataset, the resulting QoS-QoE dataset can be used to train other supervised ML algorithms as well. Some ML algorithms such as SVM and neural networks require the standardization of the data (feature scaling) to perform well. Other classifiers such as Decision Trees and Random Forests do not require such standardization. However, to have a fair comparison, we standardize the dataset and then obtain the cross-validation scores for different classifiers. The results are given in Table 4.2 for default parameter configurations using the Python Scikit-learn library. The best three models are neural network (Multi-layer Perceptron), SVM (RBF kernel) and Random Forests giving accuracy around 80%.

ML	Class	Prec	Recall	F1	Avg F1	Fit Time
Nearest Neighbors	0	0.79	0.76	0.77	0.75	1 ms
	1	0.71	0.73	0.72		
Linear SVM	0	0.75	0.76	0.76	0.72	36 ms
	1	0.69	0.67	0.68		
RBF SVM	0	0.82	0.82	0.82	0.8	89 ms
	1	0.78	0.77	0.77		
Decision Tree	0	0.76	0.76	0.76	0.73	7 ms
	1	0.71	0.70	0.70		
Random Forest	0	0.77	0.84	0.81	0.77	51 ms
	1	0.78	0.70	0.73		
Neural Net (MLPC)	0	0.82	0.82	0.82	0.8	1360 ms
	1	0.77	0.77	0.77		
AdaBoost	0	0.80	0.77	0.78	0.76	166 ms
	1	0.72	0.75	0.74		
Naive Bayes	0	0.72	0.69	0.70	0.68	1 ms
	1	0.63	0.67	0.65		
QDA	0	0.78	0.72	0.75	0.74	1 ms
	1	0.69	0.76	0.73		
Logistic Regression	0	0.74	0.76	0.75	0.72	2 ms
	1	0.70	0.67	0.68		

TABLE 4.2: Cross-validation scores for common ML algorithms *with standardization* ($k = 10$ with a test set size equal to 20% of training set). Class 0: *Unacceptable*. Class 1: *Acceptable*.

Features	NN	RF	SVM
RTT, DL_BW, Bitrate	0.795	0.784	0.799
RTT, DL_BW	0.667	0.647	0.658

TABLE 4.3: Performance gain with the video bitrate feature

4.3.6 Gain of using the video bitrate feature

To investigate the gain of using the video bitrate feature in addition to the network QoS features, we train the aforementioned ML algorithms with and without the *video bitrate* feature. The results for the cross-validation accuracy are given in Table 4.3. Modeling the QoE using only the network QoS features results in a low accuracy of about 65%. However, if we use bitrate with the network QoS, the classification accuracy improves to around 80% giving us a gain of around 13% thus validating the importance of using the video bitrate feature in our QoE modeling scenario.

For our subsequent analysis, we use Random Forests as our predictive QoE model, θ, to devise our global QoE indicator, *YouScore*, as they do not require standardization on the training data which allows us to use them directly on new data for prediction without any preprocessing. They also show good classification accuracy according to Table 4.2. We can improve their accuracy further by performing a grid search over parameters of

min number of samples per leaf and *number of estimators*; by doing so, we obtain an accuracy of around 80% with values of 15 and 25 for these parameters respectively. Note here that θ takes as input the features of *RTT*, *downlink bandwidth* and *video bitrate* (representing the network QoS and the video content) with as output an estimation of the binary QoE (*acceptable/unacceptable*), while the derived *YouScore* will only take as input the *RTT* and *downlink bandwidth* to give as output an estimate of the ratio of videos that play out smoothly for a given network QoS.

4.4 YouScore: A QoE indicator for YouTube

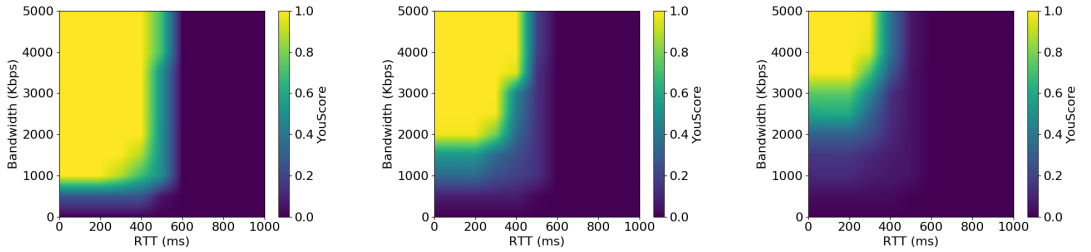
Using θ, we define $YouScore$, a global QoE indicator for YouTube that takes into account the different video contents of our catalog. Theoretically, we aim to give a probability for YouTube videos of a given resolution to play out smoothly, for a given state of network QoS quantified by the tuple of downlink bandwidth and latency (RTT). To obtain such a probability, we test θ over all the videos in the given catalog and use the model's predictions to compute the final QoE score ranging from 0 to 1. Such a QoE score inherently contains meaningful information for network operators to gauge their performance w.r.t YouTube, where a score of x for a given network QoS and a given resolution translates into an estimated $x\%$ of videos of that resolution that would have *acceptable* QoE (start within 10 seconds and play out smoothly). Formally, we define $YouScore$ for a given resolution r and a network QoS as:

$$YouScore_r = f_\theta(bandwidth, RTT). \tag{4.1}$$

The function $f_\theta(bandwidth, RTT)$ is computed by testing θ with a test set $\mathcal{T}_r = \{< bandwidth, RTT, bitrate_i >\}_{i=1}^{N_r}$ composed of N_r samples of same *bandwidth* and RTT while the values for $bitrate_i$ are taken from the video catalog (composed of N_r videos) for resolution r. To elaborate further, let $\mathcal{Y}_r$ denote the set of predictions of θ for $\mathcal{T}_r$ and $\mathcal{Y}_r^{acceptable} \subseteq \mathcal{Y}_r$ denote the set of *acceptable* predictions in $\mathcal{T}_r$. So the final score is given below, which is the ratio of the number of videos to play out smoothly to the total number of videos in the catalog for resolution r:

$$YouScore_r = \frac{\mid \mathcal{Y}_r^{acceptable} \mid}{N_r} \tag{4.2}$$

A single score can be computed by taking a weighted sum such that $YouScore_{final} = \sum_{r=1}^{R} w_r YouScore_r$, where R is the number of resolutions covered in the video catalog and w_r can be chosen to give preference to each of the resolutions.



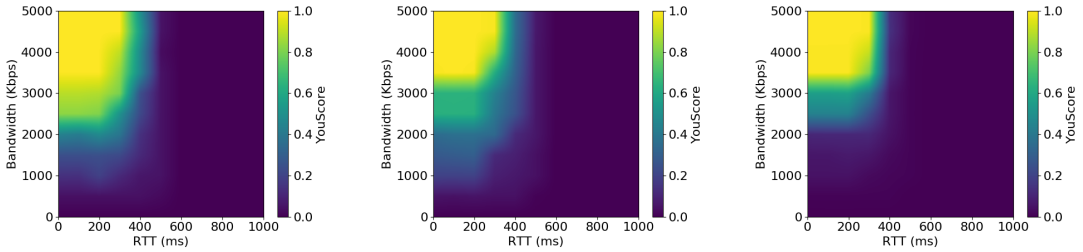
(A) 480p (B) 720p (C) 1080p

FIGURE 4.9: $YouScore_r$ using θ for different resolutions

Using this methodology, we obtain the $YouScore_r$ for each resolution r and plot it in terms of heat maps in Figure 4.9. The plots are a *billinear interpolation* of the YouScores obtained at sampled points in the space of RTT and *bandwidth*: 11 uniformly spaced points on each axis resulting in a total of 121 points. The colors represent the *YouScores* ranging from zero to one. As we can see, the thresholds of *bandwidth* and RTT, where the transitions of the score take place clearly show an increasing trend as we move from the lowest resolution *480p* to the highest resolution *1080p*. For example, the $YouScore$ begins to attain a value of 1 for *bandwidth* of around 1 Mbps for *480p*, 1.5 Mbps for *720p* and 2.5 Mbps for *1080p* for RTT less than 200 ms. This threshold also varies on the RTT axis as well. For *240p* (not shown here), high YouScores are observed for an RTT less than around 500-600 ms. The same threshold reduces to less than 400 ms for resolution *1080p*.

To illustrate the variation in the YouScores across different categories, the model θ is tested over a sampled video set from each *category*. The resulting scores are given in Figure 4.10 for the resolution *1080p*. We can see a difference in the scores obtained between different categories. Consider the "Science & Technology" category, which has a greater spread. This category gets higher scores compared to categories such as "Sports" in the region of low bandwidth and low RTT. From another angle, for low RTT, "Science & Technology" videos obtain a $YouScore_{1080p}$ of 0.5 at the bandwidth of around 2 Mbps, while for "Sports" videos, a higher bandwidth of 3 Mbps is required to achieve the same score. This means that at a bandwidth of 2 Mbps, 50% of "Science & Technology" videos can still play out smoothly whereas no "Sports" videos can play out smoothly at the same bandwidth.

In a practical setting where the global YouScores per resolution need to be computed quickly for a large set of network measurements, we can simply use an *interpolation* function on the sampled points of Figure 4.9. To this end, we store these sampled points in a text file that can be retrieved from [30]. Each line in this file represents a mapping



(A) Science & Technology (B) People & Blogs (C) Sports

FIGURE 4.10: $YouScore_{1080p}^{(category)}$ using θ for different categories

open_uuid	country_location	client_version
open_test_uuid	**download_kbit**	network_mcc_mnc
time_utc	upload_kbit	**network_name**
cat_technology	**ping_ms**	sim_mcc_mnc
network_type	lte_rsrp	nat_type
lat	lte_rsrq	asn
long	server_name	ip_anonym
loc_src	**test_duration**	ndt_download_kbit
loc_accuracy	num_threads	ndt_upload_kbit
zip_code	platform	implausible
gkz	model	signal_strength

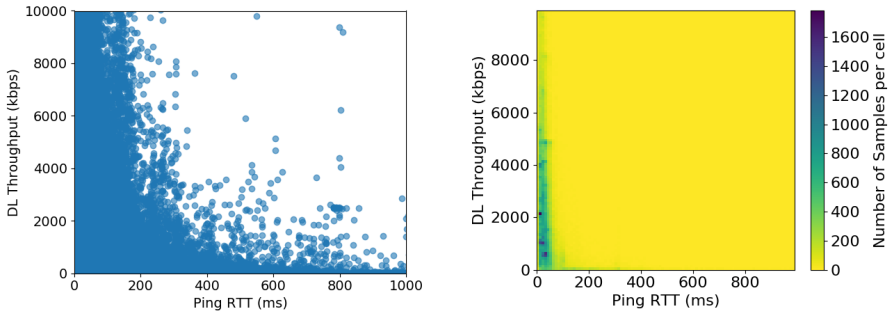
TABLE 4.4: Information provided by *RTR-NetTest*

of the tuples of *RTT* (in milliseconds) and *Bandwidth* (in kbps) to the corresponding *YouScore* for each resolution.

Our proposed YouScore model (Equation 4.1) has the benefit that it requires only two *outband* features of bandwidth and delay to estimate the QoE without requiring the application traffic. Such a model can be easily deployed by a network provider to gauge its performance w.r.t YouTube given the available network measurements. Also, we can use as input to the model, the active measurements carried out by crowd-sourced applications such as *SpeedTest* to estimate the YouTube QoE; in the next section, we demonstrate such a use case.

4.5 Application of the YouScore for network performance benchmarking

In this section, we demonstrate a practical application of *YouScore* on a dataset of real user active measurements. We use an open dataset of network measurements performed by *RTR-NetTest* [21], an application developed by the Austrian Regulatory Authority



(A) Scatter Plot (B) Density map

FIGURE 4.11: Downlink bandwidth vs RTT for *RTR-NetTest* dataset

for Broadcasting and Telecommunications (RTR). This application informs users about the current service quality (including upload and download bandwidth, ping and signal strength) of their Internet connection. The dataset is provided every month; in our analysis, we use the data for November 2017. The dataset consists of more than 200k measurements from users mostly in Austria using mobile, wifi, and fixed-line access technologies. The fields provided in the dataset are shown in Table 4.4; details can be found in [21]. Importantly for us, the dataset includes measurements for downlink throughput (estimated bandwidth) and latency, which are required by the function in Equation 4.1 to obtain the $YouScore_r$ for resolution r ranging from 240p to 1080p. Specifically, for *Downlink bandwidth*, we use the value given by the ratio of *download_kbit* and *test_duration*, and for *RTT*, we use *ping_ms* as input to the function in Equation 4.1 to obtain the predicted $YouScores$.

The visualization of the network measurements is given in Figure 4.11a over a limited scale of up to 10 Mbps for throughput and 1000 ms for RTT. In the dataset, the maximum observed throughput (estimated bandwidth) was 200 Mbps while the maximum RTT went up to 3000 ms. However the bulk of the measurements had smaller values, so we plot here the results over a smaller axis. Notice here the inverse relationship between the download throughput and the latency which is an obvious consequence of queuing in routers and of TCP congestion control. For higher RTT values, throughput is always low which signifies that high RTT alone can become a significant factor alone to predict poor QoE for TCP based applications. The same observation can be observed in $YouScores$ (Figure 4.9) as well where for RTT higher than 600 ms, the scores are zero for all resolutions. To visualize the density of the measurements, we plot a heat map in Figure 4.11b showing that most measurements have RTT less than 100 ms and bandwidth varies in the range of 0 to 4 Mbps.

We now use Equation 4.1 to translate these QoS measurements into the corresponding $YouScores$. The resulting scores are analyzed at the global granularity of

	4G	3G	2G	LAN	WLAN
$YouScore_{240p}$	0.97	0.82	0.05	0.93	0.92
$YouScore_{360p}$	0.95	0.76	0.04	0.89	0.88
$YouScore_{480p}$	0.92	0.65	0.02	0.83	0.83
$YouScore_{720p}$	0.82	0.43	0.01	0.68	0.67
$YouScore_{1080p}$	0.68	0.22	0.01	0.51	0.49
# measurements	28572	6207	723	116617	85879

TABLE 4.5: Average *YouScores* w.r.t network technology for the entire dataset

cat_technology and **network_name**. As this dataset also provides the network names for measurements made in mobile networks, we then split the scores w.r.t to different operators to perform a comparative analysis. The overall performance of each network technology is shown in Table 4.5 where we can see an obvious declining trend in the scores as we increase the resolution. The scores for 2G are mostly zero as expected while we get non-zero values for other technologies. The highest score is obtained in 4G for all resolutions. For measurements with mobile technologies, the network names are provided which allow us to dig further to compare the performance between operators. Figure 4.12 shows the average *YouScores* for top 3 network operators (names are anonymized to ensure unbiasedness). For 240p resolution, the performance is mostly similar among the three but for higher resolutions, a difference becomes evident. Using this information, it can be said that a particular operator performs better than the other in providing better YouTube QoE to its end users.

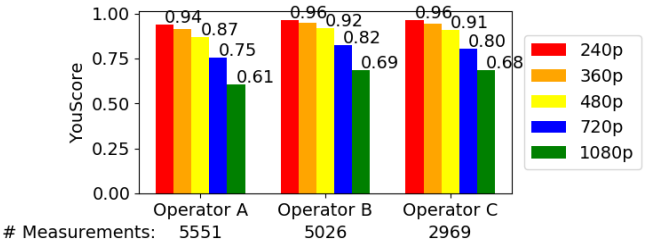


FIGURE 4.12: Overall average *YouScores* for top three network operators for all radio access technologies

Table 4.6 provides the scores for each radio access technology for the given operators where we can indeed see a difference in the scores across the different operators and the different technologies. Notice that the highest score for $YouScore_{1080p}$ is 0.77. Our work is limited to 1080p, for even higher resolutions, the scores would be even less. The analysis in this section can be enhanced further by looking at the geographic locations where the measurements are performed. This can help operators prioritize troubleshooting based on the given YouScores. Also, currently, open coverage maps that provide information such as signal strength, throughputs, delay, etc are getting common. Such maps can be enhanced with *YouScore*, to estimate YouTube QoE as well.

Operator	RAT	count	Y_{360p}	Y_{480p}	Y_{720p}	Y_{1080p}
A	2G	20	0.02	0.01	0.00	0.00
A	3G	1012	0.76	0.63	0.39	0.19
A	4G	4486	0.95	0.93	0.84	0.71
B	2G	88	0.03	0.01	0.00	0.00
B	3G	789	0.92	0.85	0.63	0.35
B	4G	4097	**0.97**	**0.95**	**0.88**	**0.77**
C	2G	42	0.03	0.01	0.00	0.00
C	3G	168	0.85	0.76	0.50	0.29
C	4G	2731	**0.97**	0.93	0.84	0.72

TABLE 4.6: Split of *YouScores* w.r.t Radio Access Technology (RAT) for top three operators

Ref.	QoE Definition	Accuracy	# Features	Type of Features	# Training
[97]	Binary	84%	~ 36	Inband	1464
[5]	3 classes	93%	4	Inband	390,000
[112]	Binary	89%	9	Inband	1060
θ	Binary	80% (93% conf.)	3	Outband (*tc*)	2268

TABLE 4.7: A Performance comparison between QoE models that take network QoS features as input

4.6 Related work

Regarding YouTube catalog analysis, authors in [113] crawled the YouTube site for an extended period and performed video popularity and user behavior analysis for a large number of videos. In our work, we take a similar approach based on crawling YouTube to get a statistical insight into YouTube video bitrates but additionally, we combine this information with QoE modeling as well.

A comparison of the accuracy of the QoS-QoE models developed in the prior work with our binary model θ –which is the basis of *YouScore*– is given in Table 4.7. The cross-validation accuracy for θ is comparable but slightly lower because our dataset has more samples from noisy regions of space due to active sampling. Furthermore, the model we present here uses only two QoS metrics enforced on *tc* (outband measurements) along with the video bitrate as input features whereas the models presented in the literature mostly use a greater number of QoS features directly obtained from the application traffic itself (inband measurements). Having more features directly from traffic traces naturally gives better correlation with the output QoE, thus normally should result in better models; the same is observed in Chapter 5 where the models that are based on inband measurements show greater accuracy than outband measurements.

Regarding mobile network performance analysis using crowd-sourced data, authors in

[114], [115] present cellular network performance studies using data collected from devices located throughout the world. In our work, we propose to use such network QoS datasets to compare and benchmark the performance of mobile networks for video QoE using QoS-QoE predictive models.

4.7 Limitations

We considered smooth play video play out to be defined to have a join time of less than 10 seconds without any stalls. This is by no means a final definition for smooth play out, rather we only use it as a possible use case for showcasing our methodology for comparative analysis of the performance of cellular networks w.r.t video streaming. We can have different and more complex subjective QoE definitions, but overall our proposed methodology for benchmarking remains the same. Furthermore, the results obtained in Figure 4.5 for different mobile operators are highly dependent on the model used and are also not a final representation of the state of today's mobile networks. In terms of generalization, there is still room to refine *YouScore* further by using a much larger catalog and use all available resolutions (going up to 4K) and new video types (such as 3D) that are supported by today's content providers. Also, our work focuses on one version of Google Chrome on Linux based machines and for *webm* videos only. The model can be improved further by considering other browsers and mobile devices as well. Finally, the work in this chapter is based on YouTube, but our overall methodology is reusable to define global QoE scores for any video streaming system.

4.8 Summary

In this chapter, we presented an approach for performance benchmarking of mobile networks for video streaming considering the diversity in the content of the videos in today's content provider systems. Overall, we started by first collecting a large video catalog for a case of YouTube, then used this catalog to build a QoE model to derive a global QoE Score for the target catalog (*YouScore*) and finally apply the global QoE score on a dataset of real user network measurements to get a global visibility into the performance of mobile networks w.r.t video streaming. Our methodology allows both the network and the content providers to gauge their performance w.r.t video QoE and network QoS respectively.

The work in this chapter uses *outband* network QoS for building the QoS-QoE model. In the next chapter, we build ML models using *inband* network QoS to infer video QoE from encrypted traffic.

Chapter 5

Monitoring QoE from Encrypted Video Streaming Traffic

In the previous chapter, we presented a QoE prediction model based on *outband* network QoS features considering a QoE forecasting and benchmarking scenario. In this chapter, we consider the QoE monitoring scenario where the QoE is estimated from within the network using *inband* network QoS. Monitoring QoE of video from within the network is a challenge as most of the video traffic of today is encrypted. In this chapter, we consider this challenge and present an approach based on controlled experimentation and machine learning to estimate QoE from encrypted video traces using network-level measurements only. We consider a case of YouTube and play out a wide range of videos under realistic network conditions to build ML models (classification and regression) that predict the subjective MOS (Mean Opinion Score) based on the ITU P.1203 model along with the QoE metrics of startup delay, quality (spatial resolution) of playout and quality variations. We comprehensively evaluate our approach with different sets of input network features and output QoE metrics. Overall, our classification models predict the QoE metrics and the ITU MOS with an accuracy of 63–90% while the regression models show low error; the ITU MOS (1–5) and the startup delay (in seconds) are predicted with a root mean square error of 0.33 and 2.66 respectively.

5.1 Introduction

In this chapter, we propose a methodology for building ML based models for QoE monitoring using controlled experimentation. In our approach, we again play out a wide range of YouTube videos under emulated network conditions to build a dataset that maps the enforced network QoS to QoE. Prior works [5], [23] have shown good performance of

machine learning in the inference of application QoS features from encrypted traffic (e.g., stalls and startup delay). However, they do not provide any subjective QoE prediction. To fill this gap, we aim to build models that predict not only the application QoS metrics but also a subjective MOS (Mean Opinion Score). For the MOS, we rely on a recently proposed model, the ITU P.1203 [1], that provides a MOS ranging from 1 to 5 taking into account the application QoS features such as the resolution and bitrate of chunks, and, the temporal location and duration of stalling events. We build models that attempt to predict the relevant QoE metrics and the ITU MOS of a video played out. For building the training data, we propose a sampling methodology for network emulation that considers real measurement statistics observed in the wild, and we implement our methodology in a grid computing environment to produce a large dataset mapping the network QoS features to video QoE. Overall, the contributions of the chapter are:

1. We present an experimental framework to build ML models for inferring video QoE (the startup delay, the stalling events, the spatial resolution of playout, the quality switches, and the MOS) from encrypted video traffic traces and apply it to YouTube. Our framework is general and can be used to build QoE estimation models for any video content provider. Furthermore, we ensure that our work is reproducible so we make available the code and the datasets online [31].

2. To the best of our knowledge, this is a first attempt at linking encrypted video traffic measurements to subjective MOS based on models such as ITU-T P.1203. Prior works [5, 23, 24] do not consider the subjective MOS, they consider objective QoE metrics only.

3. We provide a detailed performance comparison for ML modeling (classification and regression) with different types of network feature sets. Specifically, we compare three feature sets, 1) *outband*: the network features are measured outside the traffic pipe configured on the network emulator and include features such as bandwidth and RTT, 2) *inband*: the features are obtained from the traffic traces and include features such as throughput and packet interarrival time, and 3) the *inband* feature set is enriched with the *chunk sizes* which are inferred directly from the traffic traces using a clustering algorithm that we develop and validate. Overall, the best performance is achieved using the third feature set where the ML classification models predict the QoE metrics and the MOS with accuracies between 63% to 90%. For the ML regression models, we obtain low prediction error; the ITU MOS (1–5) and the startup delay (in seconds) are predicted with a root mean square error (RMSE) of 0.33 and 2.66 respectively.

The rest of the chapter is as follows: the related work is discussed in Section 5.2. In Section 5.3, we describe our overall experimentation framework. In Section 5.4, we discuss the features and the subjective QoE model used in the collected dataset followed by its statistical analysis. The evaluation of the ML models is given in Section 5.5 followed by a discussion about the limitations of our work in Section 5.6. Finally, the chapter is concluded in Section 5.7.

5.2 Related work

A comprehensive survey of the QoS-QoE modeling methods is already presented in Chapter 2. Here we elaborate on how the work presented in this chapter is different from some of the closely related prior works.

Inferring QoE related metrics from encrypted traffic of video streaming is a topic of interest in the research community due to the ever growing increase in encrypted video traffic. Recently, authors in [5], [24] use machine learning to predict QoE metrics of stalls, quality of playout and its variations using network traffic measurements such as RTT, bandwidth delay product, throughput and interarrival times. In another similar work [23] based on data collected in the lab, the authors use transport and network layer measurements to infer QoE impairments of startup delay, stalls and playback quality (three levels) in windows of 10 second duration. The prior works do not provide any estimation of the subjective MOS, rather they provide ML models for estimating the objective QoE metrics only. On the other hand, our work demonstrates ML models that not only estimate the QoE metrics but also estimate QoE in terms of subjective MOS based on the ITU P.1203 recommendation. Also, prior works mostly focus on supervised ML classification scenarios, whereas we present ML regression models as well. Also, we present an unsupervised ML based method to infer chunk sizes directly from the encrypted traffic traces (Section 5.4.1) and use them as features for the ML models to show significant improvement in ML accuracy. Prior work [5] uses chunk size as a feature in ML modeling, however, they get its real value from devices they control, instead of inferring it from encrypted packet traces. A first heuristic is proposed in [5] to automatically extract chunk size information from encrypted traffic based on identifying long inactivity periods, but this heuristic is not further developed.

5.3 The experimental setup

In our approach, we play out the YouTube videos under the different network conditions emulated using Linux traffic control, *tc* [110] to build datasets that map the network QoS

to the application QoE. Each experiment consists of enforcing the QoS and observing the QoE of the video played out. The features we use to vary the network QoS are 1) the downlink bandwidth, 2) the uplink bandwidth, 3) the RTT (i.e. bidirectional delay), 4) the packet loss rate (uniformly distributed and bidirectional) and 5) the variability in the delay to model the jitter (standard deviation of the delay following a uniform distribution on *tc*). These five features together define our experimental space.

How to vary the network QoS? In Chapter 3, we have shown that active learning gives a significant gain over uniform sampling in QoS-QoE modeling. However, active sampling requires a single output QoE definition (classification label) to be predefined before the experimentation phase, making the resulting dataset biased towards the given classification model. If we use active sampling, then to study a variety of classification and regression scenarios for different application QoS metrics in our case, we would end up having a different dataset for each classification/regression scenario. To avoid the above-mentioned situation and to reduce the resource wastage problem of uniform sampling, we devise a new sampling framework where we vary the network QoS according to how it is observed in the wild by real users. Our methodology samples the space based on the distribution of real measurements as observed in public datasets of the two well-known mobile crowd-sourced applications *RTR-NetzTest* [21] and *MobiPerf* [22].

5.3.1 Trace based sampling

In our sampling approach, we divide the network QoS space into equally sized regions called cells (20 bins per feature). For each cell, we compute the number of real user measurements observed and use it to derive the probability to experiment in that cell. Each sample in the dataset of *RTR-NetzTest* (1.45 million samples from the period of August 2017 to February 2018) provides measurements of uplink bandwidth, downlink bandwidth, and RTT. However, the dataset does not have packet loss rate and variability of delay. To obtain these last two features, we use data from *MobiPerf* (40k samples for February 2018). MobiPerf's *ping test* provides in a single test the measurements of RTT, loss rate and the standard deviation of the RTT. We combine the measurement distributions of the two datasets using the common feature of RTT and end up having a single probability distribution for the cells in the space of five features. Each cell is assigned a probability equal to the number of measurements in that cell divided by the total number of measurements. Then, upon each experiment, we choose a cell for experimentation based on its assigned probability. From the chosen cell, a random

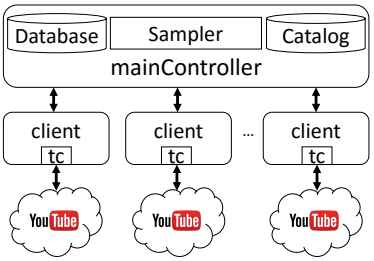


FIGURE 5.1: The experimentation framework

sample, representing the network QoS instance, is selected to be enforced on *tc*[1]. During our experiments, we also verified that the clients had a large bandwidth and low delay towards the YouTube cloud such that the network degradation was mainly caused by *tc*. Note here that by using a trace based sampling approach, we can build a QoS-QoE dataset that is closer to reality than the one based on uniform space sampling.

5.3.2 Video catalog

Today's video content providers have huge catalogs of videos that vary in the content type. To build QoE prediction models for such providers, we take into consideration this diversity of contents by considering a large number of different videos in our experiments. We use the catalog that we build in Chapter 4 that consists of around 1 million different videos supporting HD resolutions (720p and above). This catalog is built by searching the YouTube website using the YouTube Data API with around 4k keywords from Google top trends website for the period of January 2015 to November 2017. We sample this catalog to select a unique video to playout at each experiment.

5.3.3 The overall experimental framework

Our overall framework consists of a *mainController* with several *clients* as shown in Figure 5.1. The *mainController* stores the video catalog and provides the network configurations (using the trace based sampling framework) to the clients which perform the experiments. In each experiment, the *client*, 1) obtains from the *mainController* the network QoS instance and the video ID, 2) enforces the network QoS using *tc*, 3) plays out the selected video in a Google Chrome Browser, 4) obtains the traffic features from the encrypted packet traces and the application QoE metrics from the

[1]We do not consider trace samples which have very large values, we only consider values that are below a certain limit. These limits are 10 Mbps for downlink/uplink bandwidth, 1000 ms for RTT and its variation and 50% for loss rate. Note that almost all the samples in our traces have values within these limits (93% for downlink bandwidth and 98% for the remaining features). We only consider such samples free from outliers to devise our sampling framework.

browser, and 5) reports back the results (QoS-QoE tuple) to the *mainController* which stores the results in a central database. We implement our methodology in a Grid computing environment where up to 30 clients are used to perform the experiments in parallel. We use the Grid5000 [116] and the R2Lab [111] computing platforms for the clients, while the *mainController* is hosted on an AWS EC2 instance [117]. Such segregation allows large scale experimentations to be carried out by clients distributed geographically. Furthermore, we consider a large number of videos to account for the diversity of load incurred by YouTube videos on the network caused by the different content they present. Overall, we collect a dataset of around 100k unique video playouts. In the next section, we discuss the input features and the output QoE metrics we use for building QoE prediction models from this dataset.

5.4 The training dataset

5.4.1 Network features

The network features we use to build our models are collected from the encrypted packet traces in each experiment. These include the statistical metrics of the average, the maximum, the standard deviation and the 10th to 90th percentiles (in steps of 10) for the downlink throughput (in bits per second), the uplink and downlink interarrival times (in seconds) and the downlink packet sizes (in bytes). A single video session (a video running in a Chrome browser) can correspond to several flows (towards the CDNs identified by the URLs ending with googlevideo.com) at the network layer. We collect these features for all such flows that carry the traffic for each video playout. We end up having a total of 48 features comprising the $\mathcal{F}_{inband}$ feature set. With these features, we believe that we can get a fine grained view of what happens during the video stream.

Chunk information from encrypted traffic. In addition to the above-mentioned features and to further improve the modeling, we propose a method to infer the chunk sizes from the encrypted flows of YouTube. In adaptive streaming, the video is delivered to the client in chunks. For YouTube, we observe that the audio and the video chunks are requested using separate HTTP requests. Each video chunk is downloaded with a given resolution/bitrate that may vary according to the network conditions. As videos with higher resolution naturally have higher bitrates and vice versa, the video chunk sizes vary with the resolution. Note that the size of the audio chunks is observed to remain the same across resolutions for the same video playout. The variation of the video chunk size (obtained from clear text HTTP traces extracted in the Chrome browser using the

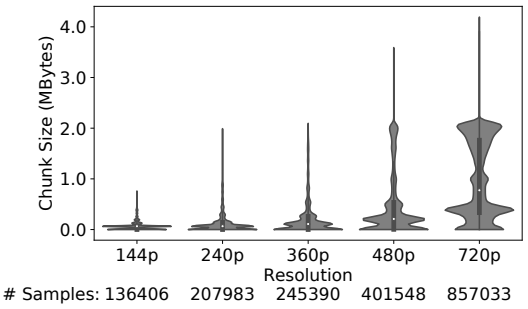


FIGURE 5.2: Variation of the video chunk sizes w.r.t resolution (VP9 codec, 30 fps). Total number of samples: 1848360.

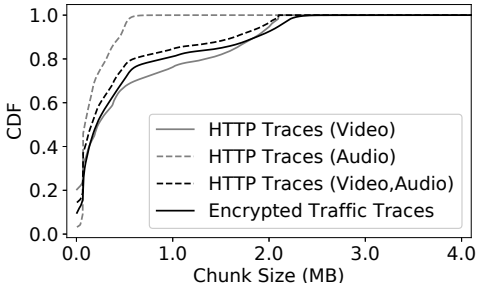


FIGURE 5.3: CDF of the chunk sizes inferred from the encrypted traces compared to the chunk sizes observed in the clear text HTTP traces (obtained in Google Chrome browser) for the same video sessions

chrome.webRequest API) [75] w.r.t each resolution is shown in Figure 5.2. The figure shows that indeed the size of the video chunks tends to increase for higher resolutions.

To infer the chunk sizes from encrypted traffic, we develop the following method. We assume that for each video flow a large sized uplink packet corresponds to a chunk *request* while small packets correspond to *acknowledgments* in TCP/QUIC based flows. We look at the size of each uplink packet and use K-means clustering to segregate the uplink packet sizes into two clusters; the first cluster represents the *acknowledgement* packets while the second cluster represents the *request* packets. Once the uplink *request* packets are identified, we sum up the amount of data downloaded between the *request* packets, which represents the chunk sizes. Figure 5.3 compares the chunk sizes extracted from encrypted traffic traces and the chunk sizes observed in the clear text HTTP traces for the same video sessions. The distribution of the encrypted chunk sizes extracted using our method is close to the combined distribution of audio and video chunks observed in the HTTP traces. Furthermore, the video chunks understandably have larger sizes than audio chunks. As can be noticed, our chunk extraction method cannot distinguish between audio or video chunks, so it does contain some redundant audio chunk information, however, the size of the video chunks especially given their large sizes w.r.t audio chunks is still captured. As we will show later in Section 5.5, and thanks to this information on chunk sizes, we will manage to make it considerably

helpful in improving the QoE models of YouTube.

Overall, for each video session, we obtain an array of chunk sizes extracted from the encrypted traces. We build a feature set composed of the average, the minimum, the maximum, the standard deviation, the 25th, the 50th and the 75th percentiles of the chunk size array. This results in the $\mathcal{F}_{chunks}$ feature set composed of 7 statistical features. We believe that the upper quartiles in this feature set should capture the information of the video chunks due to their larger size.

In addition to the above-mentioned feature sets, we also consider modeling with the five features that are configured on *tc*. We call these features $\mathcal{F}_{outband}$ as they are not gathered from traffic traces but rather configured *out-of-band* on *tc*. They represent the performance of the underlying network access over which the video is played out.

5.4.2 The subjective MOS

The subjective QoE model we use to define our MOS relies on using the ITU recommendation P.1203 [1]. It describes a set of objective parametric quality assessment modules. These are the audio quality estimation module (Pa), the video quality estimation module (Pv), and the quality integration module (Pq). The Pq module is further composed of the A/V integration (Pav) and buffering impact modules (Pb). The Pv module predicts mean opinion scores (MOS) on a 5-point scale [57]. The Recommendation further describes four different quality modules, one for each mode of ITU-T P.1203, i.e., modes 0, 1, 2 and 3. Each mode of operation has different input requirements ranging from only metadata to frame level information of the video stream. Since our goal in this chapter is to target encrypted video streams from a network point of view, we use mode 0 of the model that requires only metadata for each video stream. We consider the video quality only; the audio quality is not considered in this work. The metadata needed to obtain the MOS includes the bitrate, the codec, the duration, the frame rate and the resolution of each chunk. It also considers stalling as well, where for each stall, the media time of stall and its duration are taken as input. The model also considers the "memory" effect where each stalling duration is assigned a weight depending on its location within the video session. Finally, the model considers the device type (mobile/PC), the display size and the viewing distance. In our work, we set the device type to be PC, with display size set to 1280x780 and viewing distance to be 250 cm.

Using the metadata for each stream, the outputs of the model include 1) the final audiovisual coding quality score (O.35), 2) the perceptual stalling indication (O.23), and 3) the final media session quality score (O.46). These scores are calculated by the Pv and the Pb modules. O.35 considers the visual quality that is dependent on

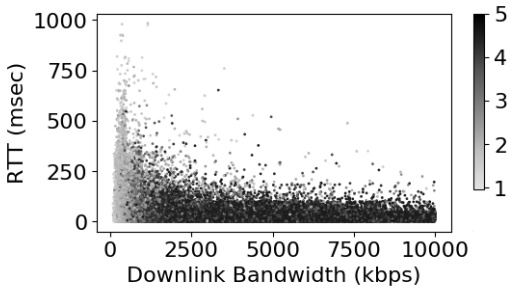


FIGURE 5.4: Variation of the ITU MOS w.r.t configured bandwidth and RTT. # samples: 104504. Unique video count: 94167.

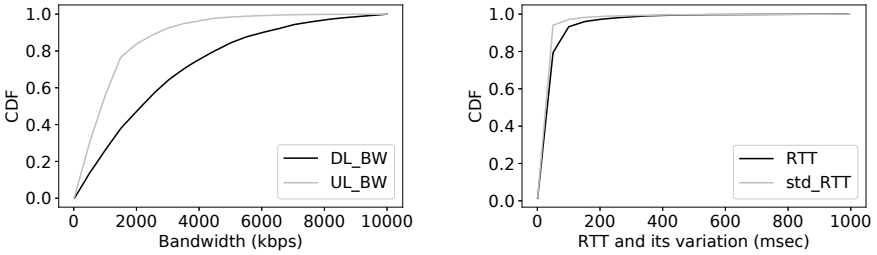
the resolution and bitrate of playout while O.23 considers stalling and startup delay. Overall the final output of the model is O.46 that considers both O.23 and O.35. We use O.46 as the final MOS and calculate it using a python implementation provided in [118], [119]. This python module requires input metadata of the bitrate, the resolution of each chunk, and the stalling information, if any, for the video played out. We obtain this information from the clear text HTTP traces in the Chrome browser and the YouTube API. Specifically, we extract *itag*, *range*, *mime* and *clen* parameters for each chunk request to infer the resolution, the codec and the bitrate of each chunk (assuming equal duration of chunks) while the timestamps and the duration of each stalling event are obtained from the YouTube API.

It should be noted here that YouTube videos can play out in either H.264 (mp4) or VP9 (webm) codec while the current version of the ITU-T model is standardized for H.264 codec only; standardization for VP9 is in progress [120]. To get MOS values for VP9, we use the mapping function provided by the authors in [118] to translate the H.264 MOS to a VP9 MOS for videos that play with the VP9 codec.

The visualization of the collected dataset is given in Figure 5.4 that shows the variation of the two network features of downlink bandwidth and RTT with the corresponding MOS. An obvious relationship between QoS and MOS is visible here: as the bandwidth increases (or RTT decreases), the MOS increases. Notice the absence of experiments in regions where the bandwidth and the RTT both are high. This non-uniformity comes from our trace based sampling framework (Section 5.3.3) where TCP based measurements achieve high throughput with low RTT and vice versa.

5.4.3 Statistical analysis of the collected dataset

In this section, we present the statistical analysis of our dataset to get insights into the variation of the network QoS, the observed application QoS metrics (startup delay, stalls, quality switches and resolution of playout) and the MOS.



(A) Bandwidth (B) RTT and its variation

FIGURE 5.5: The CDF for $\mathcal{F}_{outband}$ enforced on *tc*

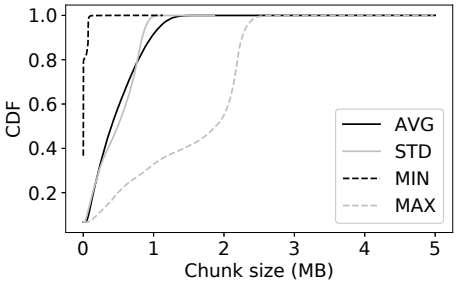


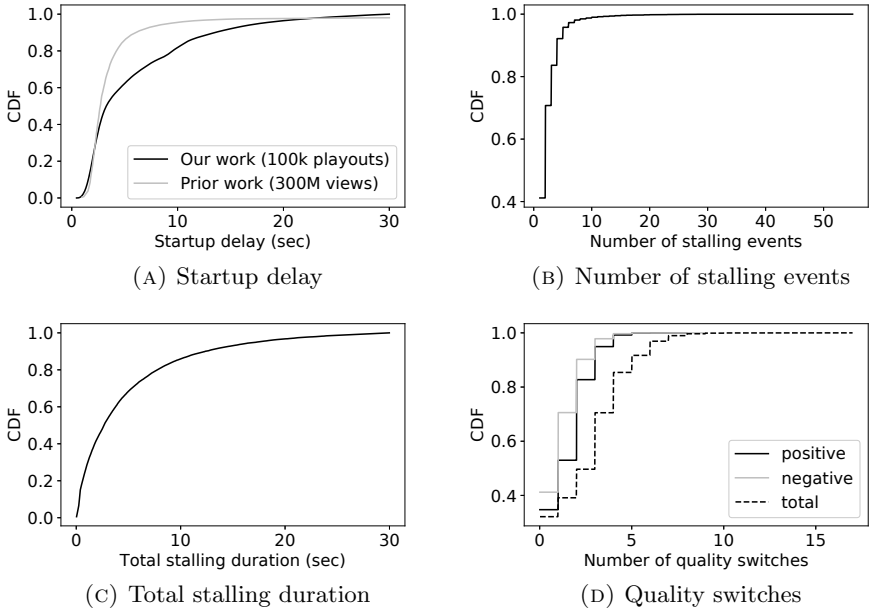
FIGURE 5.6: The CDF of the statistical *chunk size* features ($\mathcal{F}_{chunks}$)

5.4.3.1 Network features

Figure 5.5 shows the distribution of the network features configured on *tc* ($\mathcal{F}_{outband}$) using our sampling methodology. From these figures, we can see that 80% of the experiments have the uplink bandwidth of less than 1.7 Mbps while the downlink bandwidth remains below 5 Mbps. Similarly, the RTT and its variation are mostly concentrated around 100 ms while the loss rate is less than 2.3% for 90% of the total experiments (CDF for loss rate not shown here).

In $\mathcal{F}_{inband}$, the average downlink throughput per playout gets a mean value of 1.89 Mbps compared to the mean of the configured downlink bandwidth of 2.72 Mbps (average of DL_BW in Figure 5.5a). Furthermore, the mean values for the average downlink and uplink packet interarrival times are 13 and 14 ms respectively while the downlink average packet size gets a mean value of 1485 bytes.

In Figure 5.6, we show the CDF of the statistical features of the chunk sizes ($\mathcal{F}_{chunks}$). Note that the minimum chunk size can be very small; for 80% of our experiments, we observe a chunk size of less than 8.9 KB. On the other hand, the maximum chunk size can go up to 2.19 MB. The average and the standard deviation of the chunk size have similar distributions with median values of 400 KB and 493 KB respectively.



(A) Startup delay (B) Number of stalling events

(C) Total stalling duration (D) Quality switches

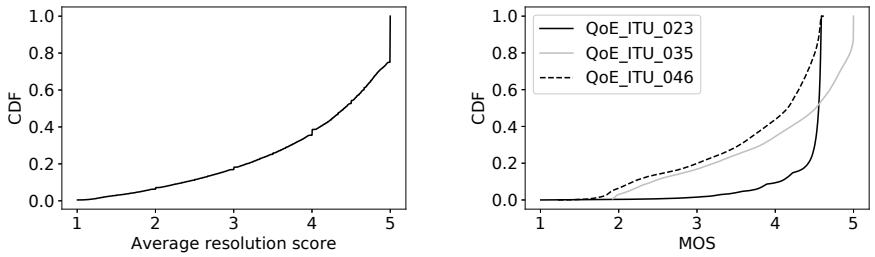
FIGURE 5.7: The CDF of the startup delay, the stallings and the quality switches

5.4.3.2 Startup delay

In our work, we define a timeout for each experiment to be 30 seconds to prevent experimenting with unreasonably large startup delays. In real user datasets, the startup delays are mostly lower than this threshold. For example, in a study based on a dataset of 300 million video views, around 98% of the total views had a startup delay of less than 30 seconds [82]. In our dataset, 93% videos start to play within 30 seconds while the remaining 7% timeout. The variation of the startup delays for the videos that started playing is shown in Figure 5.7a where the distribution is observed to be non-uniform with 80% of experiments having a startup delay of less than 10 seconds. As seen in Figure 5.7a, this observation has also been made in prior work [82] based on real user measurements, with an even more pronounced skewness towards lower startup delays.

5.4.3.3 Stalls

Since YouTube is based on HTTP Adaptive Streaming (HAS), we expect that stalls should be rare events. Indeed we observe the same in our dataset as only 12% of the video playouts suffer from stalling. We plot the CDF of the number of stalling events and the total stalling duration per playout in Figures 5.7b and 5.7c considering videos that at least had one stall. From the figure, we can see that around 40% of the videos that stalled had only one stall while 70% of the total stalled videos suffered up to two stalls only. Overall, 80% of the stalled videos had a total stalling duration of less than 7 seconds (Figure 5.7c).



(A) Average resolution score (B) ITU-T MOS

FIGURE 5.8: The CDF of the average resolution score, r and MOS

5.4.3.4 Quality switches

Contrary to stalls, we observe that quality switches are more common in our dataset due to HAS; 73% of the video playouts have at least one quality switching event. A quality switch can either be "positive" i.e. the resolution of playout increases from low resolution to high, or it can be "negative" if the resolution decreases. The CDF of these events is shown in Figure 5.7d where both the positive and the negative switch events are shown to have similar distributions.

5.4.3.5 The average resolution score

We measure the resolution of each chunk from the HTTP traces and assign a score to the resolution of each chunk, i.e., 1 for 144p, 2 for 240p, 3 for 360p, 4 for 480p and 5 for 720p and above. The *average resolution score*, r, corresponds to the mean of these scores and ranges from 1 to 5. Figure 5.8a shows that the distribution r is biased towards higher values. This is due to the measurement datasets used in our sampling methodology (Section 5.3.1) resulting in more experiments with *good* network conditions as seems to be the case with real users.

In our dataset, we observe that the highest resolution mostly goes up to 720p even though we used videos that propose even higher resolution (up to 1080p). Since the viewport in the Chrome browser in our experiments had a size of 1280x780, the YouTube player seems not requesting higher resolution segments. This suggests that YouTube client considers the viewport in choosing the quality of playout.

5.4.3.6 The ITU P.1203 MOS

Finally, the CDF of the resulting ITU MOS is shown in Figure 5.8b. The maximum values for O.23 are between 4.5 and 5.0 which is also true for the MOS (O.46). However, the CDF for O.46 is more similar to O.35 which means that in our dataset, the ITU MOS

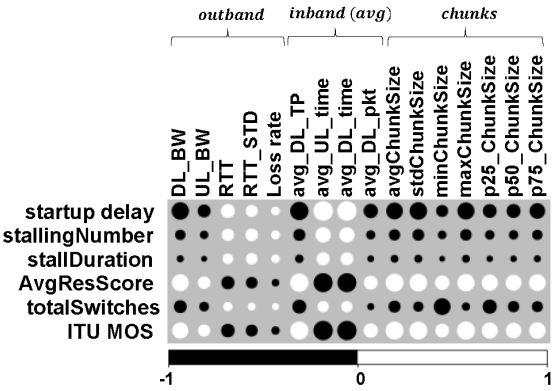


FIGURE 5.9: Correlogram for network QoS, app QoS and MOS

is affected more by the resolution and bitrate of playout compared to the rebufferings (startup delay and stalls).

5.4.4 Correlation analysis between network QoS and QoE

To understand the relationship between the input network features and the QoE, we measure the corresponding Pearson correlation coefficients and plot the correlation matrix of the feature sets in Figure 5.9. From the figure, we can see that the startup delay, the average resolution score, and the MOS have strong correlation with all the features of $\mathcal{F}_{outband}$ except for the packet loss rate while the stall duration, the stalling number (total number of stalls in a given playout) and the quality switches show little correlation with $\mathcal{F}_{outband}$. For the features in $\mathcal{F}_{inband}$, the downlink throughput and the inter-arrival times show the strongest correlation with the QoE metrics (except for stalls and number of switches) while the downlink packet size shows a comparatively lower correlation. Finally, for $\mathcal{F}_{chunks}$, the correlation with the QoE metrics is again strong except for the metrics related to stalls. The number of switches shows a better correlation here compared to the preceding two features sets with *minimum chunk size* showing the strongest correlation. To summarize, all the QoE related metrics have a good correlation with our network feature sets except for stalling; we will discuss the potential reason for this in the next section where we evaluate the ML models built from these features.

5.5 Evaluation of the ML Models

In this section, we discuss the performance of the ML models built using our dataset. We try to predict application QoS metrics and MOS using network QoS. We aim at assessing how well the following questions can be answered:

1. Predict if the video starts to play or not.
2. If the video starts, estimate the startup delay.
3. Predict if the video plays out smoothly or has stalls.
4. Predict if there are any quality switches.
5. Estimate the average resolution of playout.
6. Estimate the final QoE in terms of the ITU MOS.

We consider three sets of features; $\mathcal{F}_{outband}$, $\mathcal{F}_{inband}$ and $\mathcal{F}_{inband+chunk}$ to build predictive ML models. Our work considers both the classification and the regression scenarios as we have both discrete as well as continuous output labels.

5.5.1 Estimating startup delay

To estimate the startup delay, we need to first classify if the video started to play out. We classify the video as *started* if the startup delay is less than 30 seconds and *not started* if a timeout occurs. Considering this binary classification problem, we train a Random Forest (RF) ML model (using python Scikit-Learn library [121] in default parameter configurations) with our dataset and evaluate its accuracy using repeated cross-validation. In repeated cross-validation, the 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. The metric for classification accuracy we use is the F1-Score[2] with $k = 5$ and a data split ratio of 80:20 for training and validation. The ML performance resulting from training RF with each of the three feature sets is given in Table 5.1. The models obtain over 90% accuracy for the three sets. Furthermore, the performance improves if $\mathcal{F}_{inband}$ is used compared to $\mathcal{F}_{outband}$. With $\mathcal{F}_{inband+chunks}$, we get the best average F1-score of over 99%.

We now try to estimate the startup delay using ML regression models trained with *started* videos. We compare models based on *Linear Regression* and *Random Forest (RF) Regression*. The evaluation is done based on the well known metric of Root Mean Squared Error (RMSE) given by $\sqrt{E[(\hat{y} - y)^2]}$, where y are the true labels and $\hat{y}$ are the predictions made by the regression model. The results are shown in Table 5.2 where we obtain the minimum RMSE of 2.66 seconds using *Random Forest (RF) Regression*

[2]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.

Feature Set	Class	Count	Prec	Recall	F1	Avg F1
$\mathcal{F}_{outband}$	0	7401	85.4	84.5	85.0	92.0
	1	104518	98.9	99.0	98.9	
$\mathcal{F}_{inband}$	0	7401	95.4	95.1	95.2	97.4
	1	104518	99.7	99.7	99.7	
$\mathcal{F}_{inband+chunks}$	0	7401	99.8	99.9	99.9	99.9
	1	104518	100.0	100.0	100.0	

TABLE 5.1: ML classification accuracy of predicting video start up. Class 0: not started (startup delay > 30 seconds). Class 1: video started to play.

Feature Set	Regression Algo	RMSE
$\mathcal{F}_{outband}$	Linear Regression	4.28
	RF Regression	2.95
$\mathcal{F}_{inband}$	Linear Regression	2.99
	RF Regression	2.82
$\mathcal{F}_{inband+chunks}$	Linear Regression	2.96
	RF Regression	2.66

TABLE 5.2: RMSE (in seconds) for the predicted startup delay

with $\mathcal{F}_{inband+chunks}$. A noticeable observation is that we can obtain a reasonably low RMSE of 2.95 seconds if we use RF Regression with $\mathcal{F}_{outband}$, which can be explained by the fact that the startup delay is determined by the initial buffering time, which in turn is determined by network performance. This can justify in practice the use of out-of-band measurements alone to predict startup delay without relying on accessing the video traffic i.e. predict startup delay without playing out the videos as is the case in QoE forecasting.

5.5.2 Predicting quality switches and stalls

In this section, we present the results of the ML models for detecting the quality switches and the stalls. As YouTube uses HAS, the resolution of the videos played out can change depending upon the network conditions. To detect these quality changes, we consider a binary classification scenario where the output label 0 corresponds to a video being played out with constant quality, while the label is 1 if the video changes its quality at least once. Using the same methodology of Section 5.5.1, we obtain the performance results given in Table 5.3. We observe that with feature sets $\mathcal{F}_{outband}$ and $\mathcal{F}_{inband}$ we do not get good accuracy scores suggesting that the conventional network and traffic features used in prior works are not sufficient to predict quality switches from encrypted traces. However, if we include the $\mathcal{F}_{chunks}$ feature set for prediction, we can see a significant improvement with over 90% precision and recall for quality switch

Feature Set	Class	Count	Prec	Recall	F1	Avg F1
$\mathcal{F}_{outband}$	0	28577	44.5	40.6	42.5	61.1
	1	75941	78.4	81.0	79.7	
$\mathcal{F}_{inband}$	0	28577	55.4	54.1	54.7	68.9
	1	75941	82.8	83.6	83.2	
$\mathcal{F}_{inband+chunks}$	0	28577	87.9	84.2	86.0	90.4
	1	75941	94.1	95.6	94.9	

TABLE 5.3: ML classification accuracy of detecting quality (resolution) switches. Class 0: video plays at constant quality. Class 1: there are quality switches.

Feature Set	Class	Count	Prec	Recall	F1	Avg F1
$\mathcal{F}_{outband}$	0	91804	89.5	97.8	93.4	59.2
	1	12714	51.0	16.6	25.0	
$\mathcal{F}_{inband}$	0	91804	89.7	97.9	93.6	60.8
	1	12714	54.8	18.7	27.9	
$\mathcal{F}_{inband+chunks}$	0	91804	90.2	98.0	93.9	63.1
	1	12714	59.8	22.1	32.2	

TABLE 5.4: ML classification accuracy of detecting stalls. Class 0: video plays smoothly. Class 1: video has stalls.

detection. This improvement comes from the relationship between chunk sizes and the corresponding resolutions (Figure 5.2).

For the scenario of detecting stalls, we again consider a binary classification scenario where the label 0 corresponds to the video playing out smoothly without any stall while a label of 1 is assigned if there is stalling in the playout. The accuracy of the obtained ML model is shown in Table 5.4. A surprising observation here is that the model suffers from low recall for stall detection suggesting that the model fails to detect stalls. This is contrary to what is seen in the literature where similar statistical features were used. The reason is that prior works for stall detection used datasets mostly consisting of videos without HAS; in [97] the videos played out with constant quality, while in [5], the dataset had 97% videos that did not use HAS. However, in our work, we have HAS for all the experiments and use a very diverse set of video contents, which explains why the statistical network features fail to detect stalls. Indeed, the features we consider capture the network QoS holistically, i.e., for the entire video session without considering the temporal aspects of playout. As we know that stalling can occur at any time instant during playout, capturing those temporal degradations is very important to accurately detect stalls. For example, the video bitrate at the time of stall might be suddenly high while being low otherwise. In such a scenario, the overall statistical features would look similar to a normal playout making stall detection difficult. We believe that in the presence of HAS, stalls become rare events that depend on the temporal properties of the video playout, and thus, can hardly be detected with session-level QoS features.

Feature Set	Class	Count	Prec	Recall	F1	Avg F1
$\mathcal{F}_{outband}$	LD	27077	74.3	74.0	74.2	
	SD	29717	42.6	36.1	39.1	61.5
	HD	47724	68.1	74.8	71.3	
$\mathcal{F}_{inband}$	LD	27077	74.3	76.9	75.6	
	SD	29717	51.4	45.2	48.1	66.3
	HD	47724	73.2	77.2	75.1	
$\mathcal{F}_{inband+chunks}$	LD	27077	78.5	85.5	81.8	
	SD	29717	65.1	65.6	65.4	77.3
	HD	47724	87.1	82.3	84.6	

TABLE 5.5: ML classification accuracy of detecting resolution

Feature Set	Regression Algo	RMSE
$\mathcal{F}_{outband}$	Linear Regression	0.684
	RF Regression	0.474
$\mathcal{F}_{inband}$	Linear Regression	0.454
	RF Regression	0.428
$\mathcal{F}_{inband+chunks}$	Linear Regression	0.423
	RF Regression	0.331

TABLE 5.6: RMSE of the predicted MOS

5.5.3 Estimating average resolution of playout

To build a ML model that detects the average resolution of playout, we convert the average resolution score, r (defined in Section 5.4.3.5) into three resolution levels: 1) low definition (LD), if $1 \leq r \leq 3.5$, 2) standard definition (SD), if $3.5 < r \leq 4.5$, and 3) high definition (HD), if $4.5 < r \leq 5$.

For a classification scenario with three output labels, the model performance using Random Forests (RF) is given in Table 5.5. We again see that by using $\mathcal{F}_{inband+chunks}$, we improve the overall F1-scores of the model because of the positive relationship between chunk size and resolution. However, the individual F1-scores for SD are lower compared to LD and HD. This is because SD is an intermediary class and usually the accuracy for such classes is lower compared to edge classes in cases such as ours where the output labels and input features form a monotonic relationship. Overall, the best performance is obtained for class HD where the precision of 87% is achieved.

5.5.4 Estimating the ITU MOS

We consider both regression and classification scenarios for predicting the MOS based on the ITU P.1203 model. Table 5.6 shows the RMSE with the RF and the linear regression models. Similar to the observation in Section 5.5.1, the RMSE decreases

Feature Set	QoE	Count	Prec	Recall	F1	Avg F1
$\mathcal{F}_{outband}$	1	5391	68.1	68.8	68.4	60.0
	2	12881	58.6	58.4	58.5	
	3	22363	39.8	24.0	30.0	
	4	63869	78.2	89.1	83.3	
$\mathcal{F}_{inband}$	1	5391	69.0	74.1	71.5	63.7
	2	12881	62.0	63.2	62.6	
	3	22363	44.7	31.0	36.6	
	4	63869	80.3	88.1	84.0	
$\mathcal{F}_{inband+chunks}$	1	5391	71.6	76.1	73.8	73.2
	2	12881	65.4	69.8	67.6	
	3	22363	63.6	59.8	61.6	
	4	63869	89.7	90.0	89.8	

TABLE 5.7: ML classification accuracy with quantized MOS. Bins: 1) 1.0 – 2.0, 2) 2.0 – 3.0, 3) 3.0 – 4.0, 4) 4.0 – 5.0).

		Predicted			
	Class	1	2	3	4
Original	1	0.76	0.22	0.01	0.00
	2	0.11	0.71	0.15	0.03
	3	0.01	0.13	0.59	0.27
	4	0.00	0.01	0.08	0.90

TABLE 5.8: MOS confusion matrix using $\mathcal{F}_{inband+chunks}$

with enrichment of the input set with relevant features. The lowest RMSE of 0.33 is obtained using the $\mathcal{F}_{inband+chunks}$ feature set and RF regression. Table 5.7 shows the MOS prediction results where we quantize the MOS into 4 classes and then train the RF classifier. As we enrich the feature sets, we see improvement in performance across all classes. With $\mathcal{F}_{inband+chunks}$, the precision per class ranges from 63% to 90% while the recall per class ranges from 60% to 90%. Overall, we get an average F1-score of 73%. Note that the accuracy for the intermediary classes 2 and 3 is lower compared to the edge classes 1 and 4 due to the monotonic relationship between our features and the MOS. Furthermore, if we look at the confusion matrix in Table 5.8, we can observe that the misclassifications mostly occur to the adjacent classes.

5.6 Limitations

The dataset presented in this work was collected using Google Chrome browser based on Linux machines; we do not consider mobile devices or other browsers. The effect of network QoS on QoE may vary across different platforms and devices which is not handled by the models in this work.

Our dataset is based on TCP based video flows for YouTube collected in April 2018. Over time, we expect that video content providers can change their video delivery mechanisms which would limit the applicability of our model to be used in the future. So, over a period of time, the training data would have to be re-collected again to ensure that the models are updated according to the latest version of the targeted video provider.

The results for the MOS prediction are based on only one set of parameters used for the ITU P.1203 model. With a different set of parameters e.g. different screen sizes, the resulting MOS distribution will also change which would change the performance of the ML models as well.

5.7 Summary

In this chapter, we presented our methodology for building QoS-QoE prediction models from encrypted Internet video traffic using controlled experimentation and machine learning. The models presented predict not only the application level QoS features but also the subjective QoE modeled as MOS according to the ITU-T P.1203 model. Overall our experimentation approach is re-usable to construct large datasets in distributed computing environments for any video content provider.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we presented methodologies for building ML-based QoE prediction models for Internet video streaming. We first demonstrated the use of active learning in the scenario of controlled experimentation for building QoE forecasting models for YouTube in Chapter 3 and showed that active learning allows building models quickly with fewer experiments compared to uniform sampling. We then apply our active sampling methodology for building a QoE indicator called YouScore considering the large diversity of contents offered by today's content providers such as YouTube in Chapter 4. We use our sampling approach to intelligently sample the content space (defined by the video encoding bitrate feature) and the network QoS space (the downlink bandwidth and the delay) to build a model and subsequently the YouScore. Finally, in Chapter 5 we consider the problem of inferring QoE from encrypted video traffic where we rely on the inband network QoS measurements to predict the QoE in terms of the relevant QoE metrics of startup delay, stalls, video resolution, quality switches, and ITU MOS.

6.2 Future work

6.2.1 Active learning for QoE modeling of any Internet application

The active sampling approach we devise in this thesis is general and can work in any QoS-QoE modeling scenario for any Internet application, which can include applications such as VoIP (Skype, Viber, Whatsapp, etc.) and Web Browsing. Each application scenario would its own QoE definition, which can be subjective (with real users) or objective labels. Objective labels can be e.g., PESQ for VoIP or Google's SpeedIndex for Web

QoE. The QoS-QoE modeling setup for data collection may differ but the underlying active sampling framework would remain the same. This means that our sampling framework can be used to model QoE of any Internet application.

6.2.2 Application of the QoE models in practice

The YouScore model can easily be deployed by network operators who have end-user measurement data available in their networks. Future work can consider improving and updating the YouScore model by considering all the available resolutions of the catalog going up to 4K and consider 3D videos as well; in our work, we consider the maximum resolution of 1080p and only use conventional 2D videos only.

Predicting the subjective MOS of video from encrypted traffic is challenging. The models built in Chapter 5 obtain the MOS prediction F1-score of 73% for the ML classification scenario (4 labels) and 0.331 RMSE for the ML regression scenario. Further research work can be carried out to improve the accuracy of the models further with transport layer (TCP/QUIC) features and additional video content level features such as the variability of video bitrate as well. Also, in our work, we considered a controlled experimentation scenario where we know the exact time of the start and end of the video playout as we have separate network traces for each experiment. In practice, a given network interface of the passive QoE monitoring device (e.g., the PGW in mobile networks) can carry the video traffic for a large number of users. For the prediction models to work accurately in such cases, each flow has to be identified first and has to be assigned to a video session accurately. This can be a challenge since a video session can correspond to one or more flows, and if for some reason, the transport layer connection resets, a new flow might be generated in the middle of a playout making it difficult to distinguish whether this new flow belongs to a new session or the previous session. Further research can target this problem to make the application of the QoE monitoring models to be used in practice.

We demonstrated the application of our models for QoE forecasting, QoE monitoring and network performance benchmarking. Other potential areas where these models can be used are network dimensioning and management, optimizing video delivery in HAS, and performance evaluation of emerging technologies such as Blockchains.

Network dimensioning and management. The QoE models developed in this thesis can also be used by network operators for capacity planning and optimization. The models explicitly provide the QoS required for a certain level of QoE. This relationship can be used to dimension networks according to different QoE requirements. Also, the models can be used to optimize resource allocation as well. Prior works have shown that

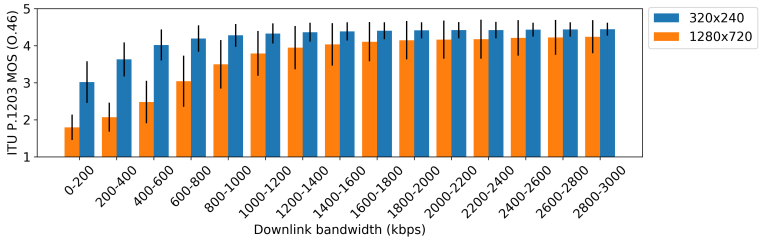


FIGURE 6.1: Average ITU MOS vs downlink bandwidth (outband QoS) for different display resolutions in the dataset presented in Chapter 5. The error bars represent one standard deviation.

the end users' display/screen resolution also affects the perceived video QoE; the ITU P.1203 model also takes as input the parameter for the display resolution. This means the QoS required to get a certain level of QoE for different display resolutions will vary. For example, a user with a small screen requires a lower network bandwidth compared to a user with a larger screen to attain the same QoE level. This is visible in Figure 6.1 where we can observe that with a display resolution of 240p, the network throughput required to achieve an ITU MOS of 4.0 is on average lower than the throughput required by the 720p display. This relationship can be used to optimize the network resource allocation such that the overall QoE of all users in the network is maximized according to the given QoS-QoE models.

Designing ABR algorithms for subjective MOS prediction models. Most works on the design of ABR algorithms, such as [46], [50], [52], try to maximize an objective QoE function (composed of QoE metrics such as stalls and video bitrate), these works do not consider the subjective MOS. A new research direction can be to devise ABR algorithms that maximize subjective MOS based on models such as ITU P.1203. A potential future work can focus on designing Reinforcement Learning (RL) based ABR algorithms that maximize the ITU MOS. Here, the ITU MOS has to be used as the total reward in the reinforcement learning process, which means that the reward at each time step is not available (the time instants when the actions are taken by the policy) as the ITU MOS can only be computed for the entire playout. For such a scenario, algorithms such as A3C [122], which require the reward at each time step, cannot be used. However, the baseline policy gradient algorithm such as REINFORCE [123] can work without the rewards per time step, but the learning process may be slow or suboptimal [124]. So, future research can focus on overcoming this challenge to devise RL based ABR algorithms that maximize subjective MOS.

Extending the QoS-QoE modeling approach for performance evaluation of emerging technologies. The QoS-QoE modeling methodology demonstrated in this thesis can also be used for the performance evaluation of new technologies such as

Blockchains. A potential work can be to build prediction models using ML and controlled experimentation that predict the performance of Blockchain-as-a-Service (BaaS) infrastructures[1]. The network conditions can affect the performance of blockchains as a higher latency can reduce the transaction throughput, i.e., the rate at which valid transactions are added to the blockchain. Modeling this relationship by controlled experimentation can help providers predict the performance of the blockchain infrastructures, which can allow better troubleshooting and ensure better visibility into the blockchain performance both for the providers and the clients.

[1]In BaaS, an external service provider is responsible for setting up and maintaining the blockchain technology and infrastructure for a customer [125]. By using the BaaS model, the clients do not have to worry about the backend issues of the infrastructures, so they can focus only on the core blockchain application (e.g., smart contracts) itself.

Bibliography

[1] ITU-T Rec. P.1203. Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport. *ITU-T Rec. P.1203*, 2017.

[2] Cisco Visual Networking Index (VNI) Forecast, 2016. `https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html`.

[3] Ericsson Mobility Report, June 2018, 2018. `https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf`.

[4] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz. Quantification of youtube qoe via crowdsourcing. In *Multimedia (ISM), 2011 IEEE International Symposium on*, pages 494–499, Dec 2011.

[5] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. Measuring video qoe from encrypted traffic. In *Proceedings of the 2016 Internet Measurement Conference*, pages 513–526, 2016. ISBN 978-1-4503-4526-2.

[6] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. Measuring the quality of experience of http video streaming. In *Integrated Network Management*, pages 485–492. IEEE, 2011. ISBN 978-1-4244-9221-3.

[7] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, Feb 2017. ISSN 1932-4553. doi: 10.1109/JSTSP.2016.2608329.

[8] Özgü Alay, Andra Lutu, Miguel Peón-Quirós, Vincenzo Mancuso, Thomas Hirsch, Kristian Evensen, Audun Hansen, Stefan Alfredsson, Jonas Karlsson, Anna Brunstrom, Ali Safari Khatouni, Marco Mellia, and Marco Ajmone Marsan. Experience: An open platform for experimentation with commercial mobile broadband networks. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, pages 70–78, New York, NY,

USA, 2017. ACM. ISBN 978-1-4503-4916-1. doi: 10.1145/3117811.3117812. URL http://doi.acm.org/10.1145/3117811.3117812.

[9] A. Schwind, M. Seufert, Ö. Alay, P. Casas, P. Tran-Gia, and F. Wamser. Concept and implementation of video qoe measurements in a mobile broadband testbed. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–6, June 2017. doi: 10.23919/TMA.2017.8002921.

[10] A. K. Moorthy, K. Seshadrinathan, R. Soundararajan, and A. C. Bovik. Wireless video quality assessment: A study of subjective scores and objective algorithms. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(4):587–599, April 2010. ISSN 1051-8215. doi: 10.1109/TCSVT.2010.2041829.

[11] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia. Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558, Feb 2014. ISSN 1520-9210. doi: 10.1109/TMM.2013.2291663.

[12] C. Wu, K. Chen, Y. Chang, and C. Lei. Crowdsourcing multimedia qoe evaluation: A trusted framework. *IEEE Transactions on Multimedia*, 15(5):1121–1137, Aug 2013. ISSN 1520-9210. doi: 10.1109/TMM.2013.2241043.

[13] Kuan-Ta Chen, Chi-Jui Chang, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. Quadrant of euphoria: A crowdsourcing platform for qoe assessment. *Netwrk. Mag. of Global Internetwkg.*, 24(2):28–35, March 2010. ISSN 0890-8044. doi: 10.1109/MNET.2010.5430141. URL http://dx.doi.org/10.1109/MNET.2010.5430141.

[14] Kuan-Ta Chen, Chen-Chi Wu, Yu-Chun Chang, and Chin-Laung Lei. A crowdsourceable qoe evaluation framework for multimedia content. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 491–500, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-608-3. doi: 10.1145/1631272.1631339. URL http://doi.acm.org/10.1145/1631272.1631339.

[15] Ujwal Gadiraju, Sebastian Möller, Martin Nöllenburg, Dietmar Saupe, Sebastian Egger-Lampl, Daniel Archambault, and Brian Fisher. *Crowdsourcing Versus the Laboratory: Towards Human-Centered Experiments Using the Crowd*, pages 6–26. 01 2017. ISBN 978-3-319-66434-7. doi: 10.1007/978-3-319-66435-4_2.

[16] 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*, pages 1–6, Oct 2015.

[17] O. Belmoukadam, T. Spetebroot, and C. Barakat. Acqua: A user friendly platform for lightweight network monitoring and qoe forecasting. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 88–93, Feb 2019. doi: 10.1109/ICIN.2019.8685878.

[18] Meteor: Free internet speed & app performance test, 2018. `https://meteor.opensignal.com/`.

[19] Paul Schmitt, Francesco Bronzino, Renata Teixeira, Tithi Chattopadhyay, and Nick Feamster. Enhancing transparency: Internet video quality inference from network traffic. In *TPRC 46: The 46th Research Conference on Communication, Information and Internet Policy 2018.*, 2018.

[20] Tobias Hoßfeld, Raimund Schatz, Ernst Biersack, and Louis Plissonneau. *Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience*, pages 264–301. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36784-7.

[21] RTR-Netz open dataset, 2017. `https://www.netztest.at/en/Opendata`.

[22] MobiPerf, 2018. `https://www.measurementlab.net/tests/mobiperf/`.

[23] M. H. Mazhar and Z. Shafiq. Real-time video quality of experience monitoring for https and quic. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018.

[24] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. A machine learning approach to classifying youtube qoe based on encrypted network traffic. *Multimedia Tools and Applications*, 76, 2017.

[25] Muhammad Jawad Khokhar, Nawfal Abbassi Saber, Thierry Spetebroot, and Chadi 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, pages 31–36, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5056-3.

[26] M. J. Khokhar, T. Spetebroot, and C. Barakat. An online sampling approach for controlled experimentation and qoe modeling. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[27] Muhammad Jawad Khokhar, Nawfal Abbassi Saber, Thierry Spetebroot, and Chadi Barakat. An intelligent sampling framework for controlled experimentation and qoe modeling. *Computer Networks*, 147:246 – 261, 2018. ISSN 1389-1286. doi: https://doi.org/10.1016/j.comnet.2018.10.011. URL `http://www.sciencedirect.com/science/article/pii/S1389128618303700`.

[28] Muhammad Jawad Khokhar, Thierry Spetebroot, and Chadi Barakat. A methodology for performance benchmarking of mobile networks for internet video streaming. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '18, pages 217–225, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5960-3. doi: 10.1145/3242102.3242128. URL `http://doi.acm.org/10.1145/3242102.3242128`.

[29] Muhammad Jawad Khokhar, Thibaut Ehlinger, and Chadi Barakat. From Network Traffic Measurements to QoE for Internet Video. In *IFIP Networking Conference 2019*, Warsaw, Poland, May 2019. URL `https://hal.inria.fr/hal-02074570`.

[30] Youtube QoE datasets. `http://www-sop.inria.fr/diana/acqua/datasets`.

[31] Code and Datasets, 2018. `https://github.com/mjawadak/youtubeapi`.

[32] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp), 1998.

[33] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications, 2003.

[34] P. Juluri, V. Tamarapalli, and D. Medhi. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys Tutorials*, 18(1):401–418, Firstquarter 2016. ISSN 1553-877X. doi: 10.1109/COMST.2015.2401424.

[35] Christos George Bampis and Alan C. Bovik. Learning to predict streaming video qoe: Distortions, rebuffering and memory. *CoRR*, abs/1703.00633, 2017.

[36] Y. Sani, A. Mauthe, and C. Edwards. Adaptive bitrate selection: A survey. *IEEE Communications Surveys Tutorials*, 19(4):2985–3014, Fourthquarter 2017. ISSN 1553-877X. doi: 10.1109/COMST.2017.2725241.

[37] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, MMSys '11, pages 157–168, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0518-1. doi: 10.1145/1943552.1943574. URL `http://doi.acm.org/10.1145/1943552.1943574`.

[38] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*, pages 1–6, July 2012. doi: 10.1109/QoMEX.2012.6263849.

[39] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 211–224, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1705-4.

[40] K. D. Singh, Y. Hadjadj-Aoul, and G. Rubino. Quality of experience estimation for adaptive http/tcp video streaming using h.264/avc. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 127–131, Jan 2012. doi: 10.1109/CCNC.2012.6181070.

[41] I. Sodagar. The mpeg-dash standard for multimedia streaming over the internet. *IEEE MultiMedia*, 18(4):62–67, April 2011. ISSN 1070-986X. doi: 10.1109/MMUL.2011.71.

[42] IIS smooth streaming transport protocol, 2014. `http://www.adobe.com/products/httpdynamicstreaming/`.

[43] Adobe HTTP Dynamic Streaming, 2014. `http://www.adobe.com/products/httpdynamicstreaming/`.

[44] HTTP Live Streaming, Informational In- ternet Draft, 2011. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-06`.

[45] DASH Industry Forum, 2019. `https://dashif.org/`.

[46] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 187–198, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2836-4. doi: 10.1145/2619239.2626296. URL `http://doi.acm.org/10.1145/2619239.2626296`.

[47] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016. doi: 10.1109/INFOCOM.2016.7524428.

[48] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. *IEEE/ACM Trans. Netw.*, 22(1):326–340, February 2014. ISSN 1063-6692. doi: 10.1109/TNET.2013.2291681. URL `https://doi.org/10.1109/TNET.2013.2291681`.

[49] Guibin Tian and Yong Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 109–120, New

York, NY, USA, 2012. ACM. ISBN 978-1-4503-1775-7. doi: 10.1145/2413176.2413190. URL http://doi.acm.org/10.1145/2413176.2413190.

[50] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of ACM SIGCOMM*, pages 325–338, New York, 2015. ISBN 978-1-4503-3542-3.

[51] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: Auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 44–58, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5567-4. doi: 10.1145/3230543.3230558. URL http://doi.acm.org/10.1145/3230543.3230558.

[52] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 197–210, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4653-5.

[53] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella. D-dash: A deep q-learning framework for dash video streaming. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):703–718, Dec 2017. ISSN 2332-7731. doi: 10.1109/TCCN.2017.2755007.

[54] P. Brooks and B. Hestnes. User measures of quality of experience: why being objective and quantitative is important. *IEEE Network*, 24(2):8–13, March 2010. ISSN 0890-8044. doi: 10.1109/MNET.2010.5430138.

[55] Mohammed Alreshoodi and John Woods. Survey on qoe qos correlation models for multimedia services. *International Journal of Distributed and Parallel systems*, 4, 06 2013. doi: 10.5121/ijdps.2013.4305.

[56] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. de Veciana. Video quality assessment on mobile devices: Subjective, behavioral and objective studies. *IEEE Journal of Selected Topics in Signal Processing*, 6(6):652–671, Oct 2012. ISSN 1932-4553. doi: 10.1109/JSTSP.2012.2212417.

[57] ITU-T. Subjective video quality assessment methods for multimedia applications. *ITU-T Recommendation P.910*, 2008.

[58] S. Aroussi and A. Mellouk. Survey on machine learning-based qoe-qos correlation models. In *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 200–204, April 2014.

[59] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 339–350, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2056-6. doi: 10.1145/2486001.2486025.

[60] Y. Qi and M. Dai. The effect of frame freezing and frame skipping on video quality. In *2006 International Conference on Intelligent Information Hiding and Multimedia*, pages 423–426, Dec 2006. doi: 10.1109/IIH-MSP.2006.265032.

[61] Tahir Nawaz Minhas and Markus Fiedler. Impact of disturbance locations on video quality of experience. EuroITV 2011 Workshop: Quality of Experience for Multimedia Content Sharing, 2011.

[62] Q. Huynh-Thu and M. Ghanbari. Temporal aspect of perceived quality in mobile video broadcasting. *IEEE Transactions on Broadcasting*, 54(3):641–651, Sep. 2008. ISSN 0018-9316. doi: 10.1109/TBC.2008.2001246.

[63] R. Serral-Gracià, E. Cerqueira, M. Curado, M. Yannuzzi, E. Monteiro, and X. Masip-Bruin. An overview of quality of experience measurement challenges for video applications in ip networks. In Evgeny Osipov, Andreas Kassler, Thomas Michael Bohnert, and Xavier Masip-Bruin, editors, *Wired/Wireless Internet Communications*, pages 252–263, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[64] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.

[65] Toward A Practical Perceptual Video Quality Metric, 2016. `https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652`.

[66] Eero P. Simoncelli Zhou Wang. Reduced-reference image quality assessment using a wavelet-domain natural image statistic model, 2005. URL `https://doi.org/10.1117/12.597306`.

[67] Weisi Lin and C.-C. Jay Kuo. Perceptual visual quality metrics: A survey. *Journal of Visual Communication and Image Representation*, 22(4):297 – 312, 2011. ISSN 1047-3203. doi: https://doi.org/10.1016/j.jvcir.2011.01.005. URL `http://www.sciencedirect.com/science/article/pii/S1047320311000204`.

[68] B. Lewcio, B. Belmudez, T. Enghardt, and S. Möller. On the way to high-quality video calls in future mobile networks. In *2011 Third International Workshop on Quality of Multimedia Experience*, pages 43–48, Sep. 2011. doi: 10.1109/QoMEX.2011.6065710.

[69] B. Lewcio, B. Belmudez, A. Mehmood, M. Wältermann, and S. Möller. Video quality in next generation mobile networks — perception of time-varying transmission. In *2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, pages 1–6, May 2011. doi: 10.1109/CQR.2011.5996089.

[70] M. Zink, J. Schmitt, and R. Steinmetz. Layer-encoded video in scalable adaptive streaming. *IEEE Transactions on Multimedia*, 7(1):75–84, Feb 2005. ISSN 1520-9210. doi: 10.1109/TMM.2004.840595.

[71] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 463–472, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0616-4. doi: 10.1145/2072298.2072359. URL `http://doi.acm.org/10.1145/2072298.2072359`.

[72] S. Khirman and P. Henriksen. Relationship between quality-of-service and quality-of-experience for public internet service. In *3rd Passive Active Measurement Workshop*, 2002.

[73] M. Fiedler, T. Hossfeld, and P. Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41, March 2010. ISSN 0890-8044. doi: 10.1109/MNET.2010.5430142.

[74] YouTube API, 2019. `https://developers.google.com/youtube/`.

[75] WebRequest, 2018. `https://developer.chrome.com/extensions/webRequest`.

[76] Microworkers, 2019. `https://www.microworkers.com/`.

[77] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz. Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks. In *2015 European Conference on Networks and Communications (EuCNC)*, pages 239–243, June 2015. doi: 10.1109/EuCNC.2015.7194076.

[78] M. Seufert, N. Wehner, F. Wamser, P. Casas, A. D'Alconzo, and P. Tran-Gia. Unsupervised qoe field study for mobile youtube video streaming with yomoapp. In *QoMEX*, pages 1–6, May 2017. doi: 10.1109/QoMEX.2017.7965688.

[79] P. Juluri, L. Plissonneau, and D. Medhi. Pytomo: A tool for analyzing playback quality of youtube videos. In *2011 23rd International Teletraffic Congress (ITC)*, pages 304–305, Sep. 2011.

[80] Craig Gutterman, Katherine Guo, Sarthak Arora, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman. Requet: Real-time qoe detection for encrypted youtube traffic. In *Proc. ACM MMSys'19*, 2019.

[81] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 359–370, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1419-0. doi: 10.1145/2342356.2342431. URL `http://doi.acm.org/10.1145/2342356.2342431`.

[82] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. *SIGCOMM CCR*, 41(4), August 2011.

[83] Liu Yitong, Shen Yun, Mao Yinian, Liu Jing, Lin Qi, and Yang Dacheng. A study on quality of experience for adaptive streaming service. In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 682–686, June 2013. doi: 10.1109/ICCW.2013.6649320.

[84] Michalis Katsarakis, Renata Cruz Teixeira, Maria Papadopouli, and Vassilis Christophides. Towards a causal analysis of video qoe from network and application qos. In *Proceedings of the Internet QoE'16*, pages 31–36. ACM, 2016. ISBN 978-1-4503-4425-8.

[85] Tobias Hoßfeld, Raimund Schatz, and Udo R. Krieger. Qoe of youtube video streaming for current internet transport protocols. In Kai Fischbach and Udo R. Krieger, editors, *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, pages 136–150, Cham, 2014. Springer International Publishing.

[86] Ashkan Nikravesh, David Ke Hong, Qi Alfred Chen, Harsha V. Madhyastha, and Z. Morley Mao. Qoe inference without application control. In *Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*, Internet-QoE '16, pages 19–24, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4425-8. doi: 2940136.2940145. URL `http://doi.acm.org/2940136.2940145`.

[87] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura. emimic: Estimating http-based video qoe metrics from encrypted network traffic. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–8, June 2018. doi: 10.23919/TMA.2018.8506519.

[88] Muhammad Zubair Shafiq, Jeffrey Erman, Lusheng Ji, Alex X. Liu, Jeffrey Pang, and Jia 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*, pages 367–379, NY, USA, 2014. ACM. ISBN 978-1-4503-2789-3.

[89] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui 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, 2016. USENIX Association. ISBN 978-1-931971-29-4.

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

[91] Byron C. Wallace, Kevin Small, Carla E. Brodley, and Thomas A. Trikalinos. Active learning for biomedical citation screening. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 173–182, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0055-1.

[92] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988. ISSN 1573-0565. doi: 10.1007/BF00116828. URL `http://dx.doi.org/10.1007/BF00116828`.

[93] Fabian Pedregosa. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, Oct 2011.

[94] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.*, 5, 2004. URL `http://dl.acm.org/citation.cfm?id=1005332.1016791`.

[95] Luigi Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *SIGCOMM Comput. Commun. Rev.*, 27(1):31–41, January 1997. ISSN 0146-4833. doi: 10.1145/251007.251012. URL `http://doi.acm.org/10.1145/251007.251012`.

[96] ITU. Subjective audiovisual quality assessment methods for multimedia applications. *ITU-T Recommendation P.911*, 1998.

[97] Vaneet Aggarwal, Emir Halepovic, Jeffrey Pang, Shobha Venkataraman, and He 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, pages 18:1–18:6, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2742-8.

[98] Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Trans. Speech Lang. Process.*, 6(3):3:1–3:24, April 2010. ISSN 1550-4875.

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

[100] Vlado Menkovski, Georgios Exarchakos, and Antonio Liotta. Tackling the sheer scale of subjective qoe. In Luigi Atzori, Jaime Delgado, and Daniele Giusto, editors, *Mobile Multimedia Communications*, pages 1–15, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30419-4.

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

[102] Raoufehsadat Hashemian, Niklas Carlsson, Diwakar Krishnamurthy, and Martin Arlitt. Iris: Iterative and intelligent experiment selection. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, ICPE '17, pages 143–154, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4404-3.

[103] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–39, Jan 2014. ISSN 2162-237X. doi: 10.1109/TNNLS.2012.2236570.

[104] H. Chang, C. Hsu, T. Hobfeld, and K. Chen. Active learning for crowdsourced qoe modeling. *IEEE Transactions on Multimedia*, pages 1–1, 2018. ISSN 1520-9210.

[105] P. Casas, M. Seufert, N. Wehner, A. Schwind, and F. Wamser. Enhancing machine learning based qoe prediction by ensemble models. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1642–1647, July 2018.

[106] V. Vasilev, J. Leguay, S. Paris, L. Maggi, and M. Debbah. Predicting qoe factors with machine learning. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[107] Johan De Vriendt, Danny De Vleeschauwer, and Dave C. Robinson. Qoe model for video delivered over an lte network using http adaptive streaming. *Bell Labs Technical Journal*, 18(4):45–62, 2014. ISSN 1538-7305. doi: 10.1002/bltj.21645.

[108] Florian Wamser, Pedro Casas, Michael Seufert, Christian Moldovan, Phuoc Tran-Gia, and Tobias Hossfeld. Modeling the youtube stack: From packets to quality of experience. *Computer Networks*, 109(Part 2), 2016. ISSN 1389-1286.

[109] VP9 Codec, 2018. `https://www.webmproject.org/vp9/`.

[110] Linux Traffic Control, 2018. `http://lartc.org/`.

[111] R2Lab, 2017. `https://r2lab.inria.fr/index.md`.

[112] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. Youtube qoe estimation based on the analysis of encrypted network traffic using machine learning. In *2016 IEEE Globecom Workshops*, Dec 2016. doi: 10.1109/GLOCOMW.2016.7849088.

[113] X. Che, B. Ip, and L. Lin. A survey of current youtube video characteristics. *IEEE MultiMedia*, 22(2):56–63, Apr 2015. ISSN 1070-986X. doi: 10.1109/MMUL.2015.34.

[114] Ashkan Nikravesh, David R. Choffnes, Ethan Katz-Bassett, Z. Morley Mao, and Matt Welsh. Mobile network performance from user devices: A longitudinal, multidimensional analysis. In *Proceedings of PAM 2014*, pages 12–22, 2014. ISBN 978-3-319-04917-5.

[115] Sanae Rosen, Hongyi Yao, Ashkan Nikravesh, Yunhan Jia, David Choffnes, and Z. Morley Mao. Demo: Mapping global mobile performance trends with mobilyzer and mobiperf. In *Proceedings of MobiSys '14*, pages 353–353, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2793-0.

[116] Grid5000, 2018. `https://www.grid5000.fr`.

[117] AWS EC2, 2018. `https://aws.amazon.com/ec2/`.

[118] Werner Robitza et al. HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software. In *9th ACM Multimedia Systems Conference*, Amsterdam, 2018. ISBN 9781450351928. doi: 10.1145/3204949.3208124.

[119] ITU P.1203 Python implementation, 2018. `https://github.com/itu-p1203/itu-p1203`.

[120] Ericsson QoE report, 2017. https://www.ericsson.com/en/ericsson-technology-review/archive/2017/video-qoe-leveraging-standards-to-meet-rising-user-expectations.

[121] Python Scikit-learn library, 2018. http://scikit-learn.org/.

[122] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/mniha16.html.

[123] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063, Cambridge, MA, USA, 1999. MIT Press. URL http://dl.acm.org/citation.cfm?id=3009657.3009806.

[124] Reinforcement Learning Course, 2017. http://rail.eecs.berkeley.edu/deeprlcourse-fa17/.

[125] Blockchain as a Service. https://www.investopedia.com/terms/b/blockchainasaservice-baas.asp.