Quality of Experience estimation of encrypted video streaming by using machine learning methods

Oršolić, Irena

Doctoral thesis / Disertacija

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:474921

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-28**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Irena Oršolić

QUALITY OF EXPERIENCE ESTIMATION OF ENCRYPTED VIDEO STREAMING BY USING MACHINE LEARNING METHODS

DOCTORAL THESIS

Zagreb, 2020

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Irena Oršolić

QUALITY OF EXPERIENCE ESTIMATION OF ENCRYPTED VIDEO STREAMING BY USING MACHINE LEARNING METHODS

DOCTORAL THESIS

Supervisor:
Associate Professor Lea Skorin-Kapov, PhD

Zagreb, 2020

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Irena Oršolić

PROCJENA ISKUSTVENE KVALITETE ZA STRUJANJE ŠIFRIRANOGA VIDEA PRIMJENOM METODA STROJNOGA UČENJA

DOKTORSKI RAD

Mentorica:
izv. prof. dr. sc. Lea Skorin-Kapov

Zagreb, 2020.

The doctoral thesis was completed at the University of Zagreb Faculty of Electrical Engineering and Computing, Department of Telecommunications.

Supervisor: Associate Professor Lea Skorin-Kapov, PhD

This thesis has: 182 pages

Thesis number: _____

About the Supervisor

Lea Skorin-Kapov is Associate Professor and head of the Multimedia Quality of Experience Research Lab (MUEXLab) at the University of Zagreb Faculty of Electrical Engineering and Computing. Her research interests include Quality of Experience assessment and modelling of advanced multimedia applications, QoE monitoring of encrypted video traffic, and cross-layer negotiation and management of QoS/QoE in networks. She teaches courses at bachelor, masters, and doctoral levels dealing with multimedia services, heuristic optimisation methods, and communication networks.

She attended elementary school and high school in Canada and the USA, after which she received her Dipl.-Ing., M.S., and Ph.D. degrees in Telecommunications from the Faculty of Electrical Engineering and Computing (FER) at the University of Zagreb, Croatia, in 2001, 2004, and 2007, respectively. From 2001–2009 she was employed in the Research and Development Center of Ericsson Nikola Tesla d.d. (ETK), Zagreb, Croatia, doing research on QoS signalling, negotiation, and adaptation for multimedia services. From 2002-2009 she was also an adjunct teaching and research assistant at the Department of Telecommunications, FER, University of Zagreb.

Since 2010 she has been employed at the Department of Telecommunications, FER, University of Zagreb (as Assistant professor from 2010-2016 and as Associate professor from 2016 until the present). She is currently involved in a number of research and industry funded projects, and is principal investigator for the project "Modeling and Monitoring QoE for Immersive 5G-Enabled Multimedia Services" funded by the Croatian Science Foundation. She is a member of the Croatian Centre of Research Excellence for Data Science and Advanced Cooperative Systems, and a member of the Management Board of the Center for Artificial Intelligence at FER. She previously participated in EU COST Action IC1304 ACROSS (Autonomous Control for a Reliable Internet of Services), COST Action IC1003 (European Network on Quality of Experience in Multimedia Systems and Services, QUALINET), Celtic-Plus project "Quality of Experience Estimators in Networks" (QuEEN), FP7 OpenIoT, and EU SIIF project "ICT-based Competence Network for Innovative Services for Persons with Complex Communication Needs".

She has published over 100 scientific papers, has served on numerous conference and workshop TPCs including ACM Multimedia, ACM Multimedia Systems, QoMEX, IEEE ICC, IEEE Infocom, ITC, and others, and has served as Program co-chair of the IEEE flagship Region 8 conference EUROCON, and the International Conference on Quality of Multimedia Experience (QoMEX 2017). She is on the editorial board of IEEE Transactions on Network and Service Management and Springer's Multimedia Systems journal, and has served as Guest Editor for the IEEE Journal of Selected Topics in Signal Processing, and ACM Transactions on

Multimedia Computing, Communications, and Applications. She acts as reviewer for top rated journals including IEEE/ACM Transactions on Networking, IEEE Communications Magazine, IEEE Computer, Springer Multimedia Tools and Applications, IEEE Surveys and Tutorials. She was a contributing author to ETSI TS 103 294 “Speech and multimedia Transmission Quality (STQ); Quality of Experience: A Monitoring Architecture”.

She is currently serving as Chapter chair of the IEEE Communications Society —Croatia Chapter.

O mentoru

Lea Skorin-Kapov je izvanredna profesorica na Zavodu za telekomunikacije Fakulteta elektrotehnike i računarstva (FER) Sveučilišta u Zagrebu, gdje radi od 2010. godine. Voditeljica je istraživačkog laboratorija Multimedia Quality of Experience Research Lab (MUEXLab). Njezino glavno područje istraživačkog interesa jest modeliranje iskustvene kvalitete višemedijskih usluga, praćenje iskustvene kvalitete te mehanizmi upravljanja i optimizacije kvalitete usluge/iskustvene kvalitete u mrežama. Podučava na FER-ovom preddiplomskom, diplomskom i doktorskom studiju o višemedijskim uslugama i komunikacijama, heurističkim metodama optimizacije i komunikacijskim mrežama.

Pohađala je osnovnu i srednju školu u Kanadi i SAD-u, nakon čega je diplomirala 2001. godine, magistrirala 2004. godine, te doktorirala 2007. godine na Sveučilištu u Zagrebu FER, smjer telekomunikacije i informatika. Od 2001.-2009. godine radila je u Istraživačkom odjelu tvrtke Ericsson Nikole Tesle d.d. (ETK), Zagreb u jedinici za Istraživanje i razvoj, gdje se bavila istraživanjem područja signalizacije, pregovaranja i prilagodbe kvalitete usluge za napredne višemedijske usluge u telekomunikacijskim mrežama nove generacije. Godine 2002. izabrana je u naslovno suradničko zvanje mlađeg asistenta na FER-u.

2010. g. izabrana je u znanstveno-nastavno zvanje docent na FER-u, a 2016. g. u znanstveno-nastavno zvanje izvanrednog profesora. Zaposlena je na Zavodu za telekomunikacije. Uključena je u razne istraživačke i industrijske projekte. Trenutno vodi istraživački projekt “Modeliranje i praćenje iskustvene kvalitete imerzivnih višemedijskih usluga u 5G mrežama” kojeg financira Hrvatska zaklada za znanost. Članice je Znanstvenog centra izvrsnosti za znanost o podatcima i kooperativne sustave, te Upravnog odbora Centra za umjetnu inteligenciju na FER-u. Sudjelovala je kao članica Upravnog odbora COST IC1304 ACROSS (Autonomous Control for a Reliable Internet of Services), te kao članica Upravnog odbora COST IC1003 QUALINET (European Network on Quality of Experience in Multimedia Systems and Services), na Celtic-Plus projektu “Quality of Experience Estimators in Networks” (QuEEN), FP7 projektu OpenIoT i EU SIIF projektu “Kompetencijska mreža zasnovana na ICT-u za inovativne usluge namijenjene osobama sa složenim komunikacijskim potrebama”.

Autorica je i koautorica više od 100 znanstvenih radova u časopisima, zbornicima radova s međunarodnih konferencija i knjigama. Djelovala je i djeluje kao članica tehničkog odbora za brojne međunarodne konferencije i radionice, uključujući: ACM Multimedia, ACM Multimedia Systems, QoMEX, IEEE ICC, IEEE Infocom, ITC, i ostale. Djelovala je kao supredsjedatelj programa za IEEE konferenciju EUROCON i konferenciju QoMEX 2017. Članica je uredničkog odbora časopisa IEEE Transactions on Network and Service Management i Springerovog časopisa Multimedia Systems, te je sudjelovala kao gostujući urednik za časopise IEEE Journal of Selected Topics in Signal Processing i ACM Transactions on Multimedia Computing, Communications, and Applications. Recenzira radove za brojne časopise, uključujući IEEE/ACM Transactions on Networking, IEEE Communications Magazine, IEEE Computer, Springer Multimedia Tools and Applications, IEEE Surveys and Tutorials. Sudjelovala je kao koautor u pisanju tehničke specifikacije ETSI TS 103 294 “Speech and multimedia Transmission Quality (STQ); Quality of Experience: A Monitoring Architecture”.

Trenutno obnaša funkciju predsjednice Odjela za komunikacije, hrvatske sekcije IEEE.

Acknowledgements

This thesis would have not seen the light of day without the support of the people around me. I would like to take this opportunity to thank them.

First and foremost, I am eternally grateful to my advisor, professor Lea Skorin-Kapov, for giving me the opportunity to join her research team and pursue a PhD. Lea has been my mentor since the third year of undergraduate study, and I have great admiration for her commitment to working with students, doing research, managing projects. She was deeply involved in every phase of this research, more than one should expect from their advisor, but still leaving enough space for me to figure things out on my own. Thank you for all the hard work you put into this, for being at my disposal whenever I got stuck, for securing interesting projects for us to work on, thank you for being a motivator, role model, and a kind and understanding person.

A lot of the sections of this thesis are related to research conducted in the scope of the project QoMoVid (*"QoE monitoring solutions for mobile OTT video streaming"*), funded by Ericsson Nikola Tesla. This thesis greatly benefited from datasets collected in this cooperation and all the fruitful discussions that gave me a better understanding of problems found in practice. This thesis' research was also funded by the Croatian Science Foundation, in the scope of the *"Young Researchers' Career Development Project – Training of New Doctoral Students"* (DOK-2015-10-6013), Q-MANIC (*"Cooperative Quality of Experience Management for Interactive Cloud-Based Multimedia Applications in Mobile Networks"*, UIP-2014-09-5605) and Q-MERSIVE (*"Modeling and Monitoring QoE for Immersive 5G-Enabled Multimedia Services"*, IP-2019-04-9793), which secured the funding for my salary, the equipment I needed, and enabled me to present our research to wider audiences.

On the topics related to this thesis, besides Lea, I most closely worked with professor Mirko Sužnjević and Ivan Bartolec. I have learned a great deal from both, and working with them has helped me better define the objectives of this thesis. I am thankful to Mirko for sometimes shattering the dreams of my inner idealist, and teaching me to find a balance between idealism and pragmatism. Without that, many things would have remained imperfect and unfinished. This way, they are just imperfect, as all things in life should be. You have taught me well, Obi-Wan. Ivan has started working on his PhD two years after me, on complimentary topics. It is an advantage to have someone as devoted to work together and reflect on each other's ideas. Thank you, and I hope our collaboration continues to be as rewarding as it has been so far.

My deepest appreciation goes to professors Maja Matijašević, Sonja Grgić, and Tobias Hoßfeld, who reviewed this thesis and provided me with many valuable comments. Professors Matijašević and Hoßfeld were also familiar with earlier stages of our research and gave a lot of interesting suggestions on the way.

I would like to thank everybody at the Department of Telecommunications, who have made

FER feel like second home. Working towards a PhD was surely made easier being surrounded by people who understand and share both the joy and the frustration of it. Being among scientists engaged in different research areas has been an opportunity to broaden my views, to learn, and grow. It has also been fun just hanging out – deep conversations over lunch to karaoke parties in the evening (and then later deep conversations again). I would like to give my special thanks to Sara Vlahović, who has been my office mate for the longest time, and has become a friend I can talk to about anything, anytime.

Last but not least, to my family and friends, who are always there for me, and had nothing but understanding when I was too busy to offer the same level of support. I would like to thank my mum Ljiljana, for always encouraging me to follow my heart and for being that person in my life with an unbreakable spirit and contagious positivity. Thanks to my sister Daniela, for all the what's-the-worst-that-could-happens and just-do-it-nows that (sometimes) helped me stop overanalysing everything. Finally, to my dad Marko, who shared the love for all things technical and always stimulated my curiosity growing up. Although not here for this journey, he certainly helped shape me into a person who would be willing to take it up in the first place.

Abstract

With the amount of global network traffic steadily increasing, mainly due to video streaming services, network operators are faced with the challenge of efficiently managing their resources while meeting customer demands and expectations. A prerequisite for such Quality-of-Experience–driven (QoE) network traffic management is the monitoring and inference of application-level performance in terms of video Key Performance Indicators (KPIs) that directly influence end-user QoE. Given the persistent adoption of end-to-end encryption, operators lack direct insights into video quality metrics such as start-up delays, resolutions, or stalling events, which are needed to adequately estimate QoE and drive resource management decisions. This research has been motivated by the challenge to devise an approach for the estimation of QoE/KPIs from encrypted traffic, where we recognised machine learning (ML) methods as a promising way forward, and a fundamental part of the methodology.

In this thesis, we present a generic methodology for training of ML–based models for the estimation of QoE and application-level KPIs of adaptive video streaming services, applicable in the context of traffic encryption. The methodology is embodied in the form of a conceptual framework which demonstrates the key methodological steps involved in developing an in-network QoE/KPI monitoring solution, including model training, model deployment, and model re-evaluation and adaptation. The methodology is evaluated through six studies, conducted over the course of four years, involving YouTube video on demand, resulting in models for session-level QoE/KPI estimation focused on both Android and iOS, models for near real-time KPI estimation, analysis of cross-platform and cross-network model applicability, analysis of methods for automated model re-evaluation and adaptation, and the analysis of the impact of the inclusion of application-level data (possibly shared by a service provider) on the performance of QoE/KPI estimation models.

The key contribution of the thesis is a methodology that identifies relevant KPIs to be used as prediction targets, identifies relevant network traffic features obtainable on IP-level, and describes the procedures of model training, evaluation, re-evaluation, and adaptation, thus covering processes that are a prerequisite for actual model deployment. The practical focus of the thesis has been on YouTube, which is one of the most prominent video streaming services today. In that context, a unique and valuable contribution are also the models for YouTube QoE/KPI estimation focused on mobile platforms, applicable for both TCP and QUIC traffic, and employing a standardised ITU-T P.1203 model for the calculation of QoE. Moreover, the thesis presents models that include application-level context data as additional predictors, thus contributing to motivation for resolving existing issues standing in the way to QoE-centric cooperation among actors involved in the service delivery chain.

Keywords: video streaming, Quality of Experience, QoE estimation, machine learning

Procjena iskustvene kvalitete za strujanje šifriranoga videa primjenom metoda strojnoga učenja

Uslijed kontinuiranog rasta količine mrežnog prometa na globalnoj razini, čemu najviše doprinose usluge strujanja videa, mrežni operatori su suočeni s izazovom učinkovitog upravljanja mrežnim resursima, uz ispunjavanje zahtjeva i očekivanja krajnjih korisnika. Preduvjet za takvo upravljanje mrežnim prometom, vođeno iskustvenom kvalitetom (engl. *Quality of Experience*, QoE) je mogućnost praćenja i procjene performansi usluga strujanja videa na aplikacijskoj razini u obliku ključnih indikatora performansi (engl. *Key Performance Indicator*, KPI) koji direktno utječu na QoE krajnjih korisnika.

Prijašnjih godina, rješenja za praćenje QoE-a u mreži su se oslanjala na inspekciju paketa (engl. *Deep Packet Inspection*, DPI) za dobivanje informacija o kvaliteti videa, čitanjem podataka iz zaglavlja aplikacijske razine. Budući da je mrežni promet povezan s ovim uslugama sve češće šifriran s kraja na kraj, mrežni operatori više nemaju direktan uvid u mjere kvalitete strujanja, poput trajanja inicijalnog učitavanja (engl. *initial delay*, *start-up delay*), rezolucije videa ili trajanja zastoja u reprodukciji (engl. *stalling*, *re-buffering*), koje su nužne za adekvatnu procjenu QoE-a i upravljanje mrežnim resursima vođeno QoE-em. Pored šifriranja podataka s aplikacijske razine, kao što ga pruža protokol TLS (*Transport Layer Security*), sve se češće viđa korištenje protokola QUIC, koji uvodi šifriranje s kraja na kraj na transportnoj razini.

Danas, pretpostavljajući ograničenu ili nepostojeću razmjenu informacija između mrežnih operatora i pružatelja usluga strujanja videa, mrežni operatori se najčešće oslanjaju isključivo na praćenje statističkih karakteristika prometa za procjenu QoE-a i analizu uzroka potencijalnih degradacija kvalitete. Popularne usluge strujanja videa implementiraju paradigmu prilagodljivog strujanja putem protokola HTTP (engl. *HTTP adaptive streaming*, HAS), primarno kako bi smanjile pojavu zastoja u reprodukciji, kao glavnog uzroka degradacije QoE-a, time dodatno otežavajući procjenu aplikacijskih KPI-eva i ukupnog QoE-a u mreži.

Promatrajući podatke vezane za performanse usluge duž lanca dostave sadržaja, mogu se uočiti podaci na različitim razinama: u mreži se mogu pratiti različiti parametri kvalitete usluge (engl. *Quality of Service*, QoS) poput prosječne propusnosti, gubitka paketa i sl., dok se u klijentskoj aplikaciji mogu mjeriti KPI-evi aplikacijske razine (npr. trajanje zastoja u reprodukciji, trajanje početnog kašnjenja, rezolucija videa). QoS parametri mrežne razine utječu na aplikacijske KPI-eve, koji pak utječu na QoE na korisničkoj razini. Odnos između QoE-a i mjerenih (aplikacijskih i mrežnih) parametara opisuju QoE modeli. Utjecaj aplikacijskih KPI-eva na QoE je dobro proučen posljednjih godina, a 2017. godine je objavljen standardizirani QoE model za HAS u ITU-T preporuci P.1203. Za razliku od toga, odnos između parametara mrežne razine i aplikacijskih KPI-eva ili QoE-a nije općepoznat i dobro istražen, a postoji značajan interes industrije za njegovim razotkrivanjem. Ovo istraživanje je motivirano izazovom razvoja

metodologije za procjenu QoE-a/KPI-eva iz šifriranog mrežnog prometa, gdje su metode strojnoga učenja (engl. *machine learning*) prepoznate kao obećavajući put prema rješenju, te su stoga osnova metodologije ovoga rada.

U ovoj disertaciji prezentirana je generička metodologija za treniranje modela za procjenu QoE-a i aplikacijskih KPI-eva za usluge strujanja videa, temeljena na strojnom učenju, a primjenjiva u kontekstu šifriranog prometa. Metodologija je uobličena u konceptualni radni okvir (engl. *framework*) kroz koji su demonstrirani ključni koraci razvoja rješenja za praćenje QoE-a/KPI-eva u mreži, što uključuje treniranje modela, ugradnju modela u mreži te re-evaluaciju i prilagodbu modela. Metodologija je iterativno unaprjeđivana i validirana kroz šest studija provedenih tokom perioda od četiri godine, čiji je praktični fokus usluga YouTube i strujanje videa na zahtjev (engl. *Video on Demand*).

Kroz sve studije provlači se isti generički pristup za prikupljanje i obradu podataka, te treniranje modela. Najprije je potrebno prikupiti skup podataka koji uključuje podatke o performansama strujanja niza videa na testnim uređajima i istovremeno snimljeni mrežni promet. Podaci moraju biti prikupljeni u raznolikim mrežnim uvjetima, rezultirajući tako i s raznolikim degradacijama QoE-a. Iz mrežnog prometa računaju se njegove statističke značajke, a iz prikupljenih podataka o performansama strujanja na aplikacijskoj razini računaju se KPI-evi i/ili QoE (npr. temeljem postojećeg modela iz ITU-T preporuke P.1203). Izračunate statistike mrežnog prometa se označavaju stvarnim vrijednostima KPI-eva/QoE-a, a algoritmi strojnog učenja zatim uočavaju odnose između navedenih vrijednosti te grade model. Takav model, ako se ugradi u mrežu, može procijeniti KPI-eve i/ili QoE isključivo iz značajki mrežnog prometa, bez znanja o stvarnim performansama aplikacije na strani krajnjih korisnika.

Premda se oslanjaju na isti generički pristup, svaka studija pak ima svoje metodološke specifičnosti, te je rezultirala novim i jedinstvenim skupovima podataka, modelima za procjenu QoE-a/KPI-eva, razvijenim alatima i saznanjima. Sumarni cilj studija bio je odgovoriti na sljedeća istraživačka pitanja:

- Koji su ključni metodološki koraci uključeni u razvoj rješenja za praćenje QoE-a/KPI-eva HAS usluga u mreži?
- Mogu li se QoE/KPI-evi procijeniti na temelju značajki mrežnog prometa na razini pojedinog videa?
- Mogu li se KPI-evi procijeniti na temelju značajki mrežnog prometa u skoro stvarnom vremenu, tj. na razini kratkih vremenskih intervala?
- Mogu li se modeli za procjenu QoE-a/KPI-eva trenirati tako da istovremeno budu primjenjivi za više slučajeva uporabe?
- Mogu li se modeli za procjenu QoE-a/KPI-eva prilagoditi bez ponovnog treniranja

novog modela u slučaju promjene u podacima kroz vrijeme?

- Koliko se mogu poboljšati performanse modela u slučaju da su u mreži dostupne određene kontekstne informacije te se iste iskoriste prilikom procjene QoE-a/KPI-eva?

Prve tri studije su fokusirane na procjenu QoE-a/KPI-eva na razini pojedinog videa. Studija S1 (2016. – 2017.) predstavlja prvi korak u razvoju metodologije i modela za procjenu QoE-a na razini pojedinačnog videa za uslugu strujanja YouTube videa, temeljeno na analizi šifriranog mrežnog prometa. Na osnovu definirane metodologije, razvijeni su alati za prikupljanje i obradu podataka, prikupljen je skup podataka vezan za strujanje YouTube videa u Web pregledniku Android uređaja u laboratorijskom okruženju s emuliranim varijabilnim uvjetima širine pojasa, te trenirani modeli za procjenu QoE-a. Ključni rezultat ove studije jest definiran generički pristup za treniranje modela za procjenu QoE-a/KPI-eva za HAS usluge te implementacija prototipa kojim se potvrdila izvedivost predloženog pristupa i izgradio temelj za daljnje istraživanje.

Nastavno na rezultate Studije S1, pojavu novih istraživačkih saznanja u literaturi i uočene promjene u načinu dostave video sadržaja, Studija S2 (2017. – 2018.) je u mnogim aspektima unaprijedila metodologiju definiranu u Studiji S1. Konkretno, razvijene su nove metode i alati za prikupljanje podataka o performansama strujanja na testnom uređaju prilikom korištenja YouTube aplikacije, koju stvarni korisnici usluge u pravilu koriste češće nego Web preglednik. Definiran je i novi skup značajki mrežnog prometa, koje se mogu izračunati isključivo iz IP-zaglavlja (bez korištenja podataka s transportnog sloja), što je bilo nužno s YouTube-ovim prelaskom s korištenja protokola TCP na protokol QUIC. Studija S2 je također unaprijedila način izračuna QoE-a, budući da se oslanja na korištenje standardiziranog QoE modela za HAS objavljenog 2017. godine u ITU-T preporuci P.1203, za razliku od vlastitih modela korištenih u Studiji S1, temeljenih na literaturi dostupnoj u vrijeme provođenja studije. Studija je polučila modele za procjenu QoE-a/KPI-eva trenirane na podacima prikupljenim na Android uređaju, uz nabrojana metodološka unaprjeđenja.

Fokus Studije S3 (2017. – 2018.) je bio na prilagodbi metodologije kako bi se ostvarila mogućnost procjene QoE-a/KPI-eva YouTube video strujanja na iOS uređajima. S obzirom na složenost i iscrpnost prikupljanja podataka potrebnih za treniranje modela, bilo bi poželjno kada bi se modeli trenirani u jednom slučaju uporabe mogli primijeniti u drugom slučaju uporabe. Tako je jedan od ciljeva Studije S3 bio testirati primjenjivost modela treniranih na podacima prikupljenim na Android platformi za procjenu QoE-a/KPI-eva YouTube video strujanja na iOS-u. Rezultati pokazuju da su performanse modela snižene, ali u određenim uvjetima i dalje prihvatljive, ako je ključno smanjiti iscrpnost procesa prikupljanja podataka i složenost rješenja za praćenje QoE-a/KPI-eva u mreži. Dodatno, a vezano za lakše prikupljanje podataka, i Studija S2 i Studija S3 uključuju testiranje modela treniranih u laboratorijskim uvjetima na podacima

prikupljenim u mobilnoj mreži, što je dalo obećavajuće rezultate.

Studija S4 (2018. – 2019.) bavi se proučavanjem utjecaja uključivanja određenih kontekstnih informacija kao dodatnih značajki na performanse modela. Čak i modeli obogaćeni minimalnom količinom informacija s aplikacijske razine znatno pridonose točnijoj procjeni QoE-a/KPI-eva, te stoga rezultati motiviraju suradnju između mrežnih operatora i pružatelja usluga u svrhu postizanja zajedničkih ciljeva upravljanja QoE-em. Izdvaja se informacija o brzini kodiranja videa, koja znatno doprinosi performansama modela, a razmjena iste ne bi narušavala privatnost korisnika.

Za razliku od ranijih studija koje imaju za cilj procjenu QoE-a/KPI-eva na razini pojedinog videa, cilj Studije S5 (2019. – 2020.) bio je razviti metodologiju i modele za procjenu KPI-eva u skoro stvarnom vremenu, tj. na kratkim vremenskim intervalima. Za razliku od procjene QoE-a/KPI-eva na razini pojedinog videa, što je korisno mrežnim operatorima za dugoročno planiranje i dimenzioniranje mreže, praćenje KPI-eva u kratkim vremenskim intervalima omogućuje prilagodbu upravljanja mrežnim prometom unutar mreže u skoro stvarnom vremenu. Predložena metodologija je prezentirana na slučaju strujanja YouTube videa na Android platformi u laboratorijskoj bežičnoj mreži te su trenirani modeli postigli vrlo visoke performanse.

Konačno, Studija S6 (2019. – 2020.) zaokružuje svo dotad stečeno znanje o problemu procjene QoE-a/KPI-eva i adresira dodatne povezane probleme. Jedan od važnih problema kojima se bavi ova studija jest re-evaluacija i adaptacija modela, kao odgovor na potencijalne promjene u logici dostave video sadržaja. Razmotreni su pristupi iz domene prilagodljivog učenja te je predložen pristup za automatsku re-evaluaciju i prilagodbu modela. Drugi važan problem adresiran u Studiji S6 je mogućnost treniranja generičkih modela, kako bi se smanjila iscrpnost prikupljanja podataka za treniranje modela i, u konačnici, složenost izvođenja modela u mreži. Problema generalizacije modela dotakle su se i ranije studije (Studija S2 i Studija S3) u kojima su modeli trenirani na podacima prikupljenim u jednim uvjetima testirani na podacima prikupljenim u drugim uvjetima (različite platforme, različite pristupne mreže). Studija S6 se nadovezuje na navedene prethodne studije, dodatno uključujući generičke modele trenirane na kombiniranim podacima prikupljenim na različitim platformama.

Šest provedenih studija opisanih u ovoj disertaciji rezultiralo je velikim brojem modela koji procjenjuju QoE/KPI-eve usluge YouTube u mreži na razini pojedinog videa ili na razini kratkih vremenskih intervala, koristeći pritom podatke prikupljene na Android i iOS platformi, u laboratorijskoj bežičnoj mreži i u mobilnoj mreži. Sa svakom studijom unaprijeđena je sveukupna metodologija, kako bi konačno poprimila oblik radnog okvira opisanog u ovoj disertaciji. Radni okvir i metodologija koju predstavlja su generički, a pojedinačne komponente radnog okvira su, u tom smislu, demonstrirane u praksi kroz slučajeve uporabe koji se tiču usluge YouTube.

Istraživanjem opisanim u ovom radu ostvaren je znanstveni doprinos koji se sastoji od sljedećih elemenata:

- Metodologija za treniranje modela za procjenu iskustvene kvalitete i aplikacijskih ključnih indikatora performansi prilagodljivog strujanja videa putem protokola HTTP bazirana na analizi šifriranog mrežnog prometa uporabom metoda strojnog učenja, te podrška za automatsku re-evaluaciju i prilagodbu takvih modela uslijed promjena u okolini.
- Modeli za procjenu iskustvene kvalitete i aplikacijskih ključnih indikatora performansi usluga prilagodljivog video strujanja iz perspektive mreže, zasnovani na strojnom učenju i definiranoj metodologiji.
- Evaluacija utjecaja različitih količina kontekstnih informacija s aplikacijske razine na performanse modela za procjenu iskustvene kvalitete i ključnih indikatora performansi iz perspektive mreže.

Ključne riječi: strujanje videa, iskustvena kvaliteta, procjena iskustvene kvalitete, strojno učenje

Contents

Chapter 1

Introduction

This chapter motivates the thesis research by presenting the background and stating the challenges in the domain. The chapter also emphasises the challenges addressed in the scope of this thesis and summarises the key research contributions. Section 1.1 provides a brief overview of the domain, while the following sections define the problem addressed in this thesis (Section 1.2), describe the applied research methodology and research scope (Section 1.3), and highlight the novel scientific contributions of the thesis (Section 1.4).

1.1 Background and motivation

The Internet is constantly growing and evolving. Users' needs and expectations change with the arrival of new technologies, with service and network providers working to meet these expectations in terms of both Quality of Service (QoS) and Quality of Experience (QoE). Given the rapid growth of global mobile Internet traffic and the number of connected devices [1], actors in the service delivery chain are faced with the challenge of providing a satisfactory experience to end users, while efficiently using limited resources. Such QoE–aware service and network management calls for understanding and monitoring of key factors that impact the end users' perceived service quality and experience. This has resulted in extensive QoE–related research conducted over the past decade, with the networking community increasingly aiming to introduce QoE-awareness into network management cycles [2].

In the massive amounts of traffic passing through global mobile networks, the greatest share belongs to video content, mainly originating from popular video streaming services and social networks, such as YouTube, Netflix, Amazon Prime, Facebook, and Twitch [3]. According to Cisco's VNI white paper [1], video accounted for 59% of global mobile traffic in 2017, and the share is expected to grow to 79% by 2022. Recent numbers, published in the Ericsson Mobility Report [4], state that 63% of global mobile traffic was video in 2019. The same report is slightly more conservative in its forecasts, predicting the share to grow to 76% by

2025. Nevertheless, despite the emergence of new and network-intensive apps such as Virtual and Augmented Reality (VR/AR) and cloud gaming, especially in the context of 5G networks, video originating from video streaming platforms (either as Video on Demand or live, real-time video) is expected to dominate network traffic in the foreseeable future.

Aiming to manage service performance with respect to constrained resources, nowadays, most video streaming services implement the HTTP Adaptive Streaming (HAS) paradigm, commonly in compliance with standards MPEG-DASH (Dynamic Adaptive Streaming over HTTP) [5] and HLS (HTTP Live Streaming) [6]. The main idea behind HAS is to enable dynamic adaptation of video quality (e.g., in terms of bitrate and resolution) according to variable network conditions, so as to primarily avoid playback stalling and ensure a smooth playout experience. This is achieved by storing short chunks of video content on the server, available in different quality levels, and running an adaptation algorithm on the client side responsible for estimating network conditions and requesting each video chunk accordingly. The aforementioned standards define methods and formats, while each compliant service, on top of that, defines its own used parameters (e.g., chunk size), and adaptation algorithm (e.g., client-side buffering strategy).

While streaming service providers can easily obtain information about the quality of the provided service through feedback from the client application, information about the quality of streams passing through networks and application-layer performance on client devices (e.g., in terms of stalling, playback resolution) is for the most part not available to network operators, as the streams are usually encrypted. Up until recently, Web encryption efforts were primarily focused on highly sensitive information, while information such as the quality of the video being downloaded could have been retrieved in the network by using Deep Packet Inspection (DPI). Today, with the rise of privacy awareness, we are witnessing encryption in all major video streaming services as well. For example, Google reports that, as of late 2019, all YouTube content is delivered via HTTPS [7]. While services such as Netflix, Twitch, and Facebook Watch use TLS/TCP, YouTube has mostly switched from using TLS/TCP to using QUIC/UDP [8], which was designed to be secure and incorporates TLS by default. Compared to TLS/TCP content delivery, where only application-level data is encrypted and QoS parameters such as packet loss or retransmissions can be detected in the network, in QUIC/UDP such parameters are also encrypted, providing even less information about potential problems encountered during the streaming session. However, it is worth mentioning that there are standardisation efforts within IETF for a new version of HTTP, HTTP/3, that will employ QUIC by default, making the Web inherently secure [9].

As monitoring the QoE in the network is a prerequisite for QoE–aware resource allocation and network planning, there is a lot of interest from the industry for finding solutions that estimate QoE from data that is accessible. Figure 1.1 shows data available to different actors in

the service delivery chain. The client application and the server, administrated by the Over-the-Top (OTT) video streaming service provider, have access to information about played quality levels (resolution, bitrate, codec), initial delays, and other objective quality metrics. While OTTs by definition rely on the Internet for reliably delivering their content, it is common that network providers have little or no insight into the quality of the content being delivered or potential problems with the delivery that may be mitigated in the network. From the network perspective, only IP-level data is readable, and, possibly, context data from the access network, such as information about user equipment. Thus, the question arises as to whether QoE and application-level key performance indicators (KPI), such as initial delay duration, stalling, and played quality levels, can be estimated from the patterns in encrypted network traffic, or, more specifically, packet sizes and interarrival times.

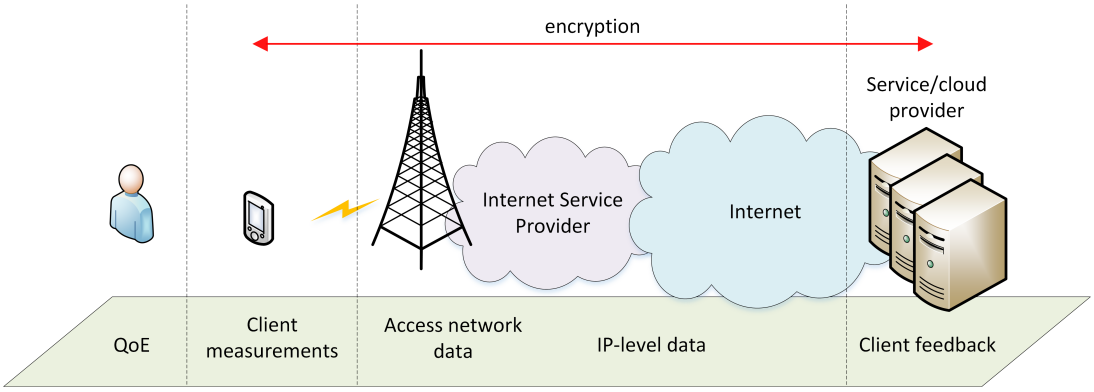


Figure 1.1: Data accessible by different actors in the service delivery chain. While the service provider typically has access to information about service performance at the client, such data is usually not accessible from the network perspective due to traffic encryption.

1.2 Problem definition

Due to the adoption of traffic encryption and the dynamics introduced by HAS, it has become challenging for network operators to monitor service performance at the application level, which is crucial when aiming to estimate the end users' QoE. The challenge is further complicated by the wide variety of video streaming services accessed from different platforms, apps, via different access networks, using different protocols, etc. (Figure 1.2). As HAS dominates global Internet traffic, there is a lot of interest from the industry to tackle these challenges. The initial hypothesis, suggested by literature available at the time when this research was started, was that solutions could be found that infer application performance in terms of KPIs and/or overall QoE, solely based on the analysis of encrypted network traffic. A generalised view of the idea is given in Figure 1.3 (adapted from [10]). These solutions require the instrumentation of measurement tools used to collect relevant application- and network-layer data. Based on collected data, models that map traffic patterns to QoE/KPIs are built either by manual analysis

or using machine learning (ML) techniques. In-network measurements (e.g., radio stats, flow data) can be provided in real time to deployed models to estimate the desired QoE/KPI metrics. A network operator can thus monitor QoE impairments, generate alerts regarding severe issues, aggregate QoE data to create trending reports, and finally use collected data for deriving the root causes of QoE degradations. At the same time, new application-level ground truth data can be collected to re-evaluate models over time and detect the need for a model update. Such updates may be necessary in response to changes such as those related to OTT service quality adaptation logic, or changes in underlying protocols (e.g., move from TLS/TCP to QUIC/UDP).

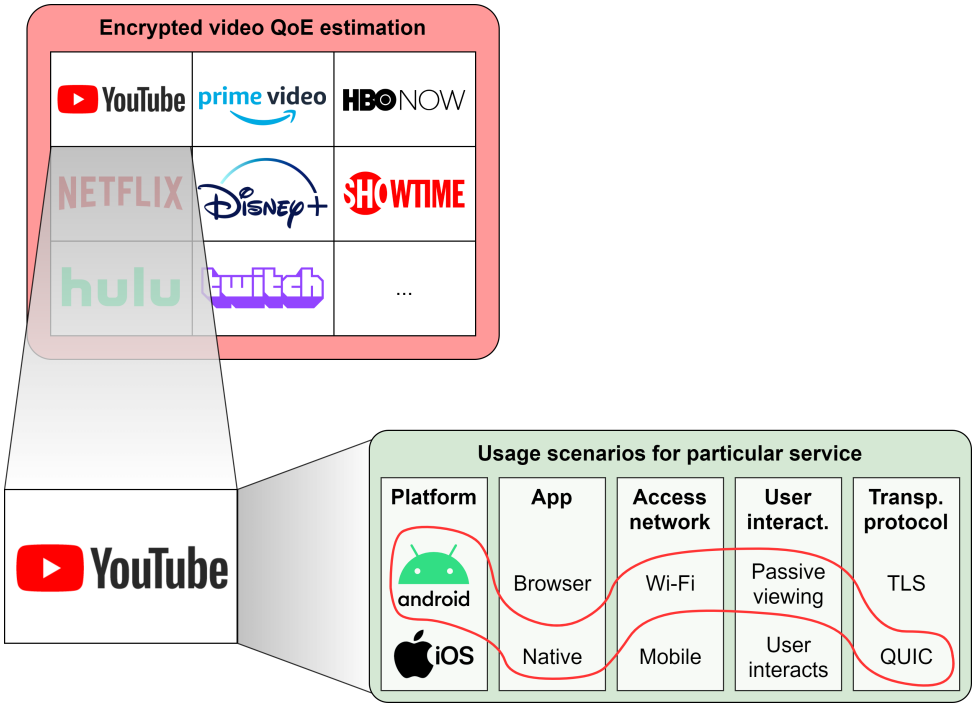


Figure 1.2: Various use-cases may differ in terms of service implementation, and thus in network traffic patterns for video delivery platforms. Consequently, the estimation of QoE/KPIs for these use-cases may need to be addressed separately. An example of a use-case is indicated, where YouTube is accessed via the native application on Android platform with QUIC used for transport, while connected to a Wi-Fi network and the user is passively viewing the content.

At the beginning of the research described in this thesis, several studies emerged [10, 11], including our initial work on the topic [12, 13], proving the fundamental hypothesis to be correct on isolated use-cases – application-level KPIs and QoE can indeed be inferred from the statistical features of encrypted network traffic. These solutions were dependant on information from TCP headers, which soon made them obsolete in the context of the move to using the QUIC protocol, introduced by Google when accessing YouTube, and expected to be introduced by many other video streaming services as well with the introduction of HTTP/3. The approaches also relied on custom video streaming applications for measuring the ground-truth performance, which do not necessarily reflect the behaviour of applications that are of interest to network operators (e.g., YouTube, Netflix). However, the high-level methodology proposed

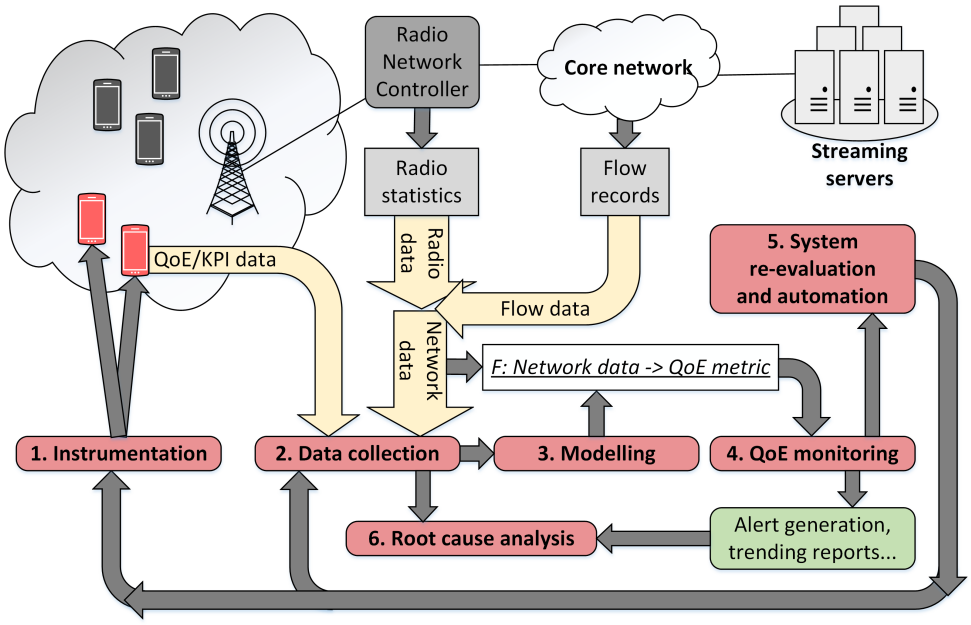


Figure 1.3: Generic service QoE/KPI monitoring approach in the context of traffic encryption (figure adapted from [10]).

in these papers is still valid and provides a solid ground for exploring challenges on the way to deploying timely, robust, and easily adjustable QoE/KPI monitoring solutions for HAS services.

Ground-truth data collection (referring to application-layer KPIs available at the client) is not as challenging for some use-cases as it is for others. Some video services provide APIs for streaming their content and simultaneously measuring the streaming performance (e.g., YouTube). While this seems like the simplest solution to obtain the ground truth, when available, such APIs may not be updated regularly and do not necessarily mimic the behaviour of the actual application of interest [14]. Some services offer performance information in the form of a video overlay (e.g., YouTube, Twitch viewed in the browser), that can be enabled by the user. However, there are services that offer neither (e.g., Netflix). Moreover, collection of ground-truth data is further complicated by different platforms, as some of the methods for obtaining the data may be applicable only on certain types of devices (e.g., Android or iOS).

Use-cases differing in HAS service, used device/platform, and access network, have shown to differ in the availability of application-level data and/or in network-level data (e.g., different protocols being used), thus requiring different data processing and consequently different ML model training methods. One of the questions related to data variety is to what extent data can be collected, formatted, and processed in a generic way across different use-cases. From an ML modelling perspective, adding information beyond IP-level statistics that has a stronger correlation with QoE/KPIs, upon availability, can improve the estimation models' performance. For example, having access to network-level data from higher layers (e.g., retransmission information on the transport layer) may result in better performing models. Also, models could potentially be improved with the availability of additional data used as predictors, such as radio

statistics or data provided by the OTT service provider. An open research question is the extent to which QoE/KPI estimation model performance could be improved with additional data provided, beyond only derived IP-level traffic statistics.

Another set of open research questions in this domain is related to the deployment and practical utilisation of trained models in the network. Currently available literature only provides information about promising model performance but does not quantify potential benefits for ISPs to deploy such models. Initial research has focused on models that estimate QoE/KPIs on a per-video session level, but a similar approach can be taken also to estimate KPIs across fixed time windows (e.g., near real-time, such as on a per-second basis, or across longer windows, such as per-minute). An important consideration for deploying such models in the network is the amount of resources needed to capture relevant probe data, calculate network traffic features, and run the model instance on flows passing through the network. The amount of required resources in some cases may potentially be reduced by using less complex ML models, while only slightly reducing prediction accuracy. Also, as HAS services may change their adaptation behaviour and used protocols over time, there is a need for change detection and automated model adaptation. This might call for a new measurement campaign and model re-training, or more advanced ML techniques used to adapt existing models to changes that occurred.

Based on a literature review at the beginning of this research, but also reflecting on the new findings in state of the art during the course of this research, the following six research questions have been identified to be addressed in the scope of this thesis:

- **RQ1**: What are the key methodological steps involved in developing an in-network QoE/KPI monitoring solution for HAS?
- **RQ2**: Can QoE/KPIs be estimated based on IP-level network traffic features on a per-video basis?
- **RQ3**: Can KPIs be estimated based on IP-level network traffic features in near real-time, i.e., in short time intervals?
- **RQ4**: Can QoE/KPI estimation models be trained so as to address multiple use-cases (e.g., differing in used client platform or access network) at once?
- **RQ5**: Can QoE/KPI models be updated without training a new model from scratch in the event of concept drift (i.e., when the relationships between traffic patterns and QoE/KPIs incorporated in models change, due to changes in service delivery)?
- **RQ6**: What potential gains may be achieved in terms of model performance if certain application-level context data is available (e.g., if OTTs shared data such as video encoding bitrates for a video being streamed, thus cooperating on in-network QoE-management)?

We further provide the mapping of these research questions to the objectives (Section 1.3) and contributions (Section 1.4) of the thesis in Figure 1.4.

1.3 Method of solution and scope

After having identified open issues and research questions to be addressed, we have defined the research methodology and steps as depicted in Figure 1.5. In **Step 1**, based on related work on QoE influence factors for HAS, we have identified KPIs that are of interest to network operators. Identified KPIs are to be predicted by the ML models devised in the following steps. Based on related work addressing a variety of traffic classification problems, we have defined network traffic features to be investigated as potentially beneficial predictors in QoE/KPI estimation. As part of Step 1, we have also inspected the available options in terms of use-cases to be addressed and data that is accessible in these use-cases.

Step 2 was aimed at specifying a methodology for data collection, data processing, and training of QoE/KPI estimation models. The methodology is considered to be generic, i.e., applicable for a variety of use-cases and as such has been presented through a generic and flexible framework for in-network video streaming QoE/KPI estimation. The methodology and the framework have been iteratively refined with new findings obtained in Steps 3 and 4.

The following phase (**Step 3**) included the design and development of ground-truth data collection tools, both for Android and iOS devices, the design of a testbed to be used for running laboratory experiments, and execution of data collection campaigns. Step 3, together with Step 4, is part of an iterative process executed through six studies (S1 – S6). The studies differ in terms of used ground-truth data collection tools, network conditions emulated in the laboratory environment, addressed platforms, and specifics of the collected dataset. The practical focus across all studies, with regards to developed instrumentation and trained models was on YouTube streamed to mobile devices. However, the methodology devised in Step 2 remains generic, with YouTube serving as a use-case for comprehensive evaluation purposes. Studies are considered to be incremental, either addressing an additional dimension of the QoE/KPI estimation problem, or adapting the methodology as a response to service behaviour changes. The exact details regarding each study are given in Section 4.2.

In **Step 4**, we have developed scripts for data processing and preparation for ML model training. This includes the calculation of target KPIs/QoE, and the extraction of network traffic features from traffic captures. We have analysed the data collected in the previous step, and trained models for QoE/KPI estimation. Depending on the study (S1 – S6), the data was recombined so as to answer certain research questions, features were subset, multiple models trained using different ML algorithms, and finally, validated and compared. The specifics of these tasks vary across the six studies, with details given in Section 4.2.

The activities defined in the research methodology depicted with Figure 1.5 are aimed towards meeting the following objectives:

- **O1**: Develop a methodology for data collection, data processing, and training of QoE/KPI estimation models (Chapter 4), and conduct an initial feasibility study (Section 5.1.).
- **O2**: Train and evaluate the models for session-level QoE/KPI estimation (Sections 5.1. – 5.3., Section 7.2.).
- **O3**: Train and evaluate the models for near real-time KPI estimation (Section 5.4.).
- **O4**: Evaluate the extent to which the models can be trained so as to be applicable for multiple platforms and access networks (Section 7.1.).
- **O5**: Evaluate the impact of concept drift on models' performance and propose a model adaptation method (Section 7.3.).
- **O6**: Evaluate the impact of selected context information added as additional predictors onto model performance (Chapter 6.).

While objective O1 is addressed in Steps 1 and 2 of the research methodology, the rest of the objectives are addressed iteratively through six studies, described in Section 4.2

1.4 Summary of contributions

The contributions of this thesis may be summarised as follows:

- **C1**: A methodology for training of machine learning based models for the estimation of QoE and application-level KPIs of HTTP adaptive video streaming services based on the analysis of encrypted network traffic, and support for the automated re-evaluation and adaptation of such models in light of changing environments [12, 13, 15, 16].
- **C2**: Machine-learning based models for in-network estimation of QoE and KPIs of adaptive video streaming services derived by using the proposed methodology [14, 17].
- **C3**: Evaluation of the impact of different amounts of application-level context information on the performance of in-network QoE and KPI estimation models [18].

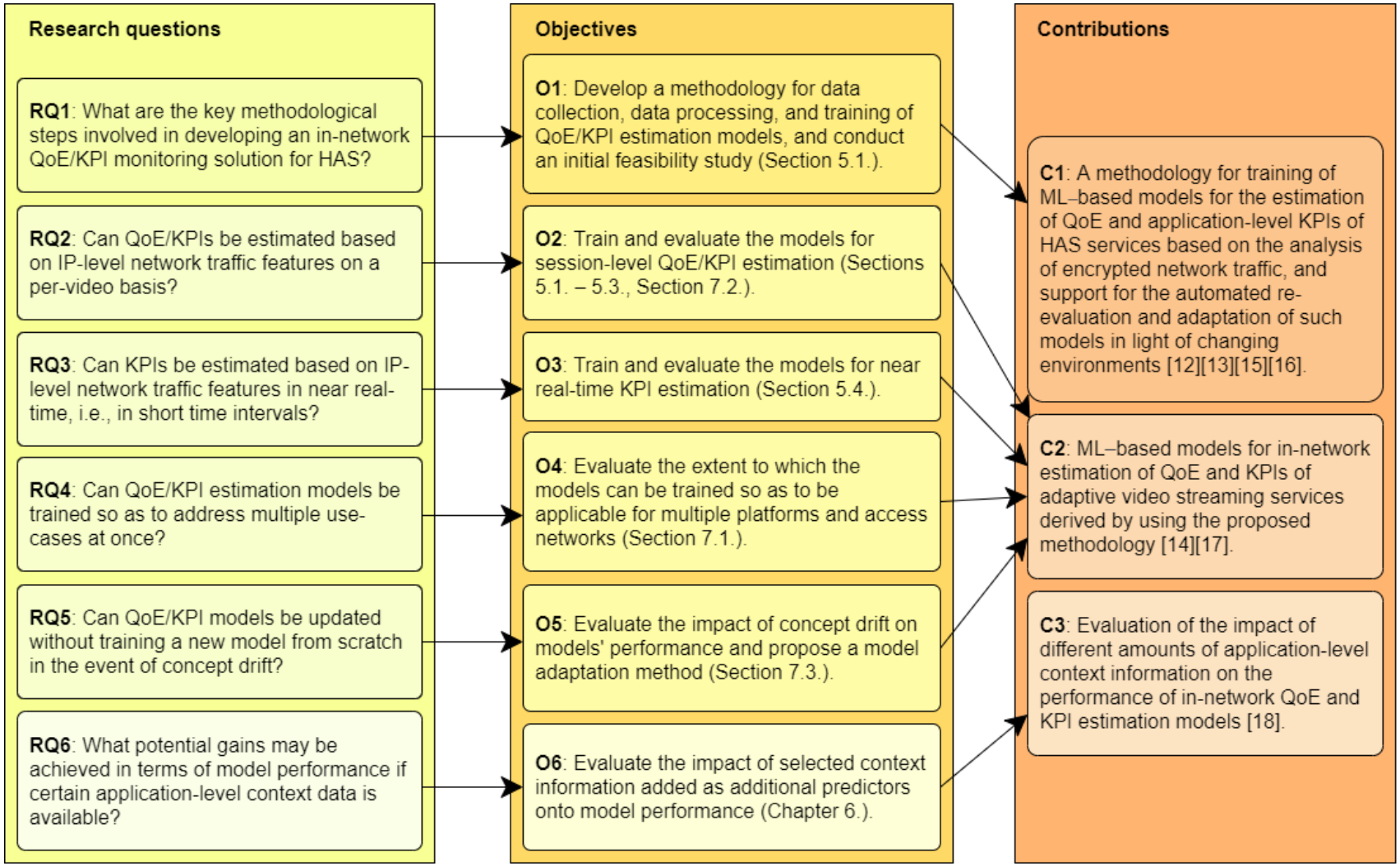


Figure 1.4: The mapping of addressed research questions, research objectives, and contributions of the thesis. Thesis author's publications corresponding to each contribution are listed.

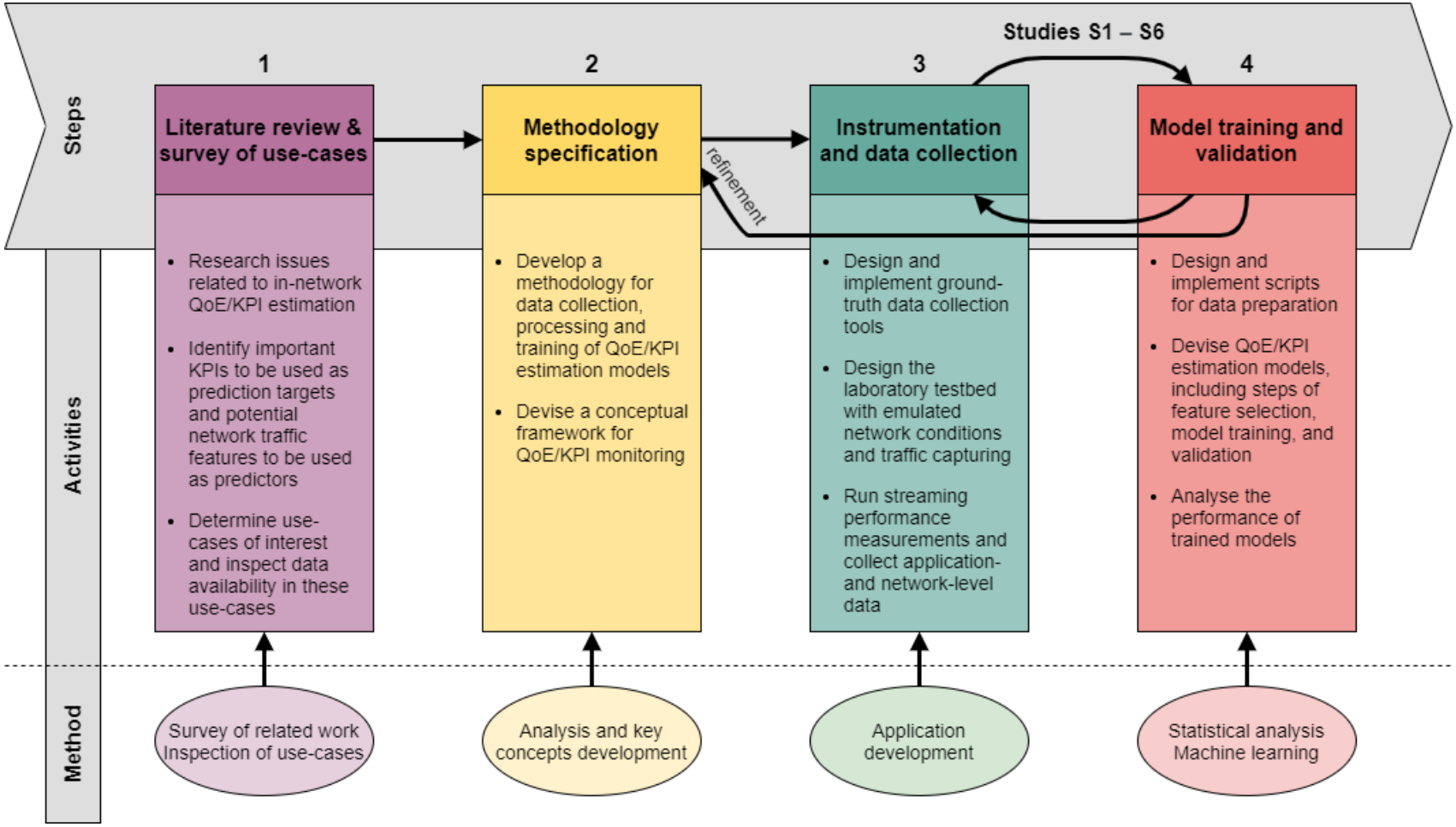


Figure 1.5: Research methodology as applied, in four steps. Steps 1 and 2 address Objective O1, while Steps 3 and 4 are iterated over six studies addressing Objectives O2 – O6. The studies are described in Section 4.2.

1.5 Thesis structure

The thesis is structured as follows. After the high-level introduction, **Chapter 2** provides deeper insight into the domain of adaptive video streaming, briefly summarising the overall evolution of video streaming, and then focusing on HAS and standards, what is today considered to be state-of-the-art technology. **Chapter 3** introduces Quality of Experience (QoE), the key definitions and methods, generic QoE management procedures, QoE in the context of HAS, and finally reviews state-of-the-art in the domain of QoE monitoring, which is the focus of this thesis. The chapter includes a survey of related work on client-side and network-centric QoE monitoring solutions, as well as work on potential exploitation of such monitoring solutions for QoE-aware application and network management.

In the context of challenges identified in the state-of-the-art review, **Chapter 4** presents a conceptual framework for in-network QoE/KPI estimation that brings together all key methodological steps involved in developing an in-network QoE/KPI monitoring solution for HAS. While the framework is generic, to reach the objectives of the thesis, the methodology it represents was applied on a number of use-cases concerning YouTube. The chapter provides a summary of dataset collection campaigns, and six conducted studies aimed at reaching these objectives. **Chapter 5** is focused on ML–based QoE/KPI estimation models, presents the specific methodologies of separate studies in terms of the defined classification targets, calculated network traffic features used as predictors, and model training. This is followed by the analysis of the performance of the aforementioned models. Sections 5.1 – 5.3 are focused on session-level (per-video) QoE/KPI estimation, and correspond to three studies addressing different use-cases. Challenges related to near–real-time (per-second) KPI estimation, on the other hand, are addressed in Section 5.4.

Chapter 6 describes the investigation of the impact of inclusion of context data on QoE/KPI estimation, which may potentially, given its availability, improve the performance of QoE/KPI estimation models. A number of questions arise when considering the deployment of QoE/KPI monitoring solutions in the network, some of which are addressed in **Chapter 7**. The chapter investigates the possibility of training models able to address multiple use-cases (namely the YouTube video streaming service delivered across two different platforms, Android and iOS), the benefits of including near–real-time KPI predictions as additional predictors for session-level QoE/KPI estimation, and the possibility of model re-evaluation and adaptation. Finally, **Chapter 8** concludes the thesis, summarises the contributions and limitations of the thesis research, and presents an outlook in the form of research challenges that are yet to be addressed.

Chapter 2

Adaptive video streaming

The most prominent OTT video streaming services at the time of writing this thesis all rely on HAS technology [19]. Aiming to monitor and estimate QoE/KPIs of these services from a network provider perspective, it is necessary to understand the technology these services are based upon. This chapter first provides a brief overview of the evolution of OTT video streaming in Section 2.1, and then focuses on HAS. The paradigm is explained in Section 2.2, while Section 2.3 describes standards currently in broad use. Section 2.4 summarises the chapter.

2.1 The evolution of video streaming services

The way people access video content has changed dramatically over the past few decades. While technologies required for OTT video streaming, in terms of coding standards, streaming container formats, and transport protocols, were available in the 1990s, given the relatively low bandwidth and high latency conditions common across networks, there was a lack of network resources to support massive video stream transports. Video streaming then was limited to demos of what was, at the time, cutting-edge technology. With the revolution in Internet access introduced by broadband at the beginning of the new millennium, video streaming services such as Dailymotion, YouTube, Google video, Crunchyroll started emerging, changing the way we access and consume video content. These video streaming services surely played a big part in the shift towards user-generated Web, known as Web 2.0 [20].

Shortly thereafter, a significant share of the population had access to such services, using their PCs, but also smartphones. As the network evolution brought more bandwidth and lower latency, and devices (especially smartphones) became more powerful, the desire to access streaming video content grew among end users, as did their expectations regarding the streaming performance. To reduce initial delays and bandwidth consumption, streaming providers such as YouTube and Netflix made use of Content Delivery Networks (CDN) to bring the content closer to the users. At first, they relied on services offered by CDN providers such as

Limelight and Akamai [20, 21, 22], but as they grew, they built their own CDNs, e.g., Google CDN and Netflix Open Connect [23, 24]. What makes caching more effective is the fact that the popularity of videos follows a Zipf-like distribution, i.e., only a small percentage of videos is highly popular [25, 26]. Today, video streaming is deeply entrenched in our lives – we watch, create, share, stream live content, and interact with streamers in real-time. Still, there is room for growth, reaching more places in the world, supporting more devices, streaming 360-degree video, 2K, 4K, 8K video, consequently leading to increased bandwidth requirements, thus pushing the networks' limits even further.

When looking into different types of video streaming, there is a clear distinction between Video on Demand (VoD) and live (also referred to as real-time) streaming. In VoD, the content is pre-stored on a server and ready to be streamed to a requesting client, while with live streaming, the content is being delivered as it is being created. Compared to VoD, where network latency and jitter are not necessarily a problem, as the content can be buffered at the client, live video streaming is very sensitive to such conditions. This is especially true for certain types of live streaming applications, such as streaming of sports events, where service providers aim to minimise broadcaster-to-client latency. Given their differences, VoD and live video streaming in their beginnings relied on different underlying technologies.

Live OTT video streams were initially most commonly delivered using the Real-time Transport Protocol (RTP) [27]. RTP provides end-to-end transport of real-time data over multicast or unicast network services. It is independent of the underlying transport protocol, but most commonly used over unreliable transport (UDP). Session monitoring is not implemented in RTP as such, but rather appointed to a paired protocol, RTCP (RTP Control Protocol). A typical live streaming scenario that relies on RTP also includes HTTP for obtaining the session description and RTSP (Real Time Streaming Protocol) [28, 29] for session management. Once the session description is retrieved from the streaming server, the client initiates a session setup in which the parameters of the session are established, and finally the content is delivered via RTP/UDP.

VoD, on the other hand, as the content is already available in its entirety, started out in a download-and-play manner. It is clear that this resulted in long initial delays. Dealing with that issue is considered to be the key for YouTube's early success. YouTube enabled video playback to start before the content was completely downloaded, by relying on Adobe's progressive download technology [20]. The videos were stored in Adobe Flash Video format and played in the Adobe Flash Player. In this case, as the content can be buffered and latency is not critical, reliable transport can be used (TCP). A typical scenario includes a user requesting information about content locations via HTTP, and then progressively retrieving segments of the video. The playback starts after a predefined amount of data is buffered, and HTTP is used to retrieve the remaining segments of the video.

Following progressive download, the next evolutionary step for VoD was the widespread

adoption of HTTP adaptive streaming (also referred to as adaptive bitrate streaming). This meant that the video was not only downloaded in segments while it was being played, but also that the segments were downloaded in quality representations matching the current dynamic network conditions and buffer status [30]. The technology was available for Adobe Flash, but later flourished alongside HTML5, with many services in the Internet replacing application-specific protocols and plug-ins with more generic ones to reduce the technical overhead and compatibility issues. To date, a number of standards for HAS have emerged, but the ones most commonly used today are MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [5] and Apple's HTTP Live Streaming (HLS) [31]. Major video streaming providers today (e.g., YouTube, Twitch, Facebook Watch) implement their services in compliance with these two standards, not only for VoD, but also for live streaming, with the key difference being that the amount of the video content in the buffer is kept significantly lower for live streaming.

Another recent change is related to used transport protocols. HTTP was traditionally delivered via TCP, long before video streaming, and a lot of video streaming providers still use HTTP/TCP for delivering HAS services. However, TCP was not designed with video streaming services in mind, and there are multiple aspects in which it may be adapted to more efficiently deliver HAS, while ensuring the reliability offered by TCP. Aiming to reduce latency and enable stream multiplexing, Google designed the Quick UDP Internet Connections (QUIC) protocol, and initiated its standardisation within IETF in 2016. Standardisation efforts yielded numerous Internet drafts, including the fundamental draft about the protocol [8] and the mapping of HTTP/2 and QUIC, which was replaced by a draft on HTTP/3 where the two protocols finally converge [9]. Today, QUIC is seen mostly in Google's services, such as YouTube, but it is expected that, with HTTP/3, it will be adopted by others as well [32, 33, 34].

2.2 Adaptive video streaming

OTT video streaming is a very bandwidth-intensive service. With the Internet providing best-effort delivery, adaptive streaming has emerged as a solution for controlling the QoS and QoE at an application level. The fundamental idea is to estimate bandwidth availability at the client during video playback, without direct feedback from the network, and dynamically adapt the quality of the content being streamed to measured conditions. For this to be possible, the content needs to be prepared on the server side, and adaptation logic needs to be implemented on the client side. Figure 2.1 illustrates the main idea, explained in the text that follows.

The content is split into discrete file segments (fragments, chunks), typically lasting 2-10 seconds, that contain audio, video or other data. The data can be multiplexed in a segment, or stored in separate audio/video/other segments. Segments are then encoded using different bitrates, resolutions, codecs, etc., where a sequence of segments with the same encoding is

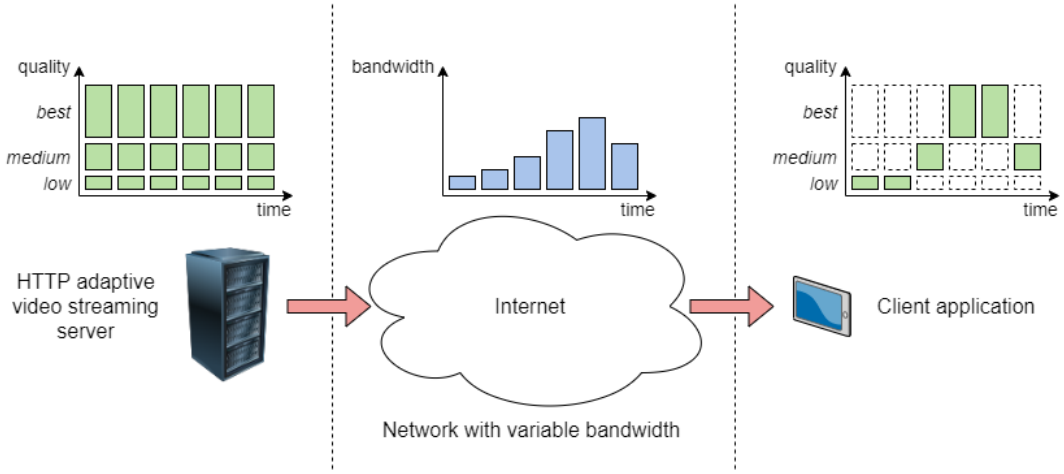


Figure 2.1: HTTP adaptive streaming. Based on the adaptation logic implemented within the client, the client requests video segments of quality levels in accordance with estimated network conditions and buffer status (figure adapted from [35]).

referred to as a profile. For example, a video clip can be split into segments of 5 seconds in duration and encoded as 6 video profiles differing in bitrate, resolution (360p, 480p, and 720p), and used codecs (H.264, VP9). Information about available profiles (bitrate, resolution, codec, etc.) is stored into a manifest file, together with the URLs at which the profile is stored [36].

Upon accessing the content, the client sends an HTTP request for the manifest file. As the client is aware of its capabilities (e.g., supported codecs), based on the adaptation logic, it selects appropriate audio and video profiles and requests the first segment accordingly. While the adaptation logic on the choice of the first segment is loosely defined and depends on the implementation, subsequent segments are chosen based on bandwidth measurements (and other parameters, such as codecs, device screen resolution). The client can monitor the changes in bandwidth availability by tracking the amount of time needed for a segment of certain size to be downloaded.

The adaptation logic becomes more complicated by introducing the client-side buffer into play. If the client application is implemented so as to keep 100 seconds of video content in the buffer, it does not necessarily make sense to react to sudden bandwidth drops immediately. On the other hand, if there is 100 seconds of low quality content in the buffer, but a surge in bandwidth is detected, it may be reasonable to drop the low quality content and request higher quality. These, and many other questions, as the amount of the content to be stored in the buffer, or the logic for triggering quality switches, are part of the service adaptation strategy and not specified by standards. Various employed adaptation strategies have been analysed over the past years [13, 37, 38, 39], but we note that these findings are susceptible to change and may become outdated, as service providers change their deployed strategies with newer service versions available on the market.

While adaptive streaming is the de facto standard for OTT video streaming today, and the

fundamental idea has not changed much since its beginnings, there are numerous active research avenues related to HAS. The available HAS standards only define the formats, while the definition of the exact adaptation and buffering strategy is left to the service. There are numerous research efforts aimed at optimising the used parameters, such as segment durations, and the adaptation algorithm [40, 41, 42, 43, 44, 45, 46].

A lively research area is HAS for video streamed to head-mounted VR and AR devices. As 360-degree video requires a significantly higher bandwidth, tile-based streaming emerged as a solution for lowering the bandwidth consumption [47, 48, 49, 50, 51, 52, 53]. The main idea it to split the camera view into tiles and stream the tiles in the current field of view in high quality, while the rest of the tiles is streamed in lower quality. Once the user turns their head, the field of view changes, and so do the tiles being downloaded in high quality. This is further complicated in cases when the user is allowed to move freely through the immersive scene (6 degrees of freedom video streaming), where point clouds may be utilised [54, 55, 56, 57, 58].

2.3 HTTP adaptive streaming architecture and standards

While they share the same underlying idea, a number of different proprietary solutions and international standards for HAS have emerged over the years. These include Microsoft Smooth Streaming (MSS) [59], Apple HTTP Live Streaming (HLS) [60], Adobe HTTP Dynamic Streaming (HDS) [61], MPEG Dynamic Adaptive Streaming over HTTP (DASH) [5], and others. Nowadays, popular video streaming services primarily rely on DASH and HLS, explained further in the following subsections.

2.3.1 MPEG Dynamic Adaptive Streaming over HTTP

MPEG-DASH is the first international standard for HAS, ratified in 2012 and published by ISO as ISO/IEC 23009-1 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats [5]. This is the fundamental document defining the format of the manifest and the segments. The adaptation logic employed by the client, the media player implementation, and exact communication between the components of the system are out of the scope of the standard. In more recent years, adjacent documents were published or are under development, with the full list of eight standards given below.

- ISO/IEC 23009-1:2019 DASH Part 1: Media presentation description and segment formats (4th edition published) [5],
- ISO/IEC 23009-2 DASH Part 2: Conformance and reference software (3rd edition under development) [62],

- ISO/IEC AWI TR 23009-3 DASH Part 3: Implementation guidelines (3rd edition under development) [63],
- ISO/IEC 23009-4:2018 DASH Part 4: Segment encryption and authentication (2nd edition published) [64],
- ISO/IEC 23009-5:2017 DASH Part 5: Server and network assisted DASH (SAND) (1st edition published) [65],
- ISO/IEC 23009-6:2017 DASH Part 6: DASH with server push and WebSockets (1st edition published) [66],
- ISO/IEC WD TR 23009-7 DASH Part 7: Delivery of CMAF contents with DASH (1st edition under development) [67],
- ISO/IEC DIS 23009-8 DASH Part 8: Session-based DASH operations (1st edition under development) [68].

Figure 2.2 illustrates a simple streaming scenario involving an HTTP server and a DASH client [69]. In the context of DASH, the manifest is stored as a Media Presentation Description (MPD) file. The file is in XML format and contains information about available content, its alternatives, URL addresses, etc. In MPD's hierarchical data model, a video can first be split into *periods*, or it can be observed as a single period. Each period has a starting time and durations (e.g., start = 0 seconds; duration = 60 seconds) and consists of multiple *adaptation sets*. An adaptation set in the XML hierarchy stores the information about a single media component and its encoding alternatives (e.g., information about audio, video or text alternatives is an adaptation set). Each adaptation set contains *representations* – encoding alternatives of the same media component, varying in bitrate, resolution, framerate, etc. Finally, each representation includes information about multiple *segments* – media stream chunks. Segment information includes a URI and temporal information about the segment.

Once a DASH client obtains the MPD file, it parses the XML and selects the set of representations that will be used, based on device/application capabilities and user's preferences (e.g, if only certain codecs are supported). If the player is set to automatically adapt the streaming quality, the adaptation algorithm within the client will dynamically select the appropriate segment representations among the supported subset. Each segment is requested using the HTTP GET method based on URIs specified in the MPD. MPEG-DASH itself is not limited with respect to codecs, uses MP4 or M2TS container formats, and supports both multiplexed and unmultiplexed content.

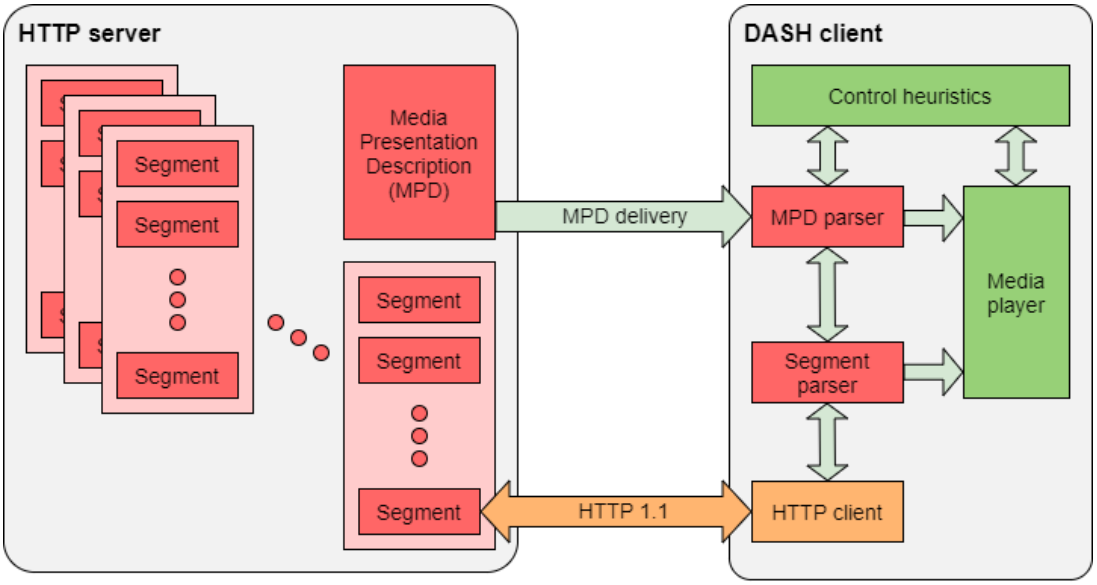


Figure 2.2: The MPEG-DASH standard specifies the formats and functionalities of the red blocks. The employed control heuristics and used media players are out of the scope of the standard (figure adapted from [69]).

2.3.2 Apple HTTP Live Streaming

HLS is a proprietary HAS solution developed by Apple. The specification of the solution is published by IETF as RFC 8216, under the Informational category [6]. HLS is widely used for video streaming on Apple devices, as they do not natively support MPEG-DASH. This is why popular services, such as YouTube, implement video streaming both using DASH and HLS, and the choice of the technology depends on the used device.

While MPEG-DASH is codec agnostic, HLS supports only MP4 media files containing HEVC or H.264 video, and AAC or AC-3 audio [70]. However, the overall architecture and communication is based on the same HAS principles. The key difference between DASH and HLS, besides the supported codecs, lies in the manifest file. As opposed to XML format used in DASH, HLS stores the manifest as M3U8 playlist – the Unicode version of MPEG Audio Layer 3 URL. Consequently, streaming providers can serve the same segments using DASH and HLS, given that both manifest formats are available, where there may be additional representations used only for DASH, with characteristics unsupported by HLS.

2.4 Chapter summary

Most of today's global network traffic is generated by video streaming services, with the HTTP adaptive streaming paradigm considered to be the de facto standard for delivering video content, both on demand and live. While there are multiple proprietary solutions and international standards, the main idea of HAS spans across all – the goal is to reduce the initial waiting time

and ensure a smooth playout without video stalling, by adapting the video quality in terms of bitrate and resolution to present network conditions.

This chapter provided an overview of the technology and the key differences among the two most commonly used standards, MPEG-DASH and Apple HLS. The standards define the formats of the data that is being exchanged between the clients and the server, but the exact algorithm used for the adaptation depends on the implementation. While the high level goals of HAS are already rooted in Quality of Experience, understanding a deeper connection between the users' perception of the quality and the objective performance of a specific service is of high importance for service providers. The following chapter thus introduces the domain of QoE and particularly sheds light on QoE modelling, monitoring, and management in the context of HAS.

Chapter 3

Quality of Experience modelling, monitoring, and management for adaptive video streaming

As multimedia services offered across the Internet have grown more complex, and end user service-usage behaviour patterns have evolved given the proliferation of new devices, content, and applications, research geared towards understanding and optimising end-user QoE has gained significant momentum. This chapter first introduces key definitions and methods related to QoE in Section 3.1, followed by the introduction of the typical QoE management cycle, from QoE modelling based on the understanding and impact of underlying QoE influence factors, QoE monitoring, and finally the management of QoE. These concepts are then explored in the context of HAS in Section 3.2, followed by a more in-depth review of state-of-the-art with regards to QoE monitoring for adaptive streaming in Section 3.3. The chapter is concluded with a summary in Section 3.4.

3.1 QoE management cycle

In the early 90s, the performance of services in the field of telecommunications was commonly described using easily quantifiable and measurable parameters such as packet loss, delay, and bitrate. In this context, the term Quality of Service (QoS) emerged as the *"totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service* (ITU-T definition [71]). While the definition mentions the needs of the user, QoS is primarily focused on technical performance parameters. This is related to provisioning the service within the specified quality levels, in line with the Service Level Agreement (SLA) – *"a formal document listing a set of performance characteristics and target values (or range) to be delivered for a service or portfolio of services by the service provider"*.

With the changes in the Internet ecosystem and the rise of multimedia services, it was becoming challenging to describe the users' needs with respect to quality using QoS only. There was a need for a multidimensional construct, that takes into account not only technical aspects of quality, but also different contextual factors and users' subjective perception of the quality. QoE came into play as a measure of the *"overall acceptability of an application or service, as perceived subjectively by the end user"* (ITU-T definition [72]), as the research community was aiming to unveil relationships between QoE and QoS that can be measured and managed. However, while the notion of QoE is relatively intuitive, gaining a proper understanding of what constitutes QoE for various services, and how it can be assessed and taken into account while designing services, remains challenging.

Since the late 90s, there was a plethora of research on QoE, uncovering how users perceive different services. In 2013, the research community gathered in the scope of QUALINET, the European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), published a White Paper providing key definitions related to QoE, and more firmly establishing future grounds for QoE research. The "Qualinet White Paper on Definitions of Quality of Experience (QoE) and Related Concepts" [73] defines QoE as *"the degree of delight or annoyance of the user of an application or service"*. This definition was also adopted by the ITU-T Recomm. P.10/G.100 [74] in 2017, replacing the definition from the beginning of this section. The white paper defines QoE influence factors as *"any characteristic of a user, system, service, application, or context whose actual state or setting may have influence on the Quality of Experience for the user"*, further grouping them into human, system, and context influence factors. It also states that QoE can be decomposed into multiple perceptual features – *"perceivable, recognised and nameable characteristics of the individual's experience of a service which contributes to its quality"*.

The relationships between QoE influence factors, QoE features and QoE are quantified by QoE models, where the fundamental way of obtaining end user QoE ratings is through subjective quality assessment. These are experiments in which users (test subjects) provide quality ratings for the experience of using a service in a given scenario. Depending on the service type, different guidelines have been established for constructing the test scenarios and running such experiments, either in a lab, in the field, or via crowdsourcing techniques. Test subjects typically rate the quality on a 5-point absolute category rating (ACR) scale, ranging from 1-bad, to 5-excellent. While the user ratings are obtained on a discrete ACR scale, the outcome for a certain scenario is commonly averaged into a Mean Opinion Score (MOS) [75]. Insights obtained through subjective assessments can be used to develop QoE models, either by discovering fundamental mathematical relationships [76, 77] or by applying machine learning [78], that can later be reused for estimating QoE for a specific application type, without running complex and costly subjective evaluations. The clear limitation of such model-based QoE evaluations, but

also the source of robustness, is the objectivity, as user-factors often get averaged and neglected.

Prior to QoE research gaining momentum, services in the Internet were mainly delivered using the "best effort" principle with network and service management decisions driven by QoS measurements. With research in the field of QoE yielding deeper insights into users' perception of quality in the form of QoE models, there are opportunities to utilise this knowledge to manage services and networks from a more "user perspective". Figure 3.1 illustrates the three key steps towards QoE management: 1) devising QoE models that identify factors influencing QoE for a given service and describe their relationship to QoE, 2) monitoring QoE based on the defined influence factors and models, and 3) managing the service and/or network so as to efficiently use constrained resources while keeping a satisfied user base and reducing customer churn. What is being monitored clearly depends on the perspective: service providers can monitor application-level factors that more directly influence QoE, while network operators may monitor parameters obtainable in the network, such as available bandwidth or patterns in encrypted traffic (Figure 3.2).

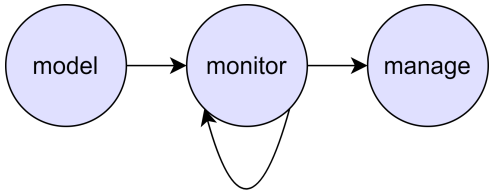


Figure 3.1: Prerequisites for QoE management are the availability of a QoE model and the solution for monitoring parameters dictated by the model.

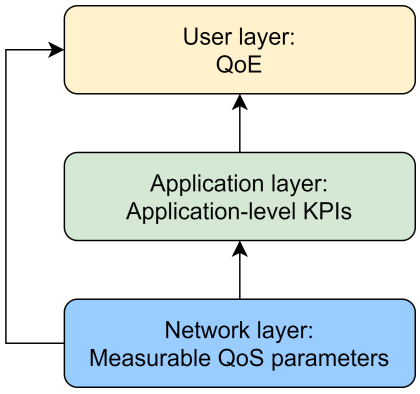


Figure 3.2: The mapping of QoE-influencing data on different layers onto QoE is defined by different models.

QoE management solutions can thus be loosely categorised into network and application management approaches (Figure 3.3). The goal of QoE management in both perspectives is to keep satisfactory levels of QoE, while efficiently allocating available resources. While network management is based on monitoring and control of the network and network parameters that are known to have influence over QoE, application management aims to adapt the quality and performance of the service at an application level (at the client application or host/cloud) [79].

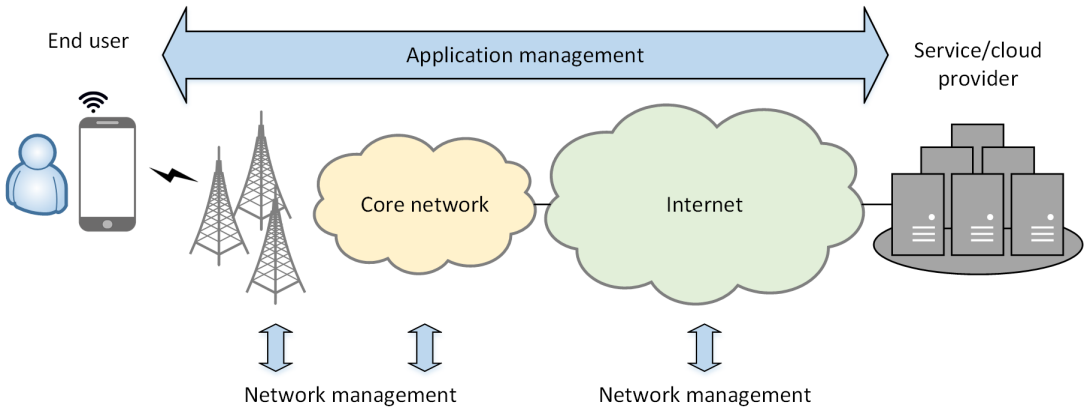


Figure 3.3: Network management and application management (figure adapted from [79]).

Besides the easily distinguishable network and application management approaches, what has also been investigated are cooperative solutions for QoE management. The idea of cross-layer QoE–driven admission control and resource allocation is not new [80, 81, 82], but reappeared in the context of software defined networks (SDN) and network function virtualisation (NFV), with researchers aiming to motivate QoE-centric collaboration between OTT providers and ISPs.

While the incentives for cooperation exist [83, 84, 85], there is a lack of business models and concrete technical solutions that would support the information exchange, as well as regulatory issues that need to be resolved. The implications of network neutrality regulation on potential QoE management and control solutions need to be considered [86, 87]. Currently, most OTTs are not keen on disclosing the information about their streams due to privacy concerns. On the other hand, ISPs are not willing to share information about their network due to the fear of losing competitive advantage and regulations with regard to national security. There is currently a lack of technical realisations deployed in networks which offer trusted information exchange between OTT service providers and network providers, thus resulting in QoE-aware service and network management being commonly performed independently of one another. Standardisation efforts working towards defining entities and interfaces for information exchange can be found in the scope of 3GPP specifications addressing 5G networks [88].

Following up, Section 3.3.3 provides more information on both application and network management specifically for HAS. For further information about QoE management in general, we refer to the following papers. Focusing on QoE management from the wireless network perspective, a survey by Barakovic *et al.* [89] provides a summary of research activities before 2012 in three management aspects: QoE modelling, monitoring and measurement, and adaptation and optimisation. While a lot has changed since the survey was published, many of the concepts mentioned are still applicable today. A newer survey by Skorin-Kapov *et al.*, published in 2018, provides a comprehensive overview of emerging concepts and challenges in QoE modelling, monitoring, and management [2].

3.2 QoE in the context of HTTP adaptive video streaming services

QoE models in general describe the relationship between measurable parameters, either on application layer, or on network layer, and QoE. Depending on QoE monitoring and management goals, and depending on the data available at the point where monitoring is performed, different models may be adequate. The survey given by Seufert *et al.* [90] from 2015 gives a comprehensive overview of subjective studies that cover QoE aspects of adaptation, and discusses QoE influence factors and corresponding QoE models for adaptive video streaming. At the time, a number of QoE models for HAS have been proposed [91, 92, 93, 94] depicting the relationship between stalling duration/length, initial delays, adaptation behaviour, and QoE. These influence factors have mostly been investigated separately, and, back then, there was no multidimensional model for HAS that would take into account the interplay between these factors. However, the aforementioned research brought interesting insights that surely shaped the way HAS services are delivered, such as the fact that users consider stalling to be the most annoying QoE degradation artefact, and would rather experience longer initial delay in order to prevent stalling [93]. Other interesting surveys that summarise research efforts on QoE measurement and QoE modelling for HAS can be found in [95] and [96].

In November 2016 the ITU-T (SG12) published "Recommendation P.1203: Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport" [97] (formerly referred to as P.NATS). This is the first standardised multidimensional QoE model for HAS, that takes into account the information about the device used for viewing the content, information about the played audio levels, about played video levels (resolution, bitrate, codec, etc.), and initial delay and stalling information. The Recommendation consists of an introductory document (P.1203) [97] and documents describing three objective parametric quality assessment modules:

- the video quality estimation module (P.1203.1) [98, 99],
- the audio quality estimation module (P.1203.2) [100], and
- the quality integration module (P.1203.3) [101, 102].

The model distinguishes between 4 modes of operation, depending on the level of encryption, that define the input parameters of the model. Input parameters are processed by 3 mentioned modules, depicted with the model scheme (Figure 3.4). The outputs of the model are specified as follows:

- O.21: Audio coding quality per output sampling interval
  Per-one-second scores provided per session and on a 1-5 quality scale.
- O.22: Video coding quality per output sampling interval
  Per-one-second scores provided per session and on a 1-5 quality scale.

- O.23: Perceptual stalling indication
  Single score on a 1-5 quality scale for the session.
- O.34: Audiovisual segment coding quality per output sampling interval
  Multiple segment scores provided per session.
  Window-size same as for/synced with O.21, O.22.
- O.35: Final audiovisual coding quality score
  Single score for the session, on a 1-5 quality scale.
  Includes aspects of temporal integration.
- O.46: Final media session quality score
  Single score for the session, on a 1-5 quality scale.

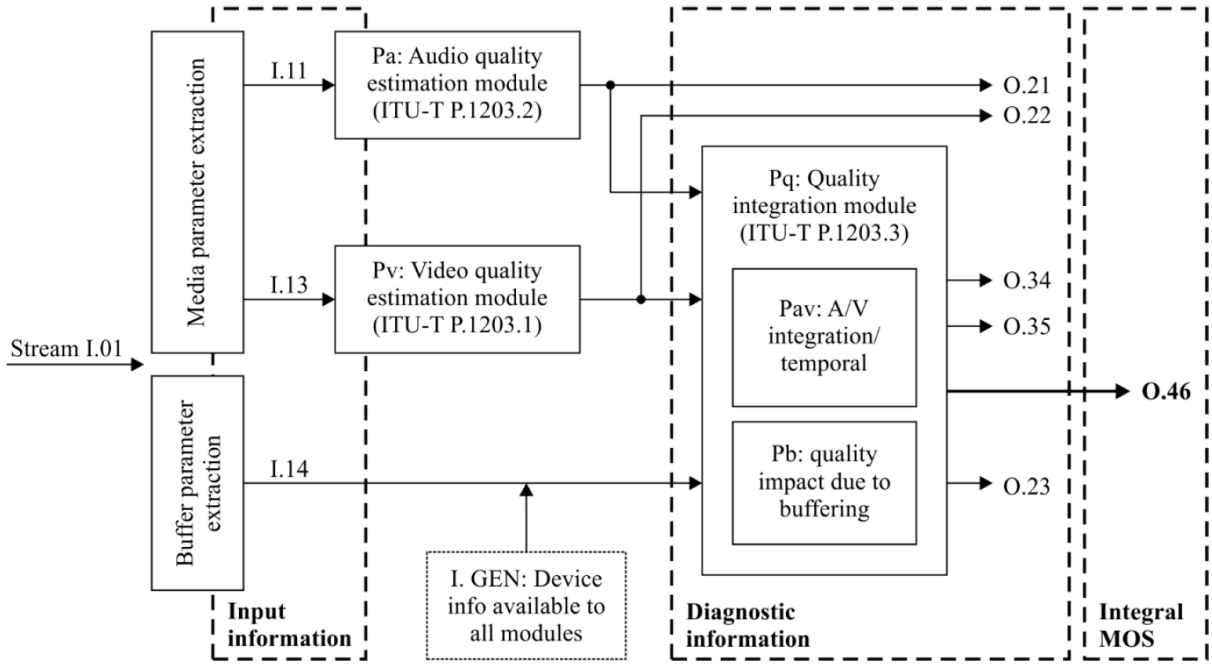


Figure 3.4: Building blocks of the ITU-T P.1203 model (figure taken from [97]).

Audio and video estimation modules provide one audio or video quality score per output sampling interval and do not perform any kind of long-term temporal integration of these scores. This interaction is handled by the integration module, which also calculates the impact due to buffering and the overall MOS. The model was built and validated based on 30 individual subjective studies using 1064 unique, 1 to 5 minutes long, processed video sequences, encoded with H.264 video and AAC audio codec. An implementation of the model is available in [103] [99, 104]. We emphasise that the model was only validated on H.264–encoded video, and thus should not be used for VP9 or H.265/HEVC video. For such applications, there is a non-standardised extension of P.1203, with implementation available at [105] [106]. The model published in P.1203 also does not support resolutions higher than 1080p, which are now addressed in a new model published in 2020 by ITU as Recomm. P.1204 [107] [108]. Finally, a big challenge that remains unaddressed by standards to date is the impact of users' interac-

tion on input parameters of the model as well as the users' expectations towards the service performance when interactions are performed.

Besides the studies and models intended at unveiling the relationships between application-level metrics and QoE, there were also studies aimed at quantifying the relationship between network QoS and QoE. Many of those employed machine learning methods for mapping QoS parameters such as packet loss and download time to QoE. The survey by Aroussi and Mellouk [78] presents an overview of QoE-QoS correlation models based on such techniques. Focusing more on evaluating different classifiers for correlating video streaming QoS to QoE, Mushtaq *et al.* [109] used six classifiers (Naïve Bayes, Support Vector Machines, k-nearest Neighbors, Decision Tree, Random Forest and Neural Networks) to correlate network parameters, characteristics of videos, terminal characteristics and users' profiles to QoE. These studies contributed to the understanding of the relationships between network QoS and QoE, but are today not applicable in the context of QoE monitoring, due to challenges related to encryption.

3.3 QoE monitoring of encrypted video streaming

In-network QoE monitoring may be considered a prerequisite for employing QoE-aware network management. With today's prevalence of encryption in OTT traffic, gaining insight into QoE and application-level performance from a network provider perspective is challenging, with a number of prototype solutions having been proposed. The state-of-the-art with regards to such solutions is reviewed in Section 3.3.2. The key parts of these solutions are the models, which are devised based on acquired ground-truth performance data. Tools for client-side QoE monitoring are thus explored first, in Section 3.3.1. The section is concluded with a survey of research on utilising QoE/KPI and their estimates for HAS application and network management in Section 3.3.3.

3.3.1 Client-side QoE monitoring

Monitoring solutions deployed on client devices provide a reliable and accurate view of application-layer KPIs, which can be further mapped to user perceived QoE, based on existing and emerging QoE models, such as the one published in ITU-T Recomm. P.1203 [97]. Over the past years, different client-side applications that monitor HAS service performance have been developed. Ericsson's NPT (Network Performance Test) [110] measures the performance of many OTT services including YouTube, Facebook Watch, and Youku. Another commercial example is Equal One by V3D [111], which includes active video streaming QoE tests from within the application. A number of applications have also emerged in the research community, mostly focused on YouTube [13, 112, 113], as it offers convenient APIs.

Such applications have been used for benchmarking network performance, analysing service behaviour, etc. An example of a wide-scale benchmarking initiative in the direction of publishing worldwide YouTube KPIs is the development of YouSlow [114], a system designed to monitor YouTube stalling events on clients, collect reported data on Google maps, and calculate ISP statistics in terms of stalling duration and location. In [115], the authors analysed the influence of both constant and dynamically changing network access conditions to better understand the QoE requirements of popular mobile apps, including YouTube. In the case of YouTube, they used YoMoApp (YouTube Performance Monitoring Application) developed by Wamser *et al.* [112], an Android application that passively monitors application-layer YouTube KPIs on smartphones using YouTube's IFrame API. A similar application, called YouQ, was developed by Orsolic *et al.* [12, 13, 14], with two versions, one based on the YouTube IFrame API [116] and the other based on the native YouTube Android API [117].

These tools were used in studies aimed at discovering the impact of different network conditions, such as variable available bandwidth, on video player behaviour [13, 115] and consequently application-layer KPIs and end user QoE. Studies analysed both network flow parameters and application-layer streaming parameters that affect QoE [118, 119, 120]. There are also studies that describe testbeds including client-side monitoring to perform large-scale network performance measurements [121] and analyse YouTube's DASH implementation [13, 122, 123].

While these applications are valuable for testing the performance of HAS streaming under different conditions, and were initially used for collecting ground-truth data for purposes of developing QoE/KPI estimation models, our study reported in [14] showed that these measurement applications do not necessarily behave in the same way as the actual YouTube application. This should be taken into consideration, as QoE/KPI estimation models trained on data from these applications may behave unexpectedly when deployed in the network to estimate the performance of YouTube streaming. The same study [14] proposes an approach called ViQMon, that extracts data from the YouTube application's *Stats for Nerds* window. Seufert *et al.* came to the same conclusions and developed a similar approach for extracting data from the YouTube application [124]. Although not found in literature, the same approach could be applied for Twitch and Netflix (and possibly other services of interest), as a similar overlay performance report window is available in these services, but only when viewed in the browser. Another approach that does not require an API and is thus applicable for a larger set of video streaming services is described in [125]. The authors capture the screen of a smartphone running a video streaming application and then use image processing methods, such as spinner pattern matching, to detect the video resolution and stalling occurrence.

When it comes to collecting data for the development of in-network QoE/KPI estimation models, collecting a large and varied (in terms of application-layer performance and network

traffic patterns) dataset is crucial. This is why most recent papers [126, 127, 128, 129] report on their data collection automation efforts, employing frameworks such as Selenium [130]. The automated approach in the aforementioned paper was applied for data collection on desktop devices, but we note that there are also similar frameworks that work for Android and iOS native applications (e.g., Selendroid [131], Appium [132]).

A different approach has been proposed by researchers investigating the potential of deploying the monitoring modules on end user devices, as opposed to test devices. Ahmad *et al.* investigate the advantages and disadvantages of QoE monitoring at the user terminal in [133], while in [134] they present their solution and analyse its impact on the end device resource utilisation. Although a monitoring application provided by an ISP and installed on the user terminal would potentially provide the best insight into QoE without directly requesting feedback from users, the solution described in the papers by Ahmad *et al.* relies on the existence of information exchange between the OTT service application and monitoring application, which raises privacy concerns and relies on the willingness of users to install such probes on their devices.

3.3.2 Network-centric QoE monitoring

Prior to the widespread use of traffic encryption, in-network QoE monitoring solutions relied on deep packet inspection (DPI) for extracting information about video quality (e.g., streamed resolution, codecs, bitrate) [135, 136]. With the adoption of encryption, such solutions were for the most part no longer viable, which opened up new research questions related to in-network QoE estimation based on the analysis of encrypted video traffic.

On a high level, two different approaches have been proposed to tackle this problem: *session-modelling–based* (SM) and *machine-learning–based* (ML). SM–based solutions require knowledge about the streaming protocol and rely on session reconstruction for the inference of QoE–influencing KPIs. Mangla *et al.* present such a solution called eMIMIC in [137], highlighting the performance it achieves in comparison to ML–based solutions. The solution is based on reconstructing the chunk–based delivery sequence of a video session from packet traces, and then the use of this sequence to model a video session and estimate average bitrate, re-buffering ratio, bitrate switches, and start-up-time. The presented system would need to be adapted to work with QUIC traffic, potentially reducing chunk-detection performance which the system is based upon. Although in [138] Krishnamoorthi *et al.* do not estimate QoE/KPIs as such, but rather try to predict buffer conditions by emulating the client buffer, parts of the approach can be applied for this problem as well.

However, given the problem dimensionality – resulting from a large variety of devices, platforms, streaming services, applications from which the content is accessed, different network types, and different protocols – finding analytical solutions for a wide range of potential use-cases becomes extremely complex. Moreover, service providers may change the streaming pro-

tocol, be it in terms of adaptation strategy, used network protocols or something else, potentially leading to the need for network operators to adapt their QoE/KPI estimation models. For such reasons, numerous recent studies have turned to utilising ML techniques for QoE/KPI estimation, arguing that such approaches are potentially more flexible and sustainable in the long run [10, 11, 13, 126, 129, 139, 140, 141]. Though in both SM and ML approaches, application-level ground-truth data needs to be collected for the learning phase, ML–based approaches typically require a larger dataset. Obtaining such a dataset can be relatively easy for some services and platforms (such as YouTube viewed on a desktop device or an Android phone, especially if automation is employed), but more difficult for others, as described in Section 3.3.1.

ML–based QoE/KPI estimation solutions proposed in literature are fairly similar in their core idea. Once the dataset (including network traffic and application-level performance measurements) is collected, statistical network traffic features are extracted from the traffic trace, and corresponding desired KPI/QoE metrics are calculated based on measured application-level performance data. The combined data is fed to ML algorithms for the purpose of training models that are then able to estimate KPIs/QoE based only on network traffic features that can be calculated in the network despite the encryption. Published papers differ in terms of: data collection methods, specified traffic features and targets, objectives, and use-cases used for validation. We provide an overview of recent studies on ML–based in-network QoE/KPI estimation for HAS in reverse chronological order in Table 3.1. The table summarises the main objectives of these studies (such as session-level vs. real-time QoE/KPI estimation, addressed use-cases, target variables), information on used datasets (e.g., collected in a lab WiFi network vs. mobile network, number of videos in the dataset), and key findings.

The table concisely sums up the evolution of the methodologies for machine-learning–based in-network QoE/KPI estimation. Building on top of ideas of applying ML for estimating QoE in the network [10, 11], our initial work on the topic [12, 13] has proven the feasibility of classifying YouTube video streams into QoE classes. While previous approaches required access to packet payloads, at least in the model training phase, the methodology we presented was fully applicable in the context of traffic encryption. Soon thereafter, YouTube switched to using QUIC/UDP, as opposed to TLS/TCP, which we observed from 2017 onward. What we also observed with the introduction of QUIC was that the video player offered in YouTube's APIs, which we relied on for the collection of ground-truth data, did not behave in the same way as the player in the actual YouTube application. We study these changes in our work in [14].

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Orsolic and Skorin-Kapov (2020) [16]	• Framework for in-network QoE/KPI monitoring • Real-time and session-level QoE/KPI estimation • Platform-agnostic models, real-time predictions as additional session-level features, model re-evaluation, session delimitation video streams	• 3 datasets from Jan. 2018 – Jan. 2019 • Two collected on Android (299 and 394 videos) and one on iOS (299 videos) • Ground truth data from *Stats for Nerds*	• Session-level QoE estimation can be improved with **real-time KPI estimates used as additional predictors** • Models can be trained so as to address multiple use-cases at once, if they are trained on data from those use-cases • **Model performance can decrease over time due to identified changes in service adaptation behaviour, protocols, etc. – need for an update**

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Bartolec *et al.* (2019, 2020) [142, 143]	• Session-level KPI classification (resolution, initial delay, video bitrate) • YouTube for Android • IP-level traffic features • **Consideration of realistic end-user playback–related interactions**	• Dataset without user interactions, 299 videos (2019) • Dataset with user interactions (pause, seek, abandon), 307 videos (2019) • Dataset with user interactions (pause, seek, abandon, multiple interactions, playback speed), 409 videos (2020) • Ground-truth data from *Stats for Nerds* • Lab env. with bandwidth emulation	• Models trained on the dataset that does not include user interactions perform worse on data with user interactions • **Need to include more realistic data, with user interactions**, into the model training phase • **Model training can be simplified by omitting some interactions that do not affect KPI estimation**
Bronzino *et al.* (2019) [139]	• Real-time startup delay and resolution estimation • Proposition of a generic models applicable across multiple services (Netflix, YouTube over QUIC and TCP, Amazon and Twitch)	• 5 datasets containing 13765 sessions (Nov. 2017 – May 2019) • 6 laptops in home networks, 5 laptops in the lab network with emulated conditions • Ground-truth data collected through a Chrome extension	• Precise initial delay regression models • Fine-grained resolution classification models on 10-second windows • **Generic models applicable for five service scenarios, if the training data included data from all services**

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Schwarzmann *et al.* (2019, 2020) [144, 145]	• Session-level MOS estimation (regression) based on **5G monitoring data** • Simulation of a mobile video streaming use-case	• Generated using an *OMNeT++* simulator • Varying simulation parameters, simulation scenarios (e.g., no. of users, their locations), client parameters, and video properties	• Understanding the correlation of 5G monitoring data to user QoE • An accuracy vs. cost analysis of ML–based QoE estimation in 5G
Wassermann *et al.* (2019) [128, 129]	• **Real-time resolution classification**; bitrate considered in [128] • Focus on YouTube in browser • IP-level traffic features in time-windows	• Dataset with more than 15000 videos (Jun. 2018 – Feb. 2019) • Home/corporate WiFi, lab WiFi with Bw emulation, and LTE mobile network • **Data collection automation – Selenium**	• Tree–based models provide highly accurate results and are execution-fast • Highly efficient **stream–based strategy for feature computation**
Gutterman *et al.* (2019) [140]	• Real-time prediction of buffer warning, streaming phase, and video resolution • YouTube in browser • **Inclusion of chunk–based features** derived by the chunk detection algorithm	• Dataset containing over 500 sessions (2018) • 3 different WiFi networks • Static and movement scenarios • Ground-truth data collection through IFrame API	• Estimated **chunk–based features improved prediction accuracy**, when compared to IP-level features only

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Seufert *et al.* (2019) [126, 127]	• Real-time **detection of stalling** • YouTube in browser • IP-level traffic features in time-windows	• 4714 YouTube sessions (2018) • Home/corporate WiFi, lab WiFI with Bw emulation, and LTE mobile network • **Data collection automation – Selenium**	• Promising results for random-forest–based stalling detection in a stream-based **real-time** fashion • Evaluation of relevance of different feature sets in [127]
Orsolic *et al.* (2018) [17]	• Session-level MOS (ITU-T P.1203) and KPI classification (resolution, stalling, initial delay, video bitrate) • IP-level traffic features • **Focus on YouTube via iOS platform**	• 4 datasets from Sept. 2017 – Mar. 2018 • Two datasets collected in the lab (emulated conditions), one on Android (394 videos), one on iOS (383 videos) • Two datasets collected in mobile networks on iOS (128 + 111 videos) • Ground-truth data from *Stats for Nerds*	• Promising performance of models for iOS, also on data from mobile network • **Android–trained models applicable to iOS data, with slight decrease in performance**

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Orsolic *et al.* (2018) [14]	• Session-level MOS (ITU-T P.1203) and KPI classification (resolution, bitrate) • YouTube on Android • IP-level traffic features • **Comparison of ground-truth data collection methodologies**	• 3 datasets from 2017 – Jan. 2018 • One collected using three app-level data collection methods (YouTube IFrame API, Android API, ***Stats for Nerds* extraction**; 300 videos) • Two collected using the *SfN* extraction: in a lab network with Bw limitations (394 videos), in a mobile network (105 videos)	• Methodology for *Stats for Nerds* extraction based on screen recording and OCR • **Approaches relying on embedding YouTube videos for test purposes exhibit different characteristics as compared to the actual YouTube application**
Mazhar *et al.* (2018) [141]	• **Real-time KPI classification** (initial delay, stalling, resolution) • YouTube in browser • IP-level features (and TCP-level, if applicable)	• Dataset containing 5488 sessions over QUIC and 5375 over TCP (2017) • Data collection automation – Selenium • Lab env. with Bw, delay and packet loss emulation	• Approach for highly accurate decision-tree–based binary classification of KPIs in real-time

Table 3.1: Comparison of selected recent studies addressing machine-learning–based in-network QoE/KPI estimation for adaptive video streaming.

Author (Year)	**Objectives**	**Datasets**	**Key findings**
Orsolic *et al.* (2016, 2017) [12] [13]	• Session-level QoE classification (custom classes; prior to the publication of ITU-T P.1203) • YouTube on Android • TCP-level traffic features	• Dataset containing 1060 videos (Apr.-Jun. 2016) • Ground-truth data from IFrame–based application called YouQ • Lab env. with Bw emulation	• Detailed methodology – data collection, network conditions, ML analysis • Proved the **feasibility of classifying YouTube video streams into QoE classes**
Dimopoulos *et al.* (2016) [11]	• Session-level classification of stalling, average video quality and quality variations • Focus on YouTube • TCP-level traffic features	• Dataset containing 390000 unique sessions collected at a Web proxy (2016) • Only a small percentage of sessions based on adaptive streaming • Significant amount of unencrypted streams in the scope of the dataset	• Framework for detecting video streaming KPIs applicable for encrypted traffic • Changes in size and interarrival times of video segments are among the most important indicators of quality impairments

In parallel with the session-level QoE/KPI estimation solutions, a number of near–real-time KPI estimation approaches were proposed [126, 127, 128, 129, 139, 140, 141]. We refer to these as *real-time* KPI estimation approaches further on, although the estimation is performed on short intervals, rather than what would typically be considered as *real-time*. Such solutions, as opposed to session-level ones, might be more appropriate for network operators looking to dynamically manage QoE and optimise resource allocation (as opposed to session-level approaches which provide per-video session metrics and may be more appropriate for network planning purposes). In [16], we combine both approaches, by defining a generic and flexible framework for in-network QoE/KPI estimation. Besides presenting new datasets and new models, this is where we also explore additional practical research questions, such as the possibility to address multiple use-cases with a single model, combining the outputs of the real-time and session-level models, etc.

Focusing more on the deployment of these ML–based models in 5G networks, in [144, 145] the authors present their simulations used to assess such models when embedded in NWDAF (Network Data Analytics Functionalities) – an analytics entity in 5G with machine learning capabilities [88]. Another interesting avenue for further research on QoE monitoring for HAS is the inclusion of user behaviour and its impact on traffic patterns and, consequently, on model performance. In [142, 143], we explore this important issue, given its relation to deploying robust QoE monitoring solutions in the network. We note, however, that the analysis of the impact of user behaviour and interactions on QoE/KPI estimation model development is outside the scope of this thesis.

3.3.3 QoE-aware application and network management

QoE management approaches can be categorised into application-level and network-centric management [79]. Additionally, different collaborative approaches to QoE management can be considered, where 1) an application receives information from the network and performs the optimisation, 2) the network receives information from the application and performs the optimisation, 3) a policy manager receives information from both network and application and globally orchestrates control actions [146].

In the context of HAS, application-level QoE management is typically performed through the adaptation algorithm. Building on top of the HAS paradigm, commonly in line with available standards [5, 60], services define the buffering strategy, triggers for adaptation events, etc. [147]. The choices are aimed at optimising the users' experience, having in mind the characteristics of the service and users' behaviour. For example, YouTube videos are short and often not viewed in their entirety [148, 149]. On the other hand, videos on Netflix are typically long and often viewed in full. The exact employed algorithms change over time with new findings and fine tuning, but are usually not aware of the network organisation, rather just the endpoints.

In the direction of defining a potential form of information exchange for HAS services, a standard known as MPEG-DASH SAND (Server and network assisted DASH) has been published in ISO/IEC 23009-5:2017 [65]. The standard solves the problems that occur due to the decentralised and client–driven nature of DASH [5] by defining DASH-assisting network elements (DANE), and the communication between a DANE and a DASH client, as well as communication between two DANEs (Figure 3.5). The standard defines concrete protocols and a list of messages these entities can exchange [150, 151].

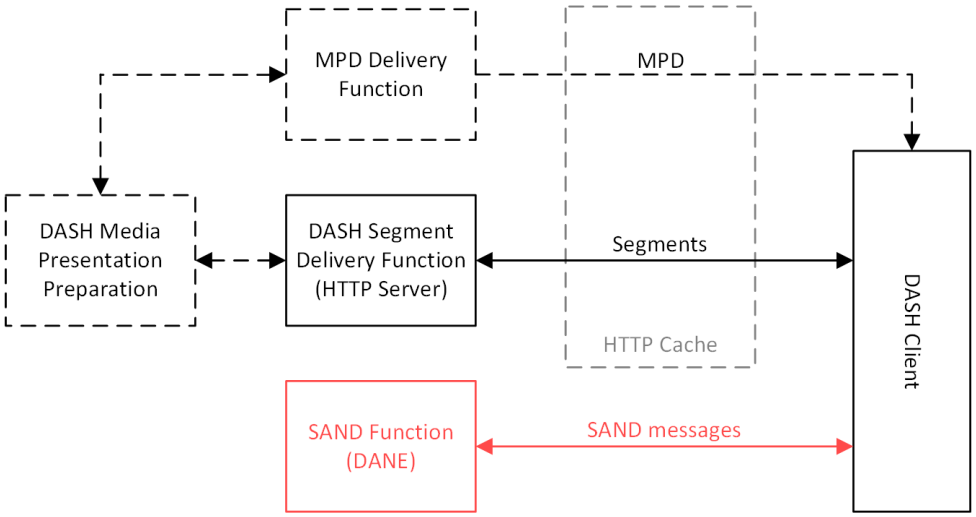


Figure 3.5: SAND–augmented DASH architecture (figure taken from [151]).

To investigate network–assisted streaming approaches, such as SAND, and their effect on QoE-related metrics among concurrent video flows sharing the common bottleneck Cofano *et al.* [152] built a video control plane that manages the network resources to maximise video quality fairness. They compared two approaches, allocation of bandwidth slices to video flows, and network-assisted bitrate selection on the client, assessing the performance through video quality fairness, video quality, and switching frequency. They conclude that both approaches provide a remarkable improvement of QoE when compared to the case without network assistance, and that the approach should be chosen depending on the QoE-related parameter we are trying to maximise and the concrete implementation of the DASH client.

Network assistance based on DASH SAND, with its Shared Resource Allocation (SRA) feature, has proven to be very helpful in alleviating issues with multiple clients competing for bandwidth in the same network bottleneck, as shown in [153]. Similarly, authors in [154] address the same problem of uneven bandwidth competition caused by the client-driven, on-off adaptation behaviour of HAS. They propose a new SDN–based dynamic resource allocation and management architecture for HAS systems, which aims to alleviate these scalability issues and improve the per-client QoE. Their architecture manages and allocates the network resources dynamically for each client based on their expected QoE.

As network functions in 5G will increasingly be deployed in the cloud, often characterised

as cloud deployed SDN/NFV, a number of other researchers have been exploring SDN/NFV based solutions for managing QoE [155, 156, 157, 158]. Softwarisation and virtualisation enable the network to be more flexible, as network functions can be run in a scalable number of instances, adapting to the current requirements. Also, new functions can easily be deployed in the cloud environment. In the context of QoE monitoring and management, virtualised monitoring architectures can be built more flexibly and scale more easily [159].

With the global view of the network that is available in the SDN architecture, and utilising QoE models, traffic management mechanisms can be invoked in a QoE-aware manner [160, 161]. Having appropriate models for estimating QoE or being provided by the performance measurements from the service provider, these technologies enable dynamic selection of paths or servers, bandwidth reservation, stream prioritisation, etc. While research on QoE monitoring and network optimisation is plentiful, there are still a lot of questions that need to be answered before recommendations can be proposed on how to bring the two together.

With the lack of QoE-centric collaboration between ISPs and OTTs, ISPs are determined to deploy QoE/KPI estimation modules in the network, to be able to detect and mitigate QoE degradations, and take informed optimisation actions. However, especially as the network is shifting towards virtualisation, it is not clear where such monitoring probes should be deployed [159]. An interesting avenue for future research is also exploration of the potential for utilising P4 [162] and deploying QoE estimations directly in the data plane [163, 164, 165].

3.4 Chapter summary

QoE is defined as *"the degree of delight or annoyance of the user of an application or service"* [74] and is as such used to express how users perceive objective quality degradations. While the measurements of QoE can be rather subjective, a lot of effort is being put into developing objective QoE models – models that, when provided with factors influencing the QoE, output a rating of an average user, commonly expressed as a value in a 1–5 interval (MOS score). This chapter explained the term QoE, and provided a high-level overview of methods, processes, and challenges in research related to the field of QoE. Moreover, the chapter went into the domain of HAS, presenting what constitutes QoE for HAS services.

Narrowing down to the specific focus of the thesis, the chapter reviewed the state-of-the-art in the area of QoE monitoring for encrypted video streaming. Network operators often lack insight into the QoE of their customers using video streaming services, due to stream encryption and lack of feedback channels. Moreover, a number of concerns, such as those related to privacy issues, make service providers unwilling to share information with network operators concerning service parameters and performance. The chapter provided an overview of efforts towards developing in-network QoE monitoring solutions applicable under these conditions.

Moreover, the chapter gave an overview of objective performance measurement techniques for HAS and also how these can potentially be used for QoE management.

Given that QoE/KPI estimation, which is the key part of QoE monitoring, is a very dynamic field, we provided a state-of-the-art overview and compared various findings, including also our own work conducted in the scope of this thesis. The following chapter gives a systematic view of all the studies conducted as a part of the research reported in this thesis, and explains the methodology applied across all of the studies.

Chapter 4

Methodology for in-network estimation of QoE/KPIs of adaptive video streaming

Following the state-of-the-art review given in the previous chapters, this chapter presents the concept of a generic and flexible framework for in-network ML–based HAS QoE/KPI monitoring. The framework provides a context for all key contributions of the thesis by outlining the challenges related to the development, deployment, and maintenance of such solutions. The practical focus of the thesis is on the development of ML–based models that are utilised to estimate QoE/KPIs within the framework, which we refer to as *QoE/KPI estimation* further on (as opposed to *QoE/KPI monitoring*, which is used in a broader sense). Figure 1.5 in Chapter 1 illustrated the overall research methodology, applied to identify and address selected challenges in the field through six studies conducted over a period of four years. This chapter presents an overview of the studies, and describes the generic, high-level methodology to data collection and model training applied across all of the studies. The term *methodology* further on refers to the methodology for in-network estimation of QoE/KPIs (steps 3 and 4 in Figure 1.5) and not to the overall research methodology. As each study addresses different use-cases and/or considers new findings and service implementation changes, the particularities of the methodology applied in the scope of individual studies are discussed in further details in the following chapters. This chapter is organised as follows: Section 4.1 presents the framework through a visual depiction of its main components, investigates challenges, and proposes potential solutions with regards to individual components. The conducted studies and datasets are systematically presented in Section 4.2. Section 4.3 describes the high-level approach applied across all the studies. Finally, Section 4.4 summarises the chapter.

4.1 Framework for in-network QoE monitoring

4.1.1 Framework overview

The concept of the framework is depicted in Figure 4.1. The framework consists of three main functional components related to the development, deployment and maintenance of QoE/KPI estimation models: (1) *Model training*, (2) *Model deployment (in-network QoE/KPI estimation)*, and (3) *Model re-evaluation and adaptation*. Additionally, the *Configuration and orchestration* interface enables an expert to specify concrete parameters to be used for each use-case (e.g., for on-demand YouTube streaming: run real-time video resolution prediction based on a decision tree model trained on the top 10 features selected using the Sequential Forward Selection (SFS) algorithm from a specified feature set).

The following sections describe possible component implementation choices and configuration options of the conceptual framework. Implementation specifics that work best in practice for a certain service need to be determined experimentally. The framework presented in this section was refined through years of extensive research in the area of in-network QoE/KPI estimation for YouTube, and can serve as a starting point for developing a custom solution for any HAS service. The rest of this thesis (Chapters 5 – 7) is centred around the development and validation of framework components for various use-cases involving the YouTube service, with the focus being on functional components 1 and 3 from Figure 4.1, which provide the grounds for deployment (functional component 2). In that sense, this whole thesis provides a vast amount of resources and guidelines for practical applications, which are relevant for other HAS services as well.

The framework is flexible with regards to differences that may exist among different services and networks, and as such extendable with new use-cases. ML–based QoE/KPI estimation models can be trained to predict any desired KPI, given that it is obtainable by the service performance measurement application. The flexibility extends to the network-level as well. Models can be trained on any attributes available in the network and on any time-scale. These attributes may be network traffic statistics only, or additional data such as information about used platform, signal strength, etc. What can also be considered here, as a solution that is potentially faster in actual deployment, is employing statistics already output by existing network probes, instead of calculating the statistics from the traffic trace. We note, however, that traffic classification and filtering (per service and user) is out of the framework's scope, and assumed to be done elsewhere.

4.1.2 Model training

The *Model training* component takes network- (flow and, if available, radio) and application-level data as input. First, the *Data preparation* module extracts network-level features and ground-truth labels, according to the specification for the service in question. The exact features and labels to be extracted are specified by an expert through the *Configuration and orchestration* interface, separately for real-time and session-level QoE/KPI estimation. The component itself can be used for the training of both types of models, with what is utilised in practice depending on the configuration. Extracted features and labels are fed to the feature selection algorithm (*Feature selection* module), to eliminate irrelevant features and reduce complexity of the models. The algorithm and algorithm parameters that are to be used in this step are specified through the *Configuration and orchestration* interface. Finally, the models are trained by the *Model training and validation* module using the algorithm and parameters selected in the configuration.

Collected data, that is used as input to the model training component, may vary across different services, but also across different use-cases concerning a single service (e.g., different apps used for accessing the content). The approach we describe in this thesis is applicable for any use-case, regardless of network-level differences, as it relies solely on packet size information. However, additional information used as predictors, such as TCP-data (where applicable), context-data, or radio-data available to network operators may improve model performance. Ground-truth application-level data extraction is more challenging in the sense that data is not always obtainable. For example, YouTube offers video performance metrics in their *Stats for Nerds* window, both on desktop and mobile devices (in the browser and in the official app), but a similar overlay offered by Twitch is only accessible in the browser (both desktop and mobile) and not in the app.

The exact features and labels to be extracted from collected data depend on the purpose and desired goals. If the goal is to manage the network traffic in real-time, ground-truth data used for training should reflect the conditions that need to be detected (e.g., detecting that multiple users in a network segment experienced a stalling moments ago requires a model to be trained on short time-windows of network traffic labelled with stalling information). If the goal is to plan and dimension the network to meet users' expectations on a larger scale, estimating QoE on a session-level might be more appropriate. A challenge also lies in identifying network-level features that correlate with these application-level events. Related work on QoE/KPI estimation, discussed in Section 3.3.2, provides information on possible network-level metrics that have proven to be promising on selected use-cases.

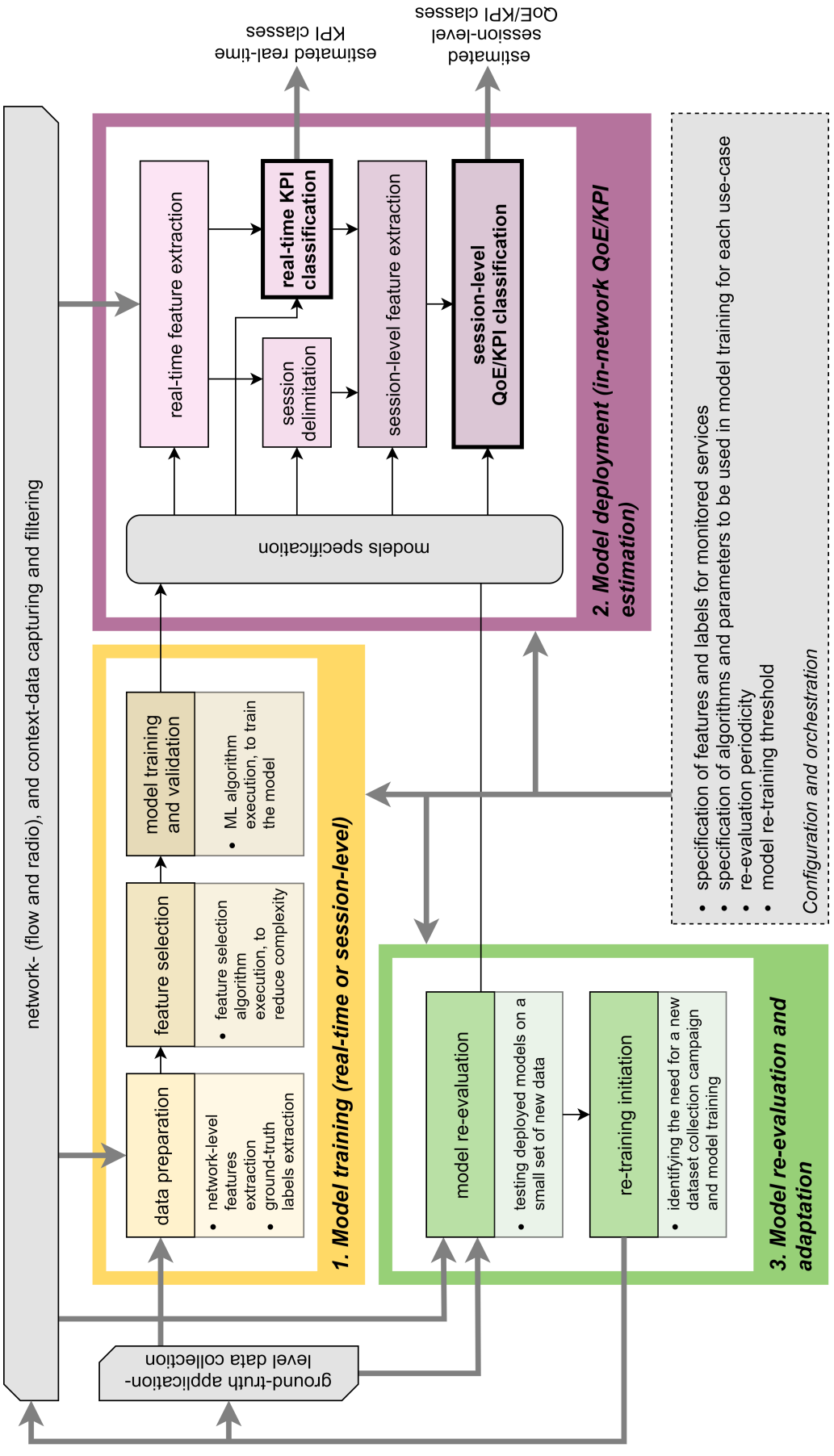


Figure 4.1: Conceptual framework for in-network ML–based QoE/KPI monitoring of HTTP adaptive streaming.

Commonly used ML libraries (such as scikit-learn [166]) offer a lot of valuable ready-to-use tools for feature selection and model training. In our work, we relied on the wrapper approach to select relevant features. This method enables the selection of features that work well together, as opposed to methods that evaluate each feature separately. For model training, we found that models trained using tree-based algorithms, such as Decision Tree or Random Forest, in most cases performed best for this purpose [13, 126, 129, 139, 140, 141, 167]. We also argue that using more lightweight models (such as decision trees instead of random forests) may be better in actual in-network applications, as they may require significantly less resources, while potentially only slightly sacrificing the performance.

4.1.3 Model deployment (in-network QoE/KPI estimation)

The *Model deployment (in-network QoE/KPI estimation) component* is intended to be deployed in the network, and output QoE/KPI estimates on per-video session-level (addressed in Sections 5.1 - 5.3) and/or in real-time (addressed in Section 5.4), based on the network- and context-data input. We note that by 'real-time' we are referring to estimations made over a chosen time interval. While models can be trained to make estimates on a per-second time frame, an operator may decide that making estimations for 5-, 10-, or 20-second intervals may be sufficient. As opposed to QoE/KPI estimations made on a per-session level, 'real-time' estimation approaches may be utilised for real-time resource allocation and optimisation mechanisms.

We assume that the data serving as input for this component has previously been filtered, i.e., we do not explicitly portray traffic classification and the separation of flows corresponding to different users. The component only receives relevant filtered information corresponding to both uplink and downlink data (e.g., packet sizes, timestamps).

The component can continuously track real-time features with its *Real-time feature extraction* module, output real-time KPI outputs, and detect session starts or ends, to be used for session-level QoE/KPI classification. In our real-time models, various network traffic statistics are tracked in 1-second windows for this purpose, but different precisions and additional features may be used as well. Real-time KPIs estimated based on the deployed model can be used as final outputs in this step to take desired actions, such as traffic rerouting or resource reallocation.

Once a video session start is detected by the *Session delimitation* module, the *Session-level feature extraction* module starts aggregating session-level traffic features. As soon as the session end is detected, calculated features can be forwarded to the *Session-level QoE/KPI classification* module, which outputs classes of interest, as defined by the model specification. We note that some statistics of the network traffic cannot be calculated in this way (such as median, which requires full state). If such features are necessary, the traffic information needs to be buffered for the whole session. Additionally, if for a certain use-case both real-time and session-level

QoE/KPI estimation is employed, session-level features can be enriched with real-time KPI estimates, for potentially better performing session-level estimation. However, in that case, models need to be trained with those estimates in mind. We investigate this idea in Section 7.2, but omit further details in Figure 4.1 for the sake of simplicity and cleanliness.

4.1.4 Model re-evaluation and adaptation

The most intuitive way to check if the *QoE/KPI estimation component* is up-to-date is to check if its performance on new data is comparable to the performance achieved during the model training phase. This requires a data collection campaign, to obtain a dataset labelled with ground-truth. The *Model re-evaluation and adaptation component* can initiate a smaller-scale periodical data collection campaign, test the models and report on results. The periodicity of re-evaluation is defined in the configuration, and will depend on the observations regarding a certain service. The component can also be configured to automatically start a larger measurement campaign for the purpose of training new models, when such action is required, i.e., if the model performance is below a certain threshold.

Instead of initiating an intensive dataset collection campaign once the model's performance drops, models can be periodically upgraded by including a smaller number of new labelled instances into the existing dataset and adapting the model. Such an approach has been investigated in Section 7.3 by applying the concept of *adaptive learning*, where we propose the idea of automatically updating a fixed window of instances periodically, and re-training of the models on that fixed number of newest instances.

4.2 Completed studies and data collection campaigns

The overview of conducted studies, together with key objectives, contributions and references to resulting publications is given in Table 4.1. The studies are listed in chronological order, clearly demonstrating the evolution of the approach, with regards to data collection and analysis methods. During the course of this research, numerous datasets were collected for validation purposes. These are listed in Table 4.2 and referenced as parts of the output of separate studies in Table 4.1.

Table 4.1: Overview of completed studies.

Study	Objectives	Outputs
S1, Section 5.1 (2016 – 2017) [12, 13]	• Assessment of the **feasibility of ML–based session-level QoE classification** for encrypted video • Investigation of potential approaches and development of a **prototype solution** • Focus on YouTube on Android	• Ground-truth data collection app called YouQ, based on YouTube IFrame API [116] • Setup of a lab environment with enabled network condition emulation based on IMUNES [168] • Dataset And16L • Detailed methodology and developed tools for data collection, network conditions manipulation, ML analysis • Results that proved the potential of applying ML for in-network QoE estimation for encrypted video
S2, Section 5.2 (2017 – 2018) [14, 15]	• Development of **new ground-truth data collection methods** and comparison to the ones used in S1 • Incorporation of the newly published **ITU-T Recomm. P.1203** [97] QoE model for adaptive video streaming • Instrumentation of bandwidth limitation scripts based on realistic throughput measurements • Adaption of the methodology to be applicable for **QUIC traffic** • Initial tests on mobile-network data • Focus on YouTube on Android	• Datasets And18L and And17M • A version of the ground-truth data collection app YouQ based on YouTube Android API [117] • Ground-truth data collection methodology based on the extraction of *Stats for Nerds* data from YouTube app, referred to as ViQMon • Detailed methodology and developed tools for more realistic data collection and network conditions manipulation, as well as ML analysis applicable for QUIC traffic

Table 4.1: Overview of completed studies.

Study	Objectives	Outputs
S3, Section 5.3 (2017 – 2018) [17]	• Adaption of the methodology to be applicable for the **iOS platform** • **Cross-testing** of Android–trained models on iOS data • Testing of models trained on lab–collected data on **mobile data** • Focus on YouTube on iOS	• Dataset iOS18L, iOS17M, and iOS18M • Detailed methodology and developed tools applicable in the context of iOS • Analysis of the performance of models trained for a specific use-case in comparison to cross-test performances
S4, Chapter 6 (2018 – 2019) [18]	• Investigation of the impact of including different amounts of context data as additional features on model performance • Focus on YouTube on Android and iOS	• Datasets And19L and iOS19L • Analysis of the performance of models that include different context data potentially provided by the service provider • Motivation for OTT-ISP collaboration for QoE management
S5, Section 5.4 (2019 – 2020) [16]	• Development of models for real-time KPI estimation • Focus on YouTube on Android	• Detailed methodology and developed tools for real-time KPI estimation model training, including the definition of example target variables and time-windowed network traffic features • Developed models and the analysis of their performance

Table 4.1: Overview of completed studies.

Study	Objectives	Outputs
S6, Section 5.5, Chapter 7 (2019 – 2020) [16]	• Tackling additional problems related to model deployment: session delimitation (as a prerequisite for session-level QoE/KPI estimation) and model re-evaluation/adaptation (as a response to potential changes in streaming logic) • Exploring the possibilities of training general models (able to address multiple platforms), and the value of the hybrid approach (that utilises real-time KPI estimates in session-level QoE classification)	• Initial promising results with regards to session delimitation and model adaptation • Models able to address multiple platforms and session-level models enhanced with real-time predictions

Table 4.2: Summary of measurement campaigns and collected datasets. The dataset label indicates the used platform, dataset collection year, and the network type (L: lab WiFi network, M: operational mobile network).

Dataset label	**Platform**	**Collect. time**	**Used in study**	**Num. of videos**	**Conditions**
And16L	Android (Samsung Galaxy S6)	Apr–Jun 2016	S1	1060	1060 different YouTube videos played within the YouQ app in 39 different network conditions: constant bandwidth limitations and induced changes
And17M	Android (Sony Xperia X)	Sep 2017	S2	105	105 different YouTube videos played within the YouTube app in an operational mobile network of a European ISP
iOS17M	iOS (Apple iPhone 8)	Sep 2017	S3	128	128 different YouTube videos played within the YouTube app in an operational mobile network of a European ISP
And18L[1]	Android (Samsung Galaxy S6)	Jan 2018	S2, S3, S6	394	100 different YouTube videos played within the YouTube app in 4 different network conditions: 4G, and 4G throughput availability, and 4G throughput availability reduced by factors of 10, 20, and 30
iOS18L[1]	iOS (Apple iPhone 6s)	Jan 2018	S3	380	100 different YouTube videos played within the YouTube app in 4 different network conditions: 4G, and 4G divided by factors of 10, 20 and 30

[1] Datasets available at `https://muexlab.fer.hr/muexlab/research/datasets`.

Table 4.2: Summary of measurement campaigns and collected datasets. The dataset label indicates the used platform, dataset collection year, and the network type (L: lab WiFi network, M: operational mobile network).

Dataset label	**Platform**	**Collect. time**	**Used in study**	**Num. of videos**	**Conditions**
iOS18M	iOS (Apple iPhone 6s)	Mar 2018	S3	111	111 different YouTube videos played within the YouTube app in an operational mobile network of an Asian ISP
And19L[1]	Android (Samsung Galaxy S8)	Feb–Mar 2019	S4, S5, S6	299	100 different YouTube videos played within the YouTube app in 3 different network conditions: 4G, 3G, and available bandwidth capped at 1.5 Mbps
iOS19L[1]	iOS (Apple iPhone 8)	Feb–Mar 2019	S4, S6	299	100 different YouTube videos (same set as in And19) played within the YouTube app in 3 different network conditions: 4G, 3G, and 1.5 Mbps

4.3 Generic approach

The high-level approach taken to train the models spans across all six conducted studies. This section first explains the approach, and then points out the key differences among the studies with respect to concrete data collection and processing methods, and model training methods and algorithms.

4.3.1 Application of ML to the QoE estimation problem

Figure 4.2 illustrates the model training approach used throughout this research. At the client (test device), application-level performance data is collected while playing the videos. At the same time, network traffic is captured in the network. From the application logs, a set of desired KPI/QoE metrics is calculated – these are the targets to be estimated by subsequently trained models. On the other hand, from the network traffic, a set of traffic features is calculated, i.e., statistics that carry information about the patterns in the traffic, such as average throughput, maximum interarrival time between packets, packet size, etc. Depending on the desired temporal granularity of the estimation (e.g., per-session, per-second), traffic statistics and QoE/KPI targets are grouped together to describe separate instances (e.g., sessions, intervals) in the ML dataset.

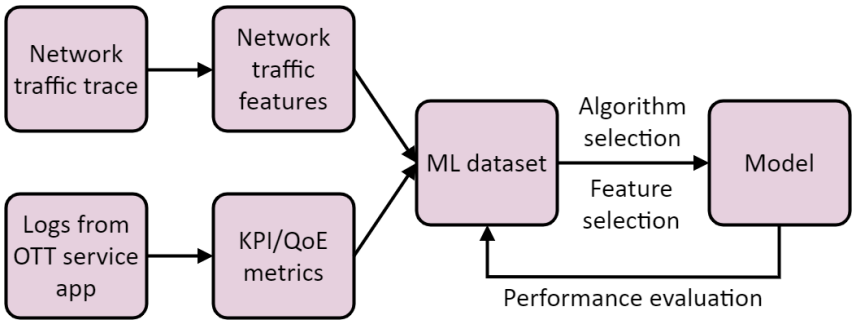


Figure 4.2: Generic approach for ML–based QoE/KPI estimation model training.

Given the complexity of the relationships between traffic patterns and QoE/KPIs, utilising machine learning techniques enables the identification of traffic features that correlate with QoE/KPIs, and the training of models that estimate QoE/KPIs based on a subset of relevant features.

4.3.2 Data collection and processing

While the high-level approach presented in the previous subsection stayed the same throughout the course of this research, the exact data collection procedures evolved. This section gives an overview of this evolution, addressing multiple aspects: data collection environment, ground-truth data collection methods, and network-level data collection and processing. More details on

how the exact tools and methods were used can be found in sections corresponding to concrete studies.

The laboratory setup used for data collection across all studies is depicted in Figure 4.3. A smartphone playing YouTube videos and running a performance measurement app is connected to the Internet through a router whose traffic is routed through the IMUNES [169] node, where bandwidth limitations are imposed. All the traffic passing through the router is replicated with an Albedo Net.Shark [170] portable TAP device and captured using tcpdump [171].

In the first study, Study S1, we collected the dataset And16L while scripting IMUNES to emulate network conditions based on 39 predefined bandwidth profiles (listed in Table 5.2). The profiles included fixed bandwidth conditions, and sudden drops and increases in available bandwidth. Aiming for more realistic network conditions, in studies S2 and S3, for the collection of datasets And18L and iOS18L, we scripted IMUNES to imitate 4G network conditions using the LTE measurement logs [172] published in [173]. As these conditions were generally good and would not result in QoE degradations, in And18L and iOS18L campaigns, we also divided the values in the logs by factors of 10, 20, and 30, so as to reduce the overall bandwidth whilst keeping the temporal bandwidth variability. Finally, for the collection of datasets And19L and iOS19L we used three bandwidth profiles scripted in IMUNES: 4G [172, 173], 3G [174, 175] bandwidth, and bandwidth limited to 1.5 Mbps, thus imitating zero-rate profiles offered by some network operators.

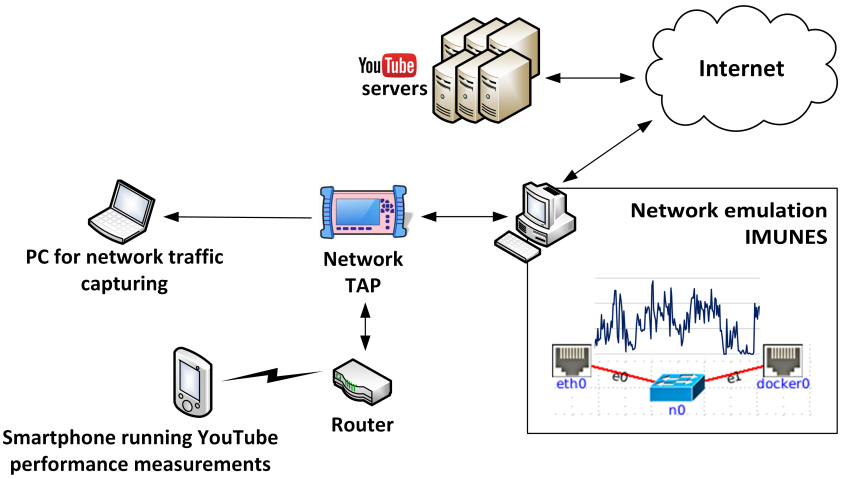


Figure 4.3: Laboratory setup for data collection.

When it comes to obtaining ground-truth application-level data, the approach also evolved to enable collection of more realistic data. For the And16L campaign (Study S1), we used an app called YouQ, developed in our previous research [176, 177] and described in Section 5.1.1. YouQ is based on YouTube IFrame API [116] and has the option to specify parameters of the experiment, including the number of videos to be played and filtering options with regards to video duration, number of views, etc. The filtering was enabled through a connection to our

YouQ server, where we store a database of video IDs and corresponding metadata. However, as YouTube's streaming logic has proven to differ in an IFrame–based app when compared to the official YouTube app [14], we devised the ViQMon approach (Video Quality Monitor), used in all of our following studies (described in more detail in Section 5.2.1). The approach requires recording of the smartphone screen while watching videos in the official YouTube app with the *Stats for Nerds* option enabled and displayed. The *Stats for Nerds* video overlay contains information about streaming performance and can be used to calculate various application-level KPIs. The recordings are later processed using the ViQMon app which relies on optical character recognition (OCR) to extract *Stats for Nerds* data in text format.

As for network-level data and the extraction and calculation of relevant traffic features, the approach applied over the course of our studies changed as a response to YouTube's increasing shift from TLS/TCP to QUIC/UDP. In Study S1, dataset And16L, all videos were delivered via TCP, which means that there were features that correlate with application-level performance that could be extracted at the transport layer. Such features include information about packet retransmission, TCP flags, etc. However, as of Study S2, we observed videos to be exclusively delivered using QUIC. Given that with delivery via QUIC only a minimal amount of information is not encrypted, we resorted to using IP-level features only. The feature set used in Study S2 was significantly expanded with new features in Study S3 and slightly expanded further in Study S4, as acknowledged in Chapter 5. In Study S5, we defined a set of IP-level network features calculated on traffic windows rather than across the whole video session, to be used also for the purpose of real-time KPI classification. This is described more thoroughly in Section 5.4.

4.3.3 Model training methods and algorithms

Our research methods also evolved over the course of conducted studies with respect to applied machine learning methods used for feature selection and model training, as well as with respect to defined targets. While in all studies we rely on supervised learning, specifically casting the problem of QoE estimation as a classification problem (e.g., classifying MOS into 2 or 3 classes, average video bitrate into 2 or 3 classes), the definition of the classes changed over time. In the initial study (Study S1), we make use of custom QoE classes devised in [176] to be used as targets. The classes were defined based on fuzzy logic and related work on the correlation of individual influence factors with QoE [91, 92, 93, 94]. Upon the publication of ITU-T Recomm. P.1203 [97], from Study S2 onward, we used the model published in the recommendation for the calculation of ground-truth QoE (studies S2–S4, S6). Starting with Study S2, we also considered KPI estimation models, concretely for the estimation of played resolution, stalling, initial delay, and video bitrate. In Study S5, which revolves around real-time KPI estimation, the focus was on video resolution and bitrate, as stalling was rarely observed in newer datasets.

With respect to used ML algorithms, in earlier studies we used different algorithms imple-

mented in Weka [178], namely OneR, Naïve Bayes, SMO (Sequential Minimal Optimization), J48 (decision tree), and Random Forest. As related work on the topic of ML–based QoE estimation for video streaming was scarce at the time, and there were limited insights into which algorithms may work well for this problem, during the initial study we experimented with a large variety of algorithms implemented in Weka, and decided to focus on the aforementioned five, for multiple reasons. OneR, a simple one-attribute (feature) rule, was chosen to serve as a baseline, to be able to better evaluate models that employ a number of features in a more complex manner. Naïve Bayes was included so as to test how a simple, fast, probabilistic algorithm would perform on this problem. Aiming for more complex algorithms, we included SMO, an algorithm that, by using heuristics, solves the SVM (Support Vector Machine) training problem (finding a hyperplane that best separates the instances of different classes in n-dimensional space, where n is the number of features). Finally, we investigate tree-based algorithms, namely J48 and Random Forest (an ensemble of trees), which both performed very well in initial tests. Seeing that tree–based algorithms yield best results, with related work also confirming this statement [126, 129, 139, 140, 141, 167], we have focused our attention on decision trees and random forests in later studies, continuing to use the implementations in Weka through Studies S2 and S3, and later using the implementations in scikit-learn Python library [166].

Prior to model training, we first balance out the number of instances across classes and then run a feature selection algorithm to identify relevant features which will be used by the model. In Study S1, there was no need for up/downsampling (also referred to as over/undersampling), as data was repeatedly collected in different network conditions so as to keep the sample count in each class roughly in balance. In studies S2 – S6 we subsample the number of instances in line with the least populated class. The exception is the classification of MOS in studies S4 and S6, as there are only 30 samples in one of the classes in dataset And19L. In that particular case we exploit SMOTE (Synthetic Minority Oversampling Technique) [179] from the Imbalanced Learn Python library [180]. SMOTE selects a random sample in the minority class, finds its k nearest neighbours, randomly selects a neighbour, and creates a synthetic sample along the line connecting the two selected samples. In our studies, in cases when we used SMOTE (MOS classification into 3 classes), we generate synthetic instances only for the least populated class, while we randomly subsample the most populated class, with both matching the amount of samples in the third class.

In order to reduce computational cost and noise from irrelevant features, in all studies we performed feature selection. As for exact methods, through studies S1–S3, we have used a wrapper method implemented in Weka, employing the greedy Best First algorithm. As opposed to filter feature selection methods, that evaluate the relationship between individual input variables and target variable, wrapper methods train and evaluate models trained on subsets of features in order to assess the relationship of different combinations of features with the target

variable. Such methods are computationally more complex, but may eventually lead to better performing models. In studies S4 – S6, we used the wrapper method Sequential Feature Selection (SFS), implemented in mlxtend library for Python [181]. Additionally, in S5, we also train models on features selected based on feature importance. As this method is significantly less computationally intensive (it only trains one model on all features and outputs the ones most utilised by the model to be used in actual model training), in the study we assess the advantages and disadvantages of the two approaches.

4.3.4 Model performance and evaluation

When it comes to model training and evaluation, the dataset is typically split into 3 parts – train, validation, and test set (unless there is another, separate, test set, then the aforementioned dataset can be split in two). In ideal circumstances, the model would be iteratively trained and validated on the validation set, with ML algorithm hyperparameters tuned in each iteration, and finally tested once on the test set, to assess the performance and model generalisation (i.e., performance on unseen data). However, while optimising the performance of one particular model for one particular purpose is a priority in many domains of application, ML–based solutions for network traffic monitoring need to be simple, lightweight and possibly the processes of training and validation need to be automated. This is why related work on QoE/KPI estimation mostly does not consider the evaluation of different hyperparameter settings, but rather just reports on the ones used (e.g., the number of trees in random forest). In this case, having a separate validation and test sets is not needed, as the model is tested only once on the validation/test set.

So far, most of the work in the area of in-network QoE/KPI estimation has been focused on feasibility studies showcasing novel methodologies and different aspects of the problem: session-level QoE/KPI estimation, real-time QoE/KPI estimation, feasibility of addressing multiple platforms/services/networks with a single model, etc. In that regard, cross-validation gives a good indication of performance, in conditions where there is often a limited number of samples available. With cross-validation, the dataset is split into k parts, and k models are trained, while using $k-1$ parts for training and one part for validation (k-fold cross-validation). The performance reported in the end is the average of performances of all k models. Given the aforementioned domain specifics, most of the related work reports the performance achieved through cross validation [129, 140, 182], or through tests on a single dataset split [126, 128]. The performance results reported in our studies S1 – S3 correspond to results achieved with 10-fold cross-validation, with the exception of models tested on separate datasets collected in mobile network. In studies S4 – S6, we report results achieved on a 67:33 dataset split (randomly selected 67% of data used for training, 33% for testing).

In Study S1, we report on the accuracy of the models, while providing more insight into the model performance by supplying the confusion matrices and visualisations of misclassifi-

cations. Besides the accuracy, in studies S2 – S6, we report precision and recall for each of the classes in the model. Similar metrics have been used in related work as well. We briefly explain these performance metrics:

- Accuracy – the percentage of correctly classified instances,
- Precision – the number of true positives over the number of all instances classified as observed class,
- Recall – the number of true positives over the number of all the instances that should have been classified into observed class.

4.4 Chapter summary

During the time from 2016 to 2020, this research has dealt with the problem of QoE/KPI estimation from different perspectives, e.g., focusing on QoE/KPI estimation for Android vs. iOS, aiming to estimate QoE/KPIs on a session-level vs. estimating KPIs in real-time. The research was conducted across six studies with different objectives addressing a variety of use-cases. We note that in all cases, we focused on videos streamed to mobile devices (namely smartphones), and relied on YouTube as one of the largest video delivery platforms currently available on the market. Furthermore, while we focus on the delivery of VoD content, with in-depth studies involving QoE monitoring of real-time live streaming considered out of scope, we note that the general methodology related to model training and evaluation is applicable for live streaming as well.

The studies we report on yielded nine datasets, used in different combinations so as to address the objectives of individual studies. This chapter has presented the overall methodology through a depiction of a framework for ML–based QoE/KPI monitoring of HAS, given an overview of the objectives, studies, and datasets, and was concluded with a summary of similarities and differences among the conducted studies. The generic framework specified in this chapter brings together the different components involved in developing and deploying an in-network QoE/KPI estimation solution for adaptive video streaming. While the actual deployment of an overall solution in an operational mobile network is beyond the scope of this thesis, key components of the framework are applied across the conducted studies reported in Chapters 5 – 7.

Chapter 5

Machine learning based models for in-network estimation of QoE/KPIs of adaptive video streaming

After having introduced the high-level methodology used in this research, this chapter provides an in-depth examination of studies aimed at training in-network YouTube QoE/KPI estimation models. Our initial feasibility study on session-level QoE/KPI estimation for Android is described in Section 5.1. The following sections describe novelties introduced over the course of our research with respect to the data collection and model training methodologies, and present new models for Android (Section 5.2) and iOS (Section 5.3). Moreover, we present results related to model cross-testing (platform- and network-wise). Finally, in (Section 5.4) we address the problem of real-time KPI estimation, with results focused on videos streamed using the Android platform. The chapter is concluded with a summary in Section 5.6.

5.1 Initial feasibility study

Our initial study, conducted in the period from 2016 – 2017 and published in [12, 13], was the first step in developing the methodology and models for estimating YouTube per-video QoE based solely on the analysis of encrypted network traffic. The results of the study proved the feasibility of the approach and provided a solid base for future research, presented in the continuation of this thesis.

In the study, we leverage advances in traffic classification and feature extraction methods [183, 184, 185], as well as existing at the time QoE models for adaptive video streaming [90, 91, 92, 93, 94], to propose an ML–based approach for classifying video traffic features into QoE classes. For instrumentation purposes, we utilised a system which we previously developed in the scope of [176, 177] called YouQ, which enables the collection of a training

dataset based on monitored client-side application-level KPIs and network traffic traces. Our YouQ Android application plays a requested number of randomly chosen YouTube videos and logs various quality–related events (on a per-video basis). These events are then used to classify each YouTube video streaming session into one of three QoE classes: “high”, “medium” or “low”. A traffic feature set is computed from the traffic trace of every video streaming session and is labelled with a QoE class, to be used as input in the form of a training dataset for ML algorithms.

5.1.1 Developed instrumentation

YouQ system for data collection

The YouQ system consists of an Android application and a server that supports the experiment processes. The YouQ Android application (Figure 5.1) is run on the client and monitors application-level data that is used to calculate various QoE–related KPIs (number of stalling events, average stalling duration, percentage of playback time spent on certain quality level, etc.). The application enables a test administrator to select a target number of videos to be played during an experiment (e.g., run experiment with 100 different videos). Filtering options enable the administrator to request that the duration of each played video falls within a set interval (e.g., play only videos that last between 60 and 120 seconds), that the videos have been viewed a certain minimum number of times, and that the videos are offered in a certain playback quality (high definition or standard definition). By choosing only hd (high definition) videos, the test administrator can ensure that all of the videos are available at least in 480p resolution, if not higher. Video IDs to be played during the experiment are retrieved from the YouQ server’s database, which we populated with over 2 million YouTube video ID entries, as described later in this section. Throughout the entire experiment during which the chosen number of videos are played (note that all played videos are different), the application collects data using the YouTube IFrame API, into three log files, as described in Table 5.1. We note that the labels for each quality level used in this study are defined by YouTube, and only refer to video resolution, as opposed to quality levels typically being associated with video representations (video resolution, bitrate, codec, etc.).

The *YouQ server* fully supports the measurement procedure, as well as processing of the measurement data. The key components of the server are listed as follows:

- database server – for retrieving video IDs based on the request for an experiment set up by the user from the YouQ Android app,
- scripts for data processing – for the calculation of network traffic features, application-level KPIs, and QoE class labels, and

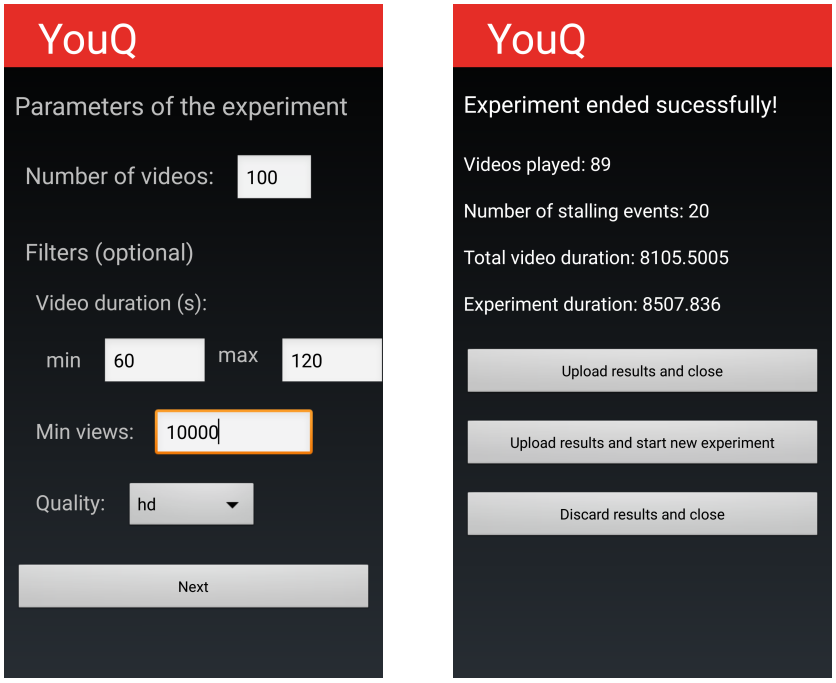


Figure 5.1: YouQ Android application.

Table 5.1: Data collected with YouQ Android application.

Log	**Description**
Event log	YouTube events: “Buffering”, “Playing”, “Paused”, “Ended”, “hd1080” (quality level playback has switched to), “hd720”, “large” (480p), “medium” (320p), “small” (240p), “tiny” (144p)
Buffer log	Amount of video buffered in every second of watch time.
URL log	URLs from all HTTP requests towards YouTube servers.

- YouQ Web application – for displaying experiment results, and analysis of model performance.

When the experiment is over, the data collected at the client device (application-level logs) and in the network (network traffic) can be uploaded to the YouQ server, where it will be processed. The output of these scripts are files with selected fields from captured network traffic for each video from the experiment and files with application KPIs for every video in each experiment. Scripts are also used to populate a database with uploaded information for the purpose of data visualisation in our YouQ Web application (Figure 5.16). What follows is the calculation of traffic features from logs with relevant packet fields, and the calculation of QoE classes from application-level KPIs, which finally forms the ML dataset.

Additionally, prior to running any experiments, we implemented an application for filling the database with YouTube video IDs and metadata, to enable the retrieval and filtering of a target number of video IDs to be used for running experiments. The application implemented

for that purpose randomly generates a string of 4 to 5 characters, and then uses the YouTube Data API [186] to query YouTube, as is normally done by using YouTube's search engine. Results that YouTube returns are processed and stored into the database. For each video, the following data is stored: videoID, title, duration, quality, number of likes, number of dislikes, and number of views.

Laboratory setup and measurement procedure

To collect the data necessary for building a model that classifies YouTube video streaming sessions into QoE classes, we set up a laboratory environment as depicted in Figure 5.2. As opposed to our subsequent studies, that used the setup depicted earlier in Figure 4.3, this setup additionally includes a connection between the router and IMUNES node, as well as the YouQ server. Contrary to subsequent studies, where bandwidth profiles are set up in a loop, in this study bandwidth envelopes were re-initiated at the beginning of every video. Using the additional connection, the IMUNES node can be accessed by the YouQ client application at the beginning of each video to perform this action automatically.

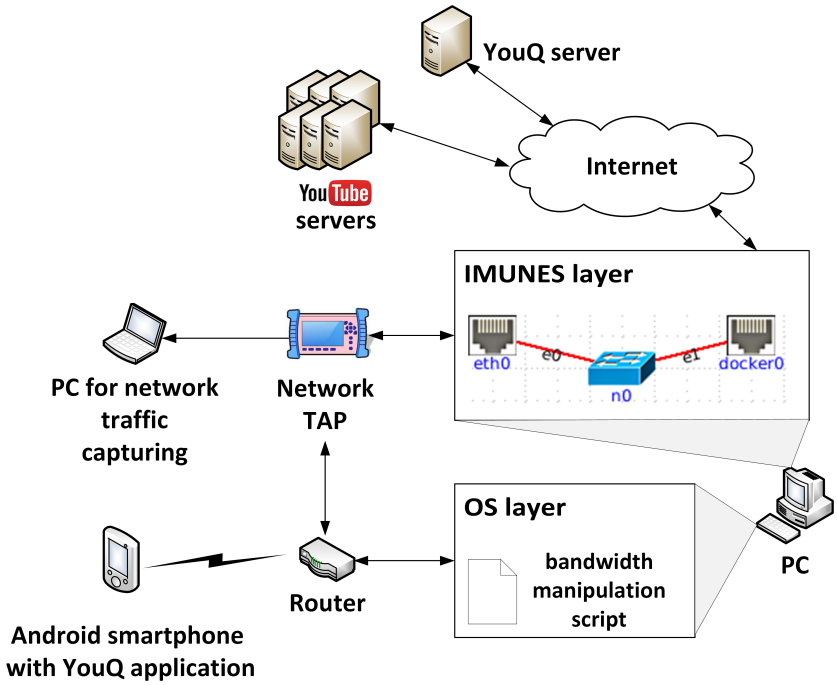


Figure 5.2: Laboratory testbed.

Figure 5.3 shows the actions a test administrator has to take when conducting an experiment. It also shows the output that every action results with. A challenge when running tests was to obtain video streaming traces with various application-level events being observed, such as stalling and quality adaptation. For that reason, different bandwidth limitations were imposed (using so-called bandwidth envelopes) so as to invoke such events. At the beginning of the experiment, a script is run to set a bandwidth envelope on the IMUNES node. The script enables

the scheduling of bandwidth changes. For example, at the beginning of a video streaming session available bandwidth is set to 5 Mbps, and after 2 minutes it is decreased to 3 Mbps. This scenario is reset for every video in the experiment automatically. Before starting the YouQ Android application, network traffic capturing must be manually started on the dedicated PC. When started, the YouQ application asks for the parameters of the experiment. An administrator can choose the number of videos to be played during the experiment, and optionally can specify the duration of the videos to be played (e.g., play only videos lasting between 10 and 20 sec), the minimal number of views, and available video quality (note that this value is set to either "sd" or "hd" as this metadata is offered via the YouTube Data API). After all the videos are played, the administrator can upload the collected data or discard it. Traffic capturing can be stopped after all the videos are played, and the traffic trace can be transferred to the server, where both application and network-level data are processed.

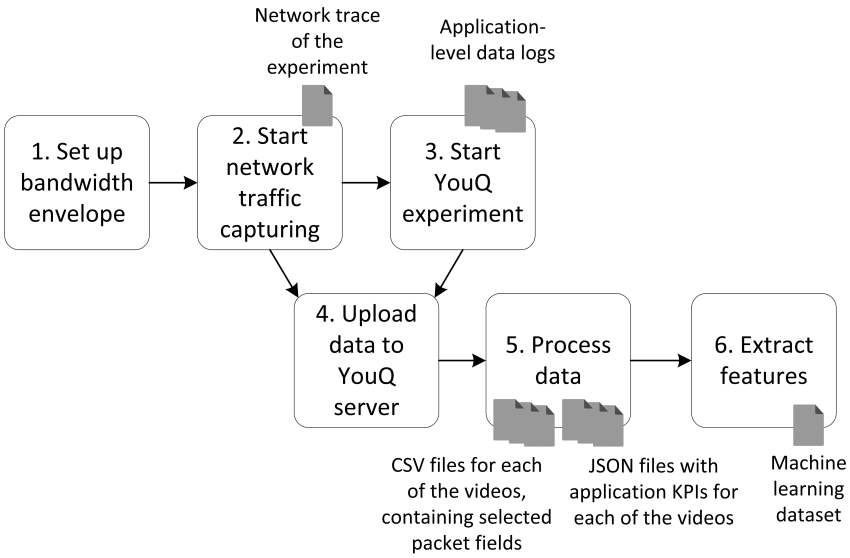


Figure 5.3: Experiment actions.

5.1.2 Analysis of YouTube's adaptation algorithm

An analysis of YouTube's adaptation algorithm was a prerequisite for conducting automated experiments. By analysing how YouTube reacts under various network conditions, it was possible to subsequently define experimental scenarios that would lead to quality switches, stalling events, etc. Hence, we could define experiments designed to collect diverse training samples and cover a large area of possible scenarios differing in application-layer metrics, and consequently user perceived quality. We concluded that manipulation of bandwidth was a key factor for inducing different QoE in videos played in each experiment.

Similar research was conducted by Wamser *et al.* [38], where the authors stated that both the flow control at the network level and the user QoE are directly related to the buffer level.

While their tests were run on PC devices, our focus has been on mobile devices with the Android operating system. Findings regarding the YouTube adaptation algorithm and buffer level thresholds are similar and are described in the following text.

Effect of bandwidth availability on chosen quality levels.

Since network bandwidth greatly influences the quality in which a video is played, the goal of this step was to determine bandwidth levels at which the YouTube adaptation algorithm forces certain quality levels. Experiments were run on 24 different bandwidth settings for each of 20 selected videos. All selected videos were available in each resolution from 144p to 1080p and were approximately 3 minutes long. We note that this is a dataset used solely for this purpose, and thus not references in the dataset list provided in Table 4.2. Figure 5.4 shows the results of this phase, i.e., the percentage of videos played in each quality level, and for each bandwidth setting.

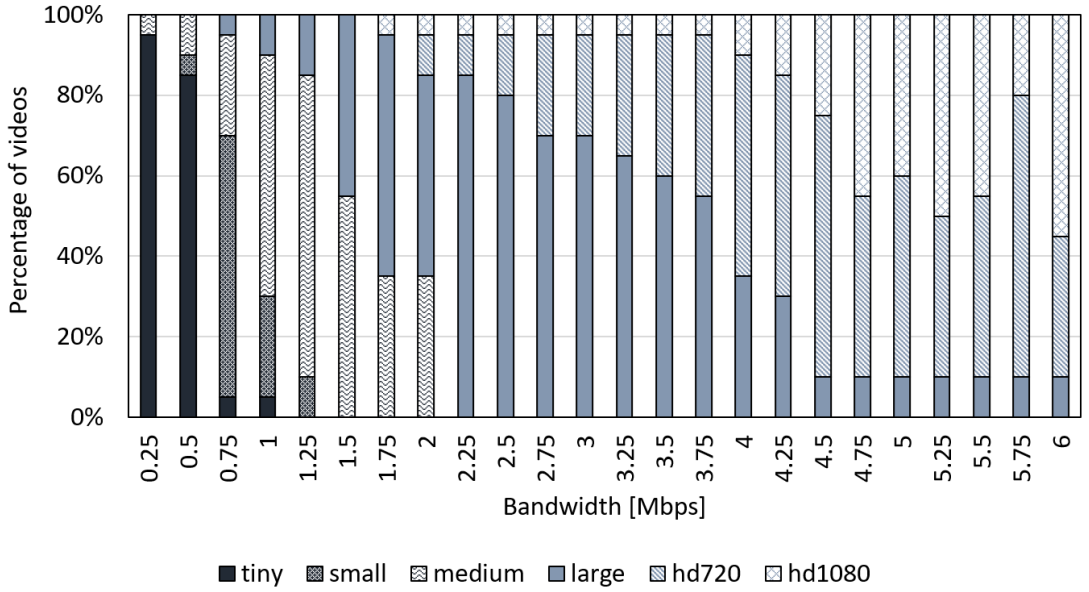


Figure 5.4: Effect of bandwidth on quality level.

It can be observed that video quality is strongly dependant on the available bandwidth. Although different videos may result in different playback qualities on the same bandwidth settings, due to the different dynamics of video content, this analysis enabled us to define experiments that are expected to result in different quality levels. However, in certain cases we were not able to explain some observed irregularities. As most of the YouTube videos available in the "hd1080" quality level are not retrieved at this quality level on smartphones, prior to this analysis, we ran experiments without bandwidth limitations, with measured bandwidth higher than 100 Mbps, to collect 20 video IDs that switch to "hd1080", even on mobile phones. What we could not explain is why 2 of the 20 chosen videos later never switched to HD quality levels, nor why the experiment with bandwidth set to 5.75 Mbps resulted with an unexpected

distribution of quality levels. This might be due to CDN-related issues such as video location or server load, but a detailed analysis on a larger number of videos is needed to draw any clear conclusions regarding this issue.

To analyse buffer thresholds and the amount of buffered video content over time, we used the YouQ Web application for data visualisation. One of the features of this application is visualisation of the buffer state portraying the duration (in seconds) of remaining video in the buffer in each second of watch time. Based on those graphs, many regularities were noticed. Experiments were conducted using both constant and variable network bandwidth.

Buffering under constant bandwidth.

With available network bandwidth set to a high and constant value, typical DASH behaviour can be noticed. During the initial phase of playback, the YouTube media server sends a large amount of data so that the buffer can be quickly filled to a certain threshold and be ready for video playback (*burst phase*). After that, new chunks of video are requested once the video is played and the buffer level drops (*throttling phase*). This behaviour is visible in Figure 5.5.

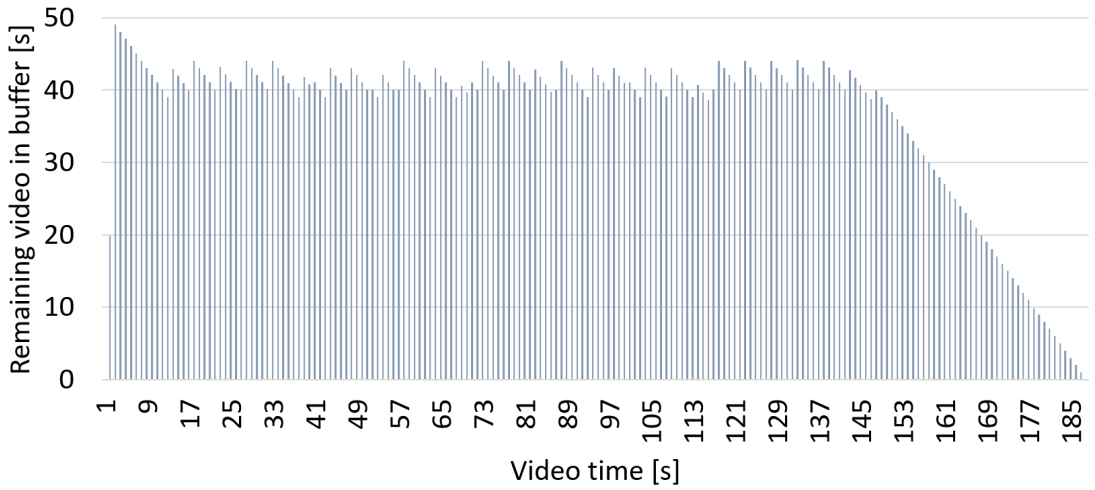


Figure 5.5: Amount of buffered video while playing at the quality level *hd1080* (bandwidth limitation not set).

In the second phase of video playback, during the throttling phase, regularities regarding buffer thresholds can also be derived. The buffer **upper threshold** is the maximum amount of unplayed video in the buffer, while the buffer **lower threshold** is the minimum amount of unplayed video in the buffer (note thresholds refer to seconds of video playback, and not to amount of data in bytes). When the buffer reaches the upper threshold, the YouTube algorithm stops requesting new video segments, and when the buffer reaches the lower threshold, video delivery is again requested.

For videos played in *hd1080*, the upper threshold was observed to be consistently around 45 seconds, while the lower threshold was around 40 seconds. Those values rarely change, but for

videos that are encoded with 60 FPS (frames per seconds) they are 10 seconds lower, 35 and 30 seconds respectively. Videos played in *hd720* were observed to have higher thresholds. The upper threshold is approximately 80 seconds, while the lower threshold is around 75 seconds. Videos played in quality level *large* were found to have an even higher threshold. The upper threshold is around 155 seconds, while the lower threshold is around 150 seconds. Videos played in *medium* quality were observed to have an upper threshold of around 280 seconds while the lower threshold is around 275 second. These results indicate that the buffer stores a predefined amount of video in bytes (and not in time). This confirms the findings published in [38].

Figure 5.6 shows the amount of buffered video over time, in the case when bandwidth is just slightly higher than needed for YouTube's algorithm to request an *hd1080* video (4.75 Mbps). It can be observed that the amount of buffered video still reaches the same aforementioned thresholds, but much slower.

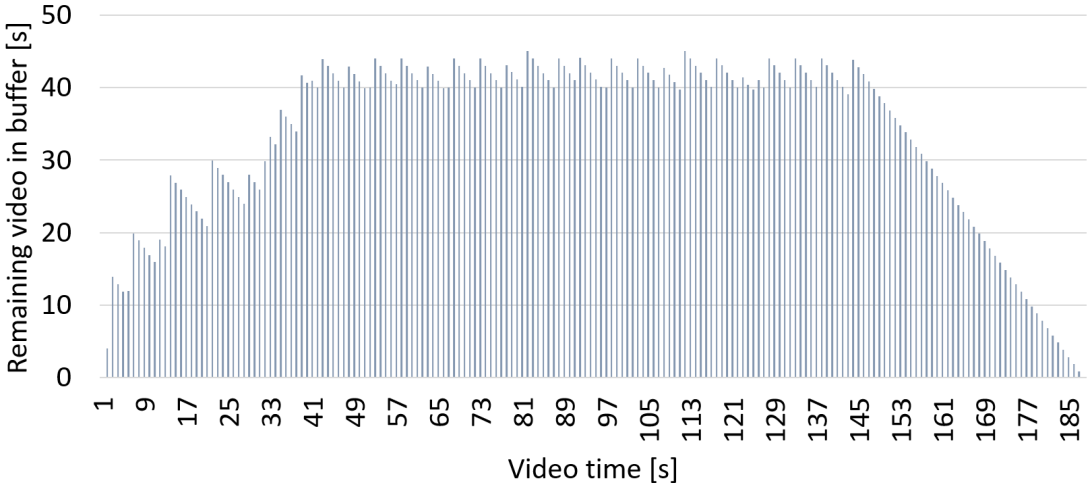


Figure 5.6: Amount of buffered video while playing at the quality level *hd1080* (bandwidth limitation set to 4.75 Mbps).

Buffering under variable bandwidth.

To determine how YouTube's adaptation algorithm manages quality switches, we analysed buffered video graphs in the cases when available bandwidth is increased or decreased during the video streaming session. We note that in related work, Sieber *et al.* [187] observed that in case of varying network conditions, YouTube client demands different video qualities in parallel in order to adapt to the network situation. The amount of redundant traffic has been shown to depend on the content (a greater variability in encoding bitrate seemed to lead to more frequent adaptation events). YouTube's behaviour regarding redundant traffic is consistent across different videos. Our aim was to classify QoE by looking into aggregate traffic. Thus, we do not differentiate between redundant and non-redundant traffic. If YouTube changes its strategy

in terms of redundant content delivery, we can use the YouQ approach to build new models.

From higher to lower bandwidth. When bandwidth changes from a higher to a lower value, the YouTube player was observed to first play out video in high quality that is already in the buffer, while video in a lower quality will not be buffered until the buffer is nearly empty and further buffering is required to prevent stalling events. Figure 5.7 shows buffer state in case of lowering network bandwidth. The player started the video in *medium* quality, and then, after measuring the available bandwidth (set to 5 Mbps), switched to *hd720* and then to *hd1080* in the 24th second. The bandwidth was lowered to 1 Mbps in the 30th second. When the player detects the lower bandwidth, it plays the buffered high quality video and pauses buffering. Because bandwidth stayed at 1 Mbps and further waiting would result in stalling, YouTube's algorithm switches the playback to quality level *large* in the 56th second, to quality level *small* in the 67th second and to quality level *medium* in the 77th second. Peaks in the buffered video amount can be observed a few seconds before these switches. Although the quality level was set to *medium* from the 77th second to the end, buffer thresholds towards the end of the video do not correspond to those for video played in *medium* quality, but rather those for video played in *1080p*. The hypothesis is that the YouTube player, after measuring high bandwidth at the beginning of playback, assumes that low network bandwidth is only a temporary network problem, and does not buffer a lot of low quality video content. This may be due to it being common for those problems in the network to be eliminated fast, in which case buffering video in high quality could be continued, while smaller amounts of buffered video in low quality could be dropped. We note that these are just assumptions based on the observed behaviour.

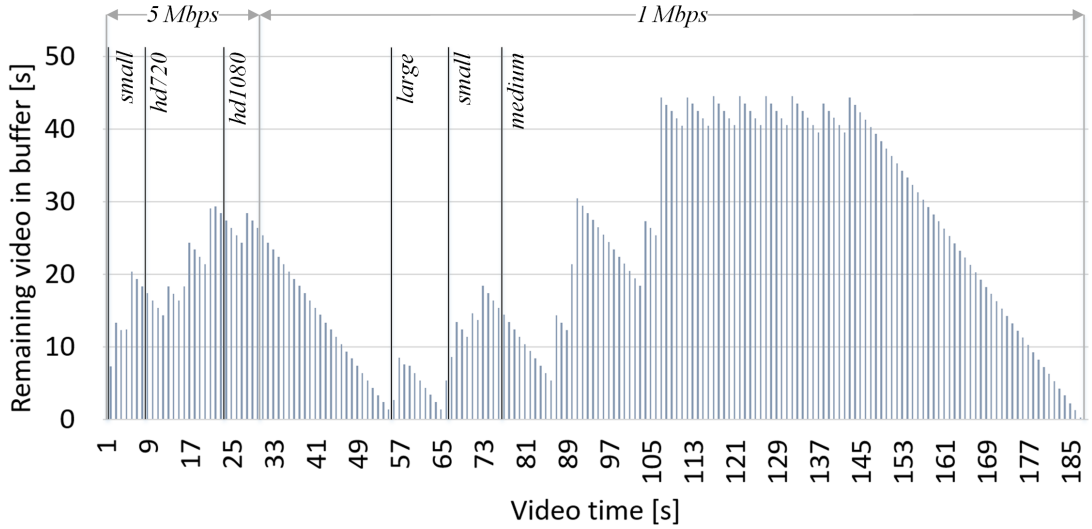


Figure 5.7: Buffer state when network bandwidth is lowered from 5 Mbps to 1 Mbps in the 30th second.

From lower to higher bandwidth. The second scenario we considered is when network bandwidth availability is increased during video playback. Our observations show that at the

moment when an increase in network bandwidth is detected, the YouTube player requests a larger amount of video in higher quality (*late burst phase*), and then stops requesting content until the buffer reaches the threshold for that quality level. When that happens, the buffer enters a throttling phase and stays there until the end of video playback. The graph that represents buffer state when bandwidth changes from a lower to a higher value is shown in Figure 5.8. In the beginning of the experiment, bandwidth is set to 1 Mbps and video is played in quality level *medium*. In the 30th second, the bandwidth increases to 5 Mbps and when the player detects the change, it starts requesting video segments in quality level *hd720* and switches the playback to that quality level in the 49th second. In the 61st second, the playback switches to *hd1080*. Drops in the amount of video in the buffer are also observed. The assumption is that they represent the dropping of previously buffered video in lower quality. While Figures 5.7 and 5.8 depict only a single run for illustration purposes, we highlight that the behaviour we observed and which is described in further text was found to be consistent across all of our experiments with a bandwidth increase or decrease.

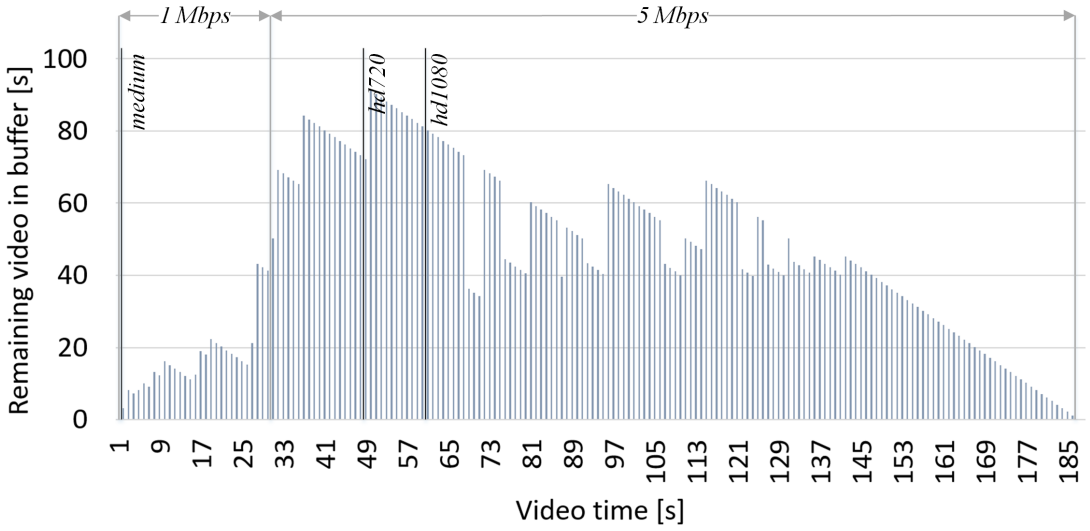


Figure 5.8: Buffer state when network bandwidth is increased from 1 Mbps to 5 Mbps in the 30th second.

5.1.3 Data collection and processing

To collect the And16L dataset, experiments with different bandwidth limitations were conducted (each experiment involved multiple videos being played). The bandwidth envelopes were derived based on prior analysis of YouTube buffering behaviour and adaptation logic, as described in the previous section. And16L consists of 1060 instances, each related to one YouTube video. Each instance is described with 33 features, and labelled with a QoE class. Characteristics of the application layer data, used to calculate QoE class, and feature selection process are described later in this section.

Experiments

We ran a total of 39 experiments with different bandwidth limitations. The goal was to cover a large number of adaptation and playback scenarios differing in played quality levels, stalling number and durations, etc. Each experiment is defined by its bandwidth envelope and video duration range requested in the YouQ Android application. Table 5.2 gives an overview of experiments with their bandwidth envelopes, number of videos played in each experiment, and selected video duration. If bandwidth limitation was not constant during the experiment, the moment of change is given on the arrow. The number on the arrow corresponds to the number of seconds passed from the initial buffering event of the currently playing video until a new bandwidth limitation was set using IMUNES. For example, the experiment with ID 16 was run as follows: at the time when video playback was initiated, available bandwidth was limited to 3 Mbps. After 60 seconds, this limitation was dropped to 1 Mbps, and after another 60 seconds it was again dropped to 0.5 Mbps.

Videos in the experiments were chosen randomly to obtain a realistic and diverse dataset for model training and testing, rather than repeating tests with a small number of chosen videos (as has been commonly done in previous studies [112]). With respect to the bandwidth patterns that we used for our experiments, besides constant bandwidth that is most commonly used in QoE modelling, we also include bandwidth patterns with fluctuations, to enable assessing QoE in the event of outages or congestion that might happen in real network environments [188]. Patterns were also chosen so as to balance the number of instances in each QoE class, using the knowledge about YouTube's adaptation strategy.

Table 5.2: Experiment scenarios used in the collection of dataset And16L.

Exp. no.	**Bandwidth envelope** $[Mbps]$	**Number of videos**	**Video duration** $[s]$
1	5	9	600-650
2	3	9	600-650
3	1	9	600-650
4	5	16	180-600
5	3	9	180-600
6	1	18	180-600
7	$1 \xrightarrow{60} 5$	14	180-600
8	$1 \xrightarrow{120} 5$	18	180-600
9	$1 \xrightarrow{240} 5$	5	480-600
10	$0.5 \xrightarrow{60} 3$	17	180-600
11	$0.5 \xrightarrow{120} 3$	16	180-600

Table 5.2: Experiment scenarios used in the collection of dataset And16L.

Exp. no.	**Bandwidth envelope** $[Mbps]$	**Number of videos**	**Video duration** $[s]$
12	$0.5 \xrightarrow{240} 3$	4	480-600
13	$3 \xrightarrow{60} 0.5$	14	180-600
14	0.5	16	150-180
15	$1 \xrightarrow{60} 0.5$	18	180-600
16	$3 \xrightarrow{60} 1 \xrightarrow{120} 0.5$	16	180-360
17	$0.5 \xrightarrow{60} 1 \xrightarrow{120} 3$	13	180-360
18	$0.5 \xrightarrow{60} 10 \xrightarrow{120} 0.5$	20	360-600
19	$0.5 \xrightarrow{90} 2$	26	180-200
20	$0.5 \xrightarrow{60} 1 \xrightarrow{120} 0.5 \xrightarrow{180} 1 \xrightarrow{240} 0.5$	5	300-360
21	0.3	27	180-200
22	$10 \xrightarrow{120} 1$	26	300-600
23	$0.3 \xrightarrow{30} 1 \xrightarrow{90} 0.5 \xrightarrow{150} 8 \xrightarrow{165} 1 \xrightarrow{225} 0.4$	18	240-600
24	7	42	240-600
25	0.7	18	120-420
26	1.5	17	120-420
27	0.8	19	120-420
28	2	17	120-130
29	4	44	120-420
30	3.5	47	120-420
31	2.7	48	60-180
32	7	25	120-300
33	1.3	43	120-300
34	0.6	43	120-300
35	0.5	45	120-300
36	0.4	56	60-120
37	0.55	89	60-120
38	$0.85 \xrightarrow{45} 0.75$	89	60-120
39	0.35	75	60-120

Feature extraction

Numerous previous studies have addressed the goal of traffic classification based on feature extraction and analysis [185, 189, 190, 191]. In [192], Callado *et al.* explain the main problems of

Internet traffic identification and potential solutions. They also present trends in this field and provide an overview of techniques for traffic analysis. Similarly, in [183], Nguyen and Armitage discuss the motivation for the application of ML to Internet traffic classification and give a comprehensive survey of relevant studies. Although the described studies calculate network traffic features and use ML algorithms for achieving traffic classification, the general approach can be applied to the problem of classifying traffic related to one service into QoE classes. Hence, in addition to considering features commonly used for traffic classification, we also analysed the behaviour and the YouTube client algorithm and corresponding traffic characteristics so as to identify additional potentially relevant features.

Following an analysis of related work on feature extraction [136, 184, 193] and also our numerous observations of YouTube adaptation behaviour, we comprised a set of 54 features, which can be divided into six categories: packet length statistics, size of transferred data in 5-second intervals, packet count statistics, interarrival time statistics, throughput statistics, and TCP flags count. Features were extracted for all captured network traffic that belongs to a single video by inspecting headers of each packet. Calculated network traffic features were inspected using the WEKA ML library [178] and further reduced to 33, primarily based on redundancy. Eliminated features include summarised values, such as overall *packetCount*, that is highly dependant on the duration of the video. Another set of eliminated features is comprised of features that are functionally dependant on other features (duplicates). Features like *maxPacketSize*, that have the same value for all of the videos, were also removed. The final list of features is given in Table 5.3.

Table 5.3: List of all the extracted features. The features are calculated for the network traffic captured during the watch time of each video.

Feature name	Feature description
avgPacketSize	Average packet size [$bytes$]
averageSizeThroughTime	Average of sizes of transferred data per 5 s interval [$bytes$]
minimalSizeThroughTime	Minimum of sizes of transferred data per 5 s interval [$bytes$]
maximalSizeThroughTime	Maximum of sizes of transferred data per 5 s interval [$bytes$]
sizeThroughTimeStdDev	Standard deviation of sizes of transferred data per 5 s interval [$bytes$]
sizeThroughTimeMedian	Median of sizes of transferred data per 5 s interval [$bytes$]
averageInterarrivalTime	Avg. packet interarrival time [s]
minimalInterarrivalTime	Min. packet interarrival time [s]

Table 5.3: List of all the extracted features. The features are calculated for the network traffic captured during the watch time of each video.

Feature name	Feature description
maximalInterarrivalTime	Max. packet interarrival time $[s]$
avgInterarrivalTimeThroughTime	Avg. of interarrival time averages per 5 s interval $[s]$
interarrivalTimeThroughTimeStdDev	Standard deviation of interarrival time averages per 5 s interval $[s]$
interarrivalTimeThroughTimeMedian	Median of interarrival time averages per 5 s interval $[s]$
effectiveThroughput	Avg. of avg. throughput values calculated per 5 s intervals, including only those intervals where throughput per interval was higher than 0.3 Mbps $[Mbps]$
minThroughputThroughTime	Min. of avg. throughputs per 5 s interval $[Mbps]$
maxThroughputThroughTime	Max. of avg. throughputs per 5 s interval $[Mbps]$
throughputStdDev	Standard deviation of avg. throughputs per 5 s interval $[Mbps]$
throughputMedian	Median of avg. throughputs per 5 s interval $[Mbps]$
initialThroughput2	Throughput in first 2 seconds $[Mbps]$
initialThroughput3	Throughput in first 3 seconds $[Mbps]$
initialThroughput5	Throughput in first 5 seconds $[Mbps]$
initialThroughput10	Throughput in first 10 seconds $[Mbps]$
dupack	Number of duplicate acknowledgements
dupackOverAll	Ratio of duplicate acknowledgements
retransmission	Number of retransmissions
retransmissionOverAll	Retransmission ratio
ackLostSegment	Number of packets that acknowledge lost segment
ackLostSegmentOverAll	Ratio of packets that acknowledge lost segment
push	Number of packets with TCP flag push set
pushOverAll	Ratio of packets with TCP flag push set
reset	Number of packets with TCP flag reset set
resetOverAll	Ratio of packets with TCP flag reset set
numberOfFlows	Number of TCP flows established
numberOfServers	Number of contacted servers

Specification of QoE classes

Previous studies have shown that QoE in HTTP-based adaptive video streaming services depends on: percentage of playback time spent on each quality level, quality switches, number of stalling events, average stalling duration, overhead time, and initial buffering [90, 91, 94]. Even though initial buffering is mentioned to have an impact on QoE, in this work it has been taken into account only through overhead time, because none of the videos had significant initial buffering duration. The study reported in [92] showed that initial delay, even if 15 seconds in duration, has low influence on QoE. For simplification purposes we also do not explicitly take into account the impact of the number of quality switches, because most of the videos ($>$ 95%) comprising our dataset had a low total number of quality switches (< 3) and, according to [194] and [195], in cases of low adaptation frequency, time on individual quality level becomes important.

At the time this study was conducted there was no multidimensional QoE model for HAS, as ITU-T Recomm. P.1203 [97] was published later in 2017. For the purpose of relating multiple KPIs with QoE, in this study we used a simplified model, taken from our earlier work in [176], which uses as a basis existing models and findings with respect to the impact of various QoE influence factors [90, 91, 92, 93, 94]. Considering the difference in model dimensionality, the classes in this work cannot be expressed as a range of MOS values provided by related works. We note that the classification approach used in this study was proposed only for proof-of-concept purposes, and that our aim was not to propose new QoE models for adaptive video streaming.

Three QoE classes have been defined: "high", "medium" and "low". Given the lack of well-defined thresholds for the considered QoE influence factors, we use a fuzzy-logic–inspired approach to model the boundaries between classes that are not fixed for each influence factor. A video streaming session (referred to as an *instance*) is evaluated to check if it belongs to the "high" or "low" QoE class. If it belongs to neither, it is classified as the "medium" QoE class. Both functions measure the level of degradation. Values greater than 1 represent high degradation and classification to the low QoE class, in terms of the boundary that is being checked ("high"-"medium" or "medium"-"low").

The first function, which checks if the QoE class of an instance i is "high", is defined as:

$$checkHigh(i) = \begin{cases} nPHQ(i) + nSD(i), & SC \leq 1 \\ 1, & otherwise \end{cases}$$

where nPHQ is the normalised value of the percentage of playback time spent on high quality (*hd1080*, *hd720*, *large*), nSD is the normalised value of stalling duration, and SC is stalling count. Normalisation functions are defined as follows (PHQ is percentage of playback spent on

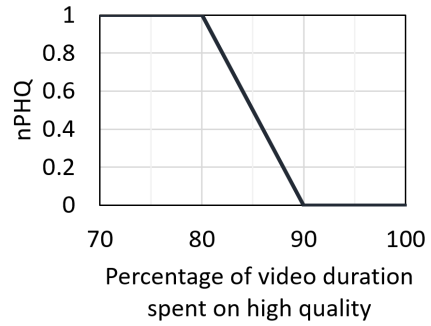
high quality and SD is stalling duration in seconds):

$$nPHQ(i) = \begin{cases} 0, & PHQ(i) > 0.9 \\ 9 - 10 \cdot PHQ(i), & PHQ(i) \in [0.8, 0.9] \\ 1, & PHQ < 0.8 \end{cases}$$

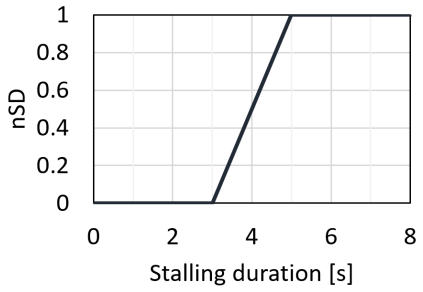
$$nSD(i) = \begin{cases} 0, & SD(i) < 3 \\ 0.5 \cdot SD(i) - 1.5, & SD(i) \in [3, 5] \\ 1, & SD(i) > 5 \end{cases}$$

If the output value of the function $checkHigh(i)$ is less then 1, degradation is considered not significant, and instance i is classified as "high" QoE. Otherwise, it is checked by the function $checkLow(i)$ and if it does not belong there either, the instance is classified as "medium".

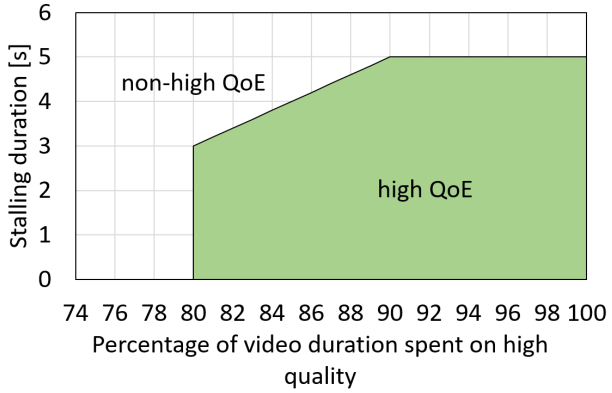
The defined normalisation functions are illustrated with Figures 5.9a and 5.9b. The checkHigh function is defined as a sum of aforementioned functions and is illustrated with Figure 5.9c. Functions described in further text are not graphically illustrated, as they follow the same logic.



(a) Normalisation of percentage of video duration spent on high quality (*hd1080*, *hd720* or *large*).



(b) Normalisation of stalling duration.



(c) checkHigh function.

Figure 5.9: Functions used to normalise the parameters of checkHigh function and their effect on the checkHigh function output.

The function used to check whether or not a video session should be classified as "low" is as follows:

$$checkLow(i) = nPLQ(i) + nSC(i) + nOR(i) + nASD(i)$$

PLQ refers to percentage of playback time spent on low quality (*small* and lower), OR refers to overhead ratio (overhead time and video duration ratio), SC is stalling count, and ASD is average stalling duration in seconds. These influence factors are normalised as follows:

$$nPLQ(i) = \begin{cases} 0, & PLQ(i) < 0.5 \\ 10 \cdot PLQ(i) - 5, & PLQ(i) \in [0.5, 0.6] \\ 1, & PLQ(i) > 0.6 \end{cases}$$

$$nSC(i) = \begin{cases} 0, & SC(i) < 4 \\ 0.5 \cdot SC(i) - 2, & SC(i) \in [4, 6] \\ 1, & SC(i) > 6 \end{cases}$$

$$nOR(i) = \begin{cases} 0, & OR(i) < 0.08 \\ 25 \cdot OR(i) - 2, & OR(i) \in [0.08, 0.12] \\ 1, & OR(i) > 0.12 \end{cases}$$

$$nASD(i) = \begin{cases} 0, & ASD(i) < 3 \\ 0.5 \cdot ASD(i) - 1.5, & ASD(i) \in [3, 5] \\ 1, & ASD(i) > 5 \end{cases}$$

If the function $checkLow(i)$ returns a value greater than or equal to 1, degradation is considered significant and the session is classified as "low" QoE. If any of the four normalisation functions returns 1, session i is labelled with "low" QoE. If the sum of output values of these four functions is less than 1, the session is classified as "medium".

For most of the values taken in the QoE models we have relied on previous work. For example, the function that normalises the stalling duration for the checkHigh function was devised from [92], where the authors describe the influence of stalling duration and number of stalling events on MOS. In [92] it can be seen that one stalling lasting 3 seconds in duration still results in high QoE (more than 3.5 on a 5 pt. ACR scale). The boundaries were relaxed to enable videos with one stalling up to 5 seconds in duration to be classified as "high" if video resolution was high at least 90% of the video duration. It is important to note that in [92] video duration is 30s, and that is another reason to allow videos with slightly longer stalling events to be classified as "high". Due to the lack of a well-defined threshold for overhead ratio, we

assume a value of 10% as corresponding to high. We thus establish the limit of 0.12 by using fuzzy logic, in which the limit gradually raises from 0.8 to 0.12 (note that those values are symmetric to the limit of 0.10).

With regards to the reasoning for choosing three QoE classes, we consider a network operator or service provider perspective. An important aim may be to distinguish between users that are highly satisfied with the service quality, those who are not completely satisfied, but are not giving up on the service, and those who are unsatisfied and will potentially quit the service or churn [196]. The notion of a so called *Net Promoter Score* [197] has also been used by service providers, grouping customers into three groups, namely those promoting the service, passive and unenthusiastic customers, and those likely to churn and spread negative word-of-mouth. QoE measures relevant to operators and discussed by Hoßfeld et al. in [198] that may be related to this notion are percentage of users rating a service as Good or Better (%GoB, e.g., 4 or 5 on a 5-pt. ACR scale, also may be considered as "high QoE"), or percentage of users rating a service as poor or worse (%PoW, 1 or 2 on a 5-pt. scale, corresponding to "low QoE"), leaving a rating of 3 to be considered as neutral, or "medium QoE". In the context of QoE management, such a classification of high/medium/low QoE may be linked to end user behaviour and acceptance (e.g., users experiencing low QoE are likely to churn), thus providing important input to providers looking to dimension and operate their services [198]. Although we initially defined three QoE classes, we also tried classifying streamed YouTube videos into two QoE classes (discussed in Section 5.1.5), as was done in the Prometheus approach [10]. It is important to note that once the network- and application-level data is collected, we can build different ML models, with different numbers of QoE classes and differently defined models that calculate a QoE class of an instance, to suit the needs of a specific system that would use a built model.

5.1.4 Data analysis

Analysis of the collected dataset

The collected dataset contains information corresponding to the streaming of 1060 videos. The percentage of instances labelled with each of the defined QoE classes per experiment is shown in Figure 5.10. A total of 333 instances were labelled as "low", 310 as "medium", and 417 as "high".

Figure 5.11 shows overall stalling statistics: number of videos with certain stalling count, average stalling duration, and number of stalling events per experiment. It can be observed that in most of the videos, no stalling event occurred. In 95% of the videos with a stalling occurrence, average stalling duration was shorter than 2 seconds. It is also interesting to note that most of the stalling events happened in experiments with low overall bandwidth, with drastic bandwidth changes, and in experiments with an increase in available bandwidth. The last observation can

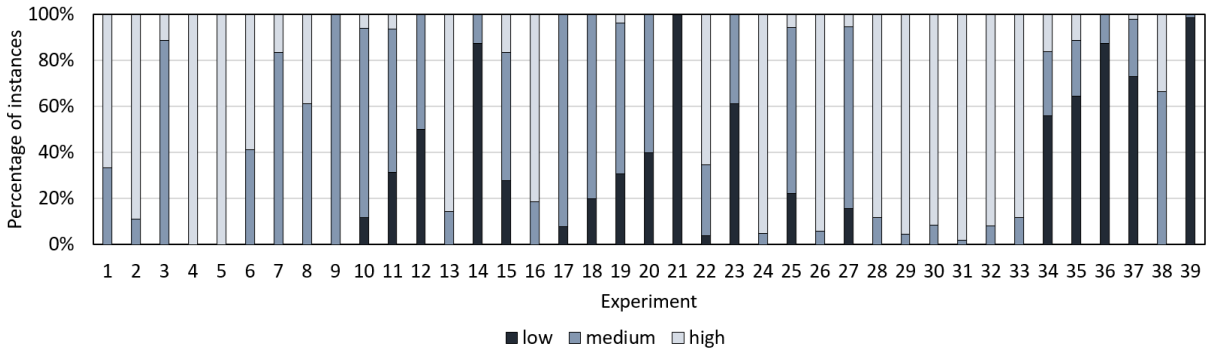
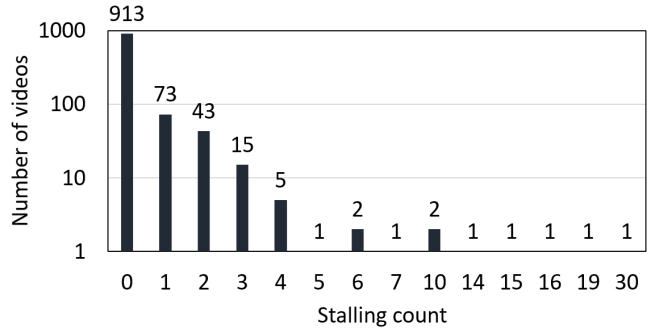
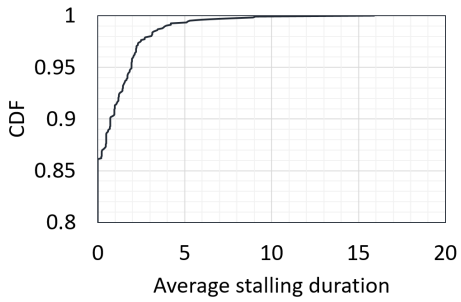


Figure 5.10: Percentage of instances in each class, per experiment.

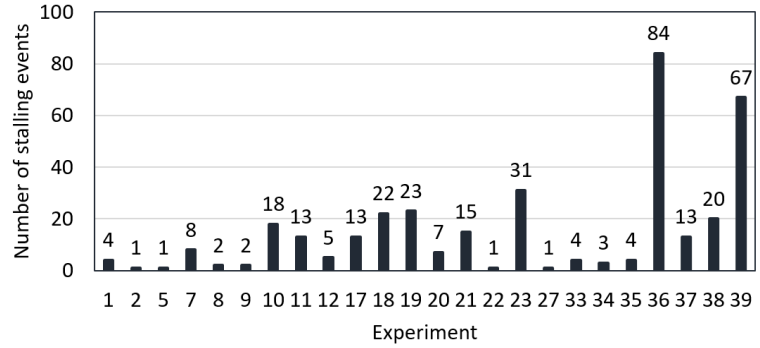
be attributed to the fact that YouTube, when switching to a higher quality level, presumably discards all of the video content buffered in a lower quality level and starts downloading and playing higher quality video immediately.



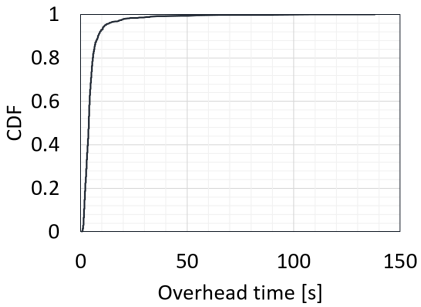
(a) Number of videos with a given stalling count.



(b) Average stalling duration.



(c) Number of stalling events per experiment. Note: experiments with 0 stalling events are not included.

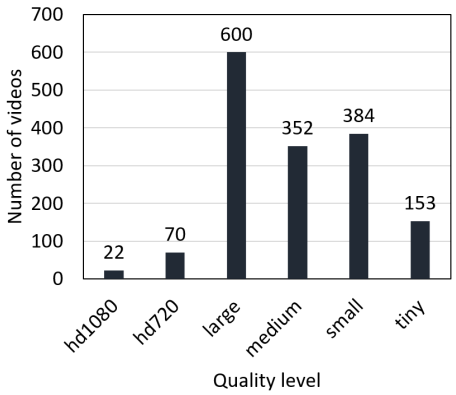


(d) Overhead time.

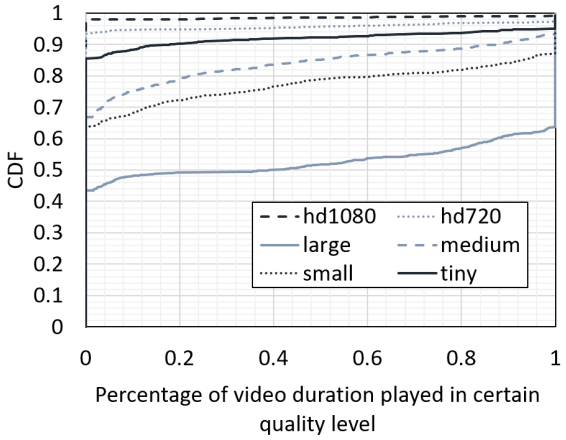
Figure 5.11: Stalling and overhead time statistics for the collected dataset.

In Figure 5.11d it can be observed that overhead time, i.e., the sum of initial buffering and stalling durations, was shorter than 10 s in more than 90% of the cases. Figure 5.12a shows the number of videos that at some point played in a certain quality level, while Figure 5.12b shows the distribution of percentage of video duration played in a certain quality level. It can be observed that a large number of videos were not played in *hd1080* or *hd720* because these quality levels were either not available, or, even if they were available, were not retrieved by the

YouTube client algorithm. This may be due to the fact that the videos were played on a mobile device and YouTube assumes that quality level *large* is good enough to achieve satisfactory QoE on a small screen. All the videos from all the experiments were available at least in quality level *large*, if not higher.



(a) Number of videos that played in a certain quality level at some point.



(b) Percentage of video duration played in a certain quality level.

Figure 5.12: Quality level statistics.

Feature selection

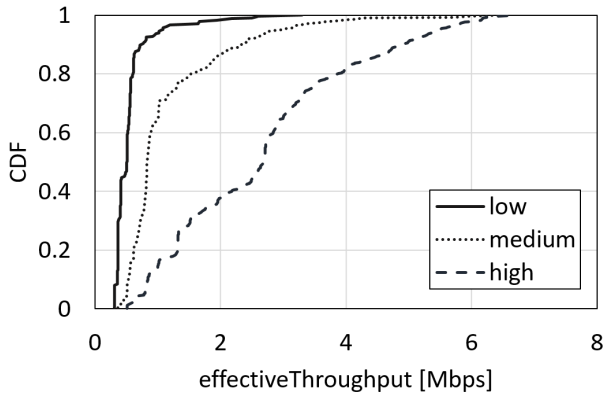
To reduce computational cost and prevent overfitting the models to training data, features were subsetted by using Wrapper methods in WEKA. Considering the fact that each classification algorithm works best with a different set of selected features, Wrapper methods were run for each tested algorithm: Random Forest, Naïve Bayes, J48, and SMO. Table 5.4 shows the final feature subsets for each algorithm, and classification accuracy of the model that uses selected features.

The complete list of features is given in Table 5.3, while most commonly selected features in the tested algorithms, ranked by the number of algorithms they are used by are listed as follows:

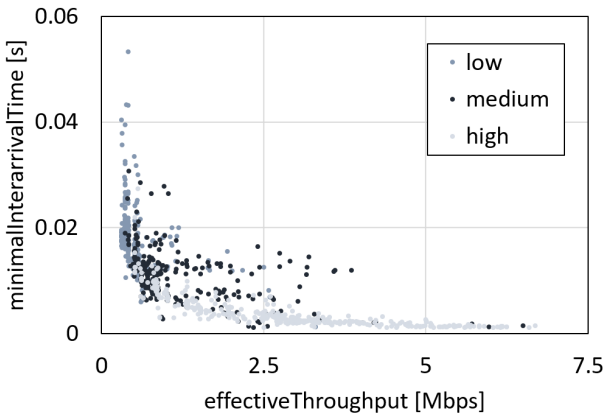
- *interarrivalTimeThroughTimeMedian*,
- *effectiveThroughput*,
- *minimalInterarrivalTime*,
- *avgPacketSize*,
- *push*, and
- *dupack*.

Obvious differences have been observed between the cumulative distributions of each at-

tribute between different classes. For example, in Figure 5.13a, it is visible that videos from classes “low” and “medium” in most cases exhibit effective throughput lower than 1 Mbps. In the case of videos from class “high”, more than 80% of the videos have higher effective throughput. If instances are plotted in 2D graphs, differences are even more evident. The graph in Figure 5.13b displays all the instances in two-dimensional space, where the dimensions are 2 attributes: *effectiveThroughput* and *minimalInterarrivalTime*. *minimalInterarrivalTime* is in most cases lower than 0.005 s for instances of class “high” and higher than 0.01 s for instances of class “low”. It should be noted that the features were calculated for the network traffic captured during the watch time of each video.



(a) Cumulative distribution of effective throughput.



(b) 2D visualisation of effective throughput and minimal packet interarrival time.

Figure 5.13: Differences in network traffic features’ values between classes.

5.1.5 Trained models

Classification results

In this section we present the models built by using five different machine learning algorithms implemented in WEKA. We first provide a short description of each ML algorithm:

- OneR: a simple learning algorithm that creates rules that classify an object on the basis of a single attribute [199],
- Naïve Bayes: a classifier that uses all the attributes from the input with the assumption that the attributes are equally relevant and statistically independent [200],
- SMO: a discriminative classifier based on finding a hyperplane that maximises the minimum distance to the training examples [201],
- J48: a decision tree algorithm that uses a top-down strategy to split the dataset on each node (i.e., attribute) depending on attribute value [202],
- Random Forest: a learning algorithm that grows many decision trees and classifies instances in the class most decision trees agree on [203].

For each of the five tested algorithms, features were selected and models built by using selected features only. 10-fold cross-validation was used for testing all the models. The results are summarised in Table 5.4. The most accurate classification (up to 80.18%) was achieved with the model built by the Random Forest algorithm. The simplest model uses only one feature (*throughputMedian*) and was built by the OneR algorithm. Naïve Bayes, J48, and SMO create models with around 77% classification accuracy.

Table 5.4: Classification results

Algorithm	**Selected features**	**Accuracy**
OneR	*throughputMedian*	74.62%
Naïve Bayes	*avgPacketSize*, *averageInterarrivalTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *push*, *interarrivalTimeThroughTimeMedian*, *initialThroughput2*	77.35%
SMO	*maximalSizeThroughTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *maxThroughputThroughTime*, *dupack*, *effectiveThroguhput*	77.35%
J48	*minimalInterarrivalTime*, *avgInterarrivalTimeThroughTime*, *sizeThroughTimeStdDev*, *interarrivalTimeThroughTimeMedian*, *throughputMedian*	78.20%
Random Forest	*avgPacketSize*, *minimalSizeThroughTime*, *push*, *initialThroughput10*, *minThroughputThroughTime*, *interarrivalTimeThroughTimeMedian*, *dupack*, *effectiveThroughput*	80.18%

To evaluate the performance of different ML algorithms in terms of computation time, we split the dataset in 2 parts, each containing 530 instances, and measured the time taken to build a model and the time taken to test the model on the testing split of the dataset. It took 0.02 s using WEKA on a PC (Intel Core i7-6500U@2.50GHz, 8GB RAM) to classify 530 instances using the model built by Random Forest, while simpler models, such as the one built by OneR,

required only 0.01 s to classify 530 instances. Such results indicate the potential utilisation of such models for real-time QoE monitoring systems. Model building in the described scenario requires slightly more time, from 0.01 for OneR up to 0.05 s for SMO algorithm, but this is not relevant for real-time utilisation as models are built offline. We also note that the accuracies of these models were in line with those we observed when using cross-validation.

To aid in interpretation of the built models, we visualise the J48 decision tree model for classifying YouTube QoE based on network traffic features. Figure 5.14 shows the decision tree with attribute values the dataset is split on, together with results in terms of total number of instances and number of misclassified instances on each leaf. The feature *minimalInterarrival-Time* proves to be the most relevant feature in terms of information gain and thus is placed at the root of the tree. Instances with lower value of *minimalInterarrivalTime* are mostly classified into QoE class "high" (356 out of 382 instances on the left part of the tree). On the right side of the tree, with higher values of *minimalInterarrivalTime*, instances are classified as "medium" or "low", depending on the values of other traffic features, with the exception of 12 instances that the model classifies as "high" (which is wrong in 5 cases). If we analyse other features, we can draw the following conclusions:

- Shorter interarrival time is a sign of higher QoE.
- Higher throughput is a sign of higher QoE.
- Higher throughput deviations are a sign of higher QoE.

Out of 1060 instances that comprised our dataset, the model built by Random Forest classifies 210 of them incorrectly. Figure 5.15 shows the number of incorrectly and correctly classified instances per experiment.

Experiments without sudden bandwidth changes have lower error rates. In reality, even though variable bandwidth availability is a likely occurrence in mobile networks, it is common that available bandwidth does not change so drastically over short periods of time. This means that the classifier's accuracy would likely increase when considering traces collected in a more realistic network environment, such as an operational mobile network.

A confusion matrix (Table 5.5) shows that most of the confusions happen between classes "low"-"medium" and "medium"-"high". Only 10 examples were classified as "low" and actually belong to "high" and only 6 examples that belong to "low" were classified as "high". When we observe instances with respect to only 2 traffic features, as shown in Figure 5.13b, we can observe that the instances overlap, and if we try to draw prediction borders between classes, we would have errors. Errors are reduced when more network traffic features are used. The reason why classification for "medium" QoE class is much worse than for classes "high" and "low" lies simply in having 2 borders.

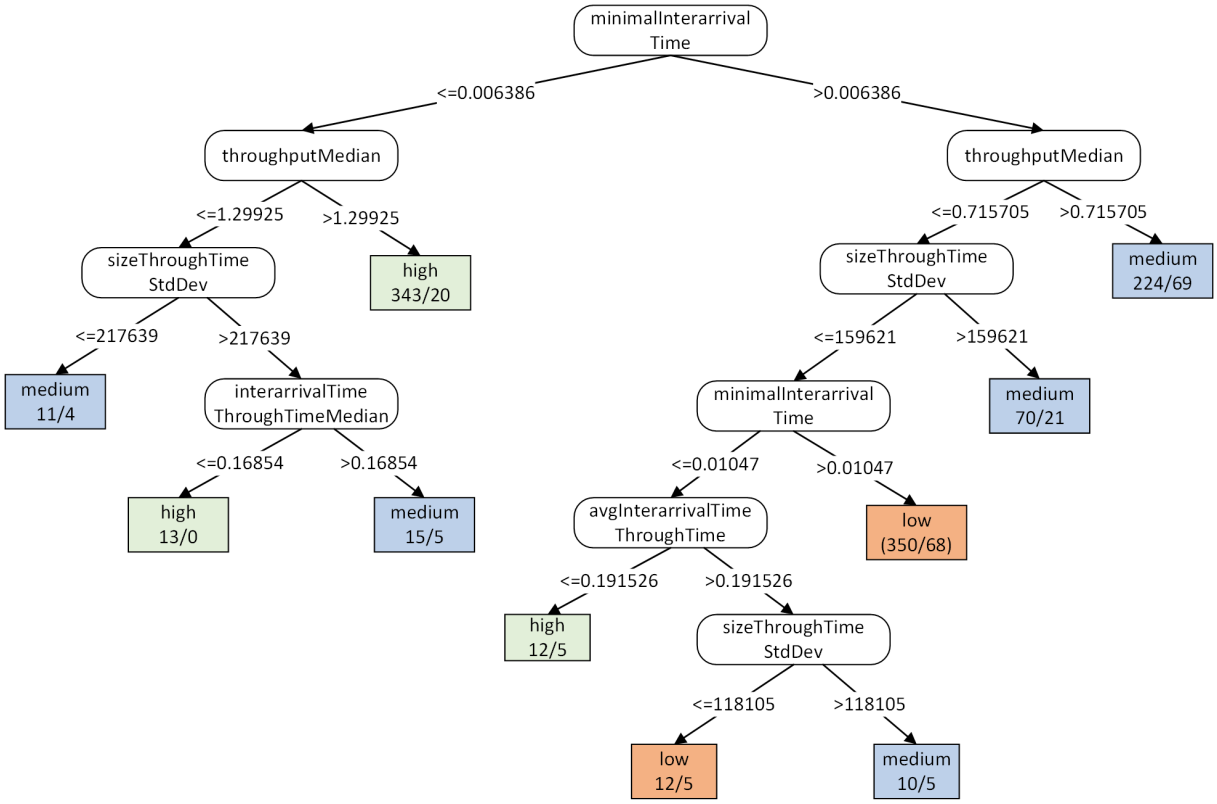


Figure 5.14: J48 tree model visualisation. Numbers on the leaves indicate the total number of classifications, followed by the number of incorrect classifications through the corresponding rule.

Table 5.5: Confusion matrix of classification using Random Forest

Classified as			Actual class
low	medium	high	↓
281	46	6	low
66	207	37	medium
10	45	362	high

Random 25 of the incorrectly classified videos were manually inspected to determine causes that might be used in the future to improve the methodology. The inspection resulted in the following conclusions for two types of classification errors:

- Instances classified in a quality class lower than the one they actually belong to were mostly static videos (presentations, song lyrics, album covers,...).
- Instances classified in a quality class higher than the one they actually belong to mostly had high resolution but because of a few stalling events, the actual class was lower.

To investigate further why prediction errors happen, a Web application for rendering confusion matrices based on subsets of instances has been developed. The application uses RStudio's Shiny framework [204] to subset the dataset based on chosen factors such as average band-

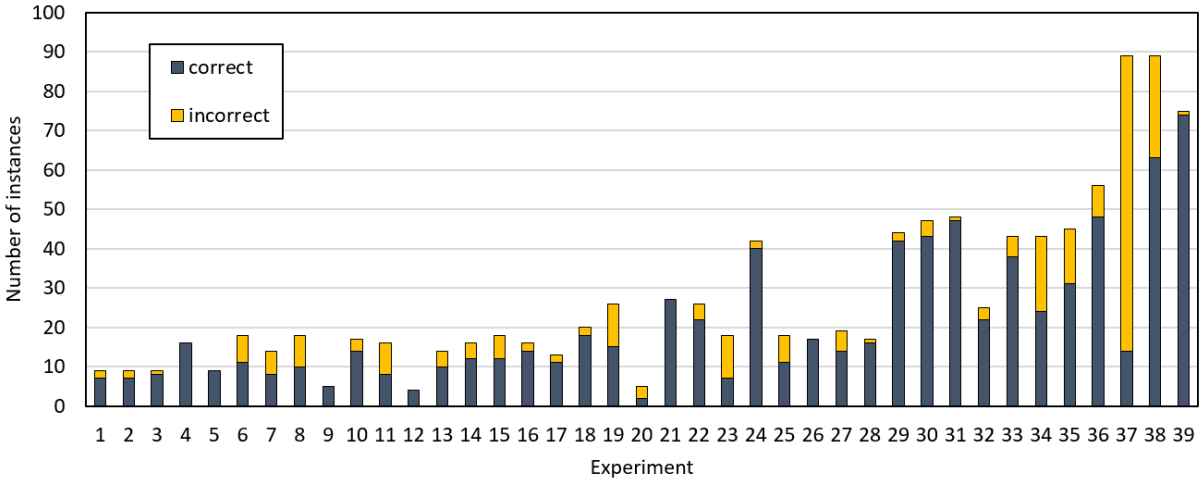


Figure 5.15: Number of correct and incorrect predictions per experiment by using the model built by Random Forest.

width in the experiment, maximum change in bandwidth (as defined by bandwidth envelopes), number of bandwidth changes, number of stalling events, average stalling duration, and quality level in which video was played the longest. The application also visualises 2D scatter plots of instances in relation to two selected features. Statistics of each feature can also be inspected separately (Figure 5.16). The application is interactive and responsive.

A confusion matrix for the Random Forest algorithm is shown in Figure 5.17. In the Figure, one filter is applied using the sliders on the left, to consider a subset of input instances corresponding to experiments with no bandwidth changes. It can be observed that classification accuracy is very high for this subset. If changes up to 1 Mbps are allowed, accuracy is somewhat lower (Figure 5.18).

By using this Web application, we also observed that classification errors increase when instances with higher stalling count and longer stalling durations are subsetted. This means that the model needs improvements to be able to recognise stalling events as degrading to QoE. However, a key conclusion from running our tests was that the primary factor contributing to QoE was quality level, while stalling events were rarely observed. This is due to the high performance of YouTube's adaptation algorithm in terms of ability to avoid stalling events.

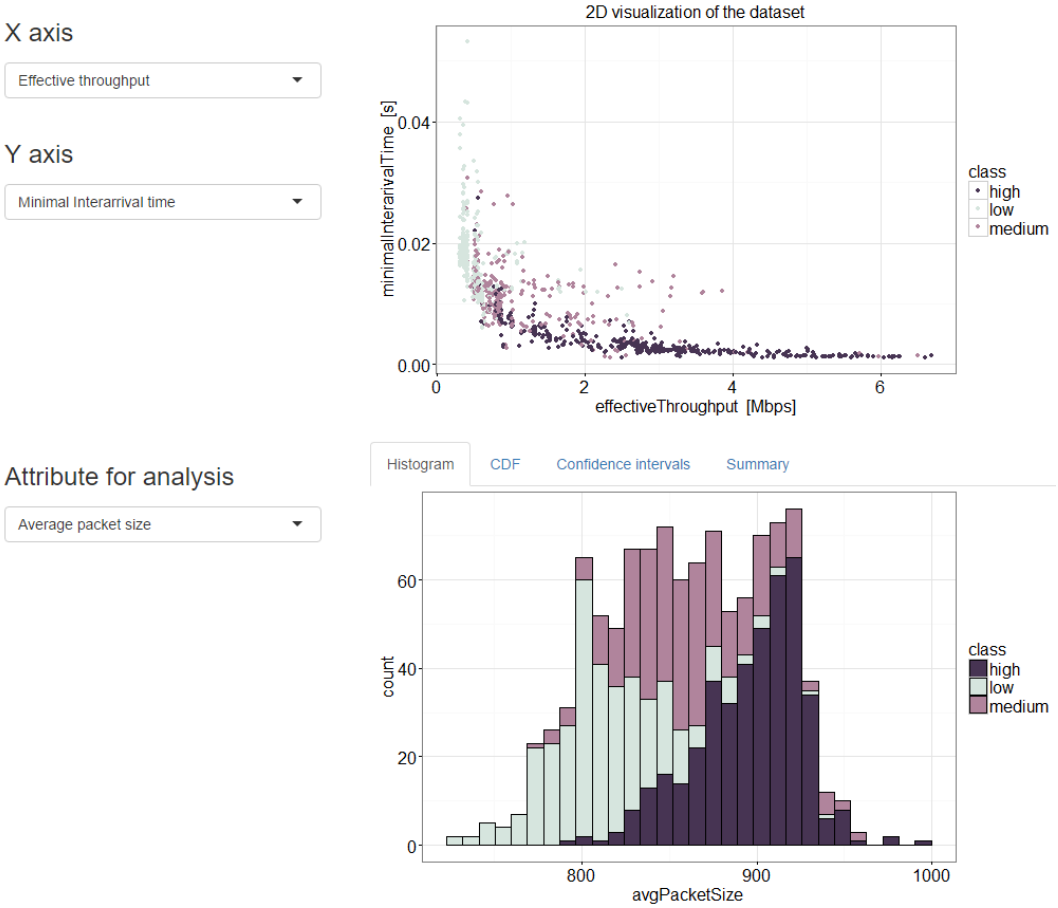


Figure 5.16: Implemented Web application for dataset visualisation.

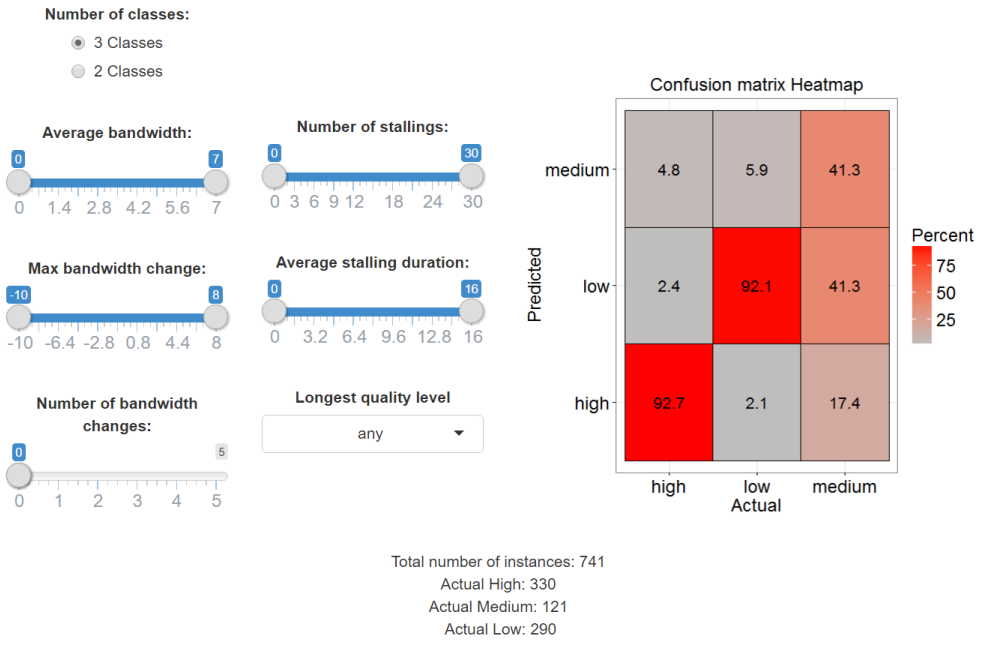


Figure 5.17: Random Forest confusion matrix for instances corresponding to experiments with no bandwidth changes during the video sessions.

Figure 5.18: Random Forest confusion matrix for instances corresponding to experiments with bandwidth changes up to 1 Mbps.

Models built using a reduced dataset involving only experiments with constant bandwidth limitations

The analysis earlier in this section showed that prediction errors happen mostly in experiments with bandwidth changes. As sudden and drastic bandwidth changes are not so common in real network conditions, we decided to measure classification accuracy using a reduced dataset, i.e., we considered only experiments with constant bandwidth limitations (i.e., experiments 16, 14, 21, 24-37 and 39). The models were tested on the reduced dataset using 10-fold cross-validation, and the classification accuracies are given in Table 5.6. The accuracy improved up to 84% for most algorithms, even OneR, that uses only one feature, *throughputMedian*.

Binary classification models

Finally, we also tested models that classify video streaming instances into 2 QoE classes, so as to compare accuracy with the case of 3 QoE classes. Instances are checked with the function $checkLow(i)$. If the function returns a value greater or equal to 1, an instance is classified as "low" QoE and if not, as "high" QoE. The function is defined as:

$$checkLow(i) = nPMQ(i) + nPLQ(i) + nSC(i) + nOR(i) + nASD(i)$$

The function uses the percentage of playback spent on *medium* quality level, percentage of playback spent on low quality (*small* or *tiny*), stalling count, overhead ratio, and average stalling duration in seconds, all normalised to values from $[0,1]$ interval. Normalisation functions are

Table 5.6: Classification results using a reduced dataset.

Algorithm	**Selected features**	**Accuracy**
OneR	*throughputMedian*	83.8%
Naïve Bayes	*avgPacketSize*, *averageInterarrivalTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *push*, *interarrivalTimeThroughTimeMedian*, *initialThroughput2*	83.94%
SMO	*maximalSizeThroughTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *maxThroughputThroughTime*, *dupack*, *effectiveThroughput*	80.97%
J48	*minimalInterarrivalTime*, *avgInterarrivalTimeThroughTime*, *sizeThroughTimeStdDev*, *interarrivalTimeThroughTimeMedian*, *throughputMedian*	83.26%
Random Forest	*avgPacketSize*, *minimalSizeThroughTime*, *push*, *initialThroughput10*, *minThroughputThroughTime*, *interarrivalTimeThroughTimeMedian*, *dupack*, *effectiveThroughput*	83.8%

the following:

$$nPMQ(i) = \begin{cases} 0, & PMQ(i) < 0.5 \\ 10 \cdot PMQ(i) - 5, & PMQ(i) \in [0.5, 0.6] \\ 1, & PMQ > 0.6 \end{cases}$$

$$nPLQ(i) = \begin{cases} 0, & PLQ(i) < 0.2 \\ 20 \cdot PLQ(i) - 4, & PLQ(i) \in [0.2, 0.25] \\ 1, & PLQ(i) > 0.25 \end{cases}$$

$$nSC(i) = \begin{cases} 0, & SC(i) < 5 \\ 0.5 \cdot SC(i) - 2.5, & SC(i) \in [5, 7] \\ 1, & SC(i) > 7 \end{cases}$$

$$nOR(i) = \begin{cases} 0, & OR(i) < 0.1 \\ 20 \cdot OR(i) - 2, & OR(i) \in [0.1, 0.15] \\ 1, & OR(i) > 0.15 \end{cases}$$

$$nASD(i) = \begin{cases} 0, & ASD(i) < 3 \\ 0.5 \cdot ASD(i) - 1.5, & ASD(i) \in [3, 5] \\ 1, & ASD(i) > 5 \end{cases}$$

We again note that the values used in the functions are used for proof-of-concept. Table 5.7 provides an overview of features selected for binary classification and models accuracy. The highest accuracy was as expected higher than in the case of 3 QoE classes and again achieved by using the Random Forest algorithm (90.66%), with the other models being only slightly less accurate.

Table 5.7: Binary classification results

Algorithm	**Selected features**	**Accuracy**
OneR	*effectiveThroughput*	87.17%
Naïve Bayes	*avgPacketSize*,*throughputStdDev*, *initialThroughput10*, *avgInterarrivalTimeThroughTime*, *interarrivalTimeThroughTimeMedian*, *effectiveThroughput*	86.79%
SMO	*ackLostSegmentOverAll*, *maximalSizeThroughTime*, *minimalInterarrivalTime*, *initialThroughput10*, *retransmissionOverAll*, *numberOfServers*, *maxThroughputThroughTime*, *dupack*, *effectiveThroughput throughputMedian*	87.07%
J48	*ackLostSegmentOverAll*, *maximalSizeThroughTime*, *dupackOverAll*, *interarrivalTimeThroughTimeMedian*	88.02%
Random Forest	*averageInterarrivalTime*, *ackLostSegmentOverAll*, *pushOverAll*, *initialThroughput10*, *minThroughputThroughTime*, *maxThroughputThroughTime*, *interarrivalTimeThroughTimeMedian*, *retransmission*, *effectiveThroughput*	90.66%

5.2 Session-level QoE/KPI estimation for Android

Following the promising results achieved in Study S1, and given the changes in how the YouTube service is delivered, we conducted Study S2 in 2017 – 2018 and published the results in [14, 15], improving the methodology in multiple aspects. The study compares different approaches to ground-truth data collection, and presents our newly developed approach to collecting application-level data using a tool called ViQMon, which was used from this study onward. Furthermore, addressing the fact that our studies should that YouTube started predominantly delivering its content via the QUIC protocol, we define a new set of traffic features that can be calculated solely from IP-level data. Finally, with the QoE model for HAS published in ITU-T Recommendation P.1203 [97], we use this standardised model for deriving overall QoE of video sessions. Derived QoE scores are then used to classifying video sessions into QoE classes, rather than using the custom model devised in Study S1. Additionally, this study considers the classification of various application-level KPIs, as opposed to just classifying QoE. The models in this study are trained and validated on data collected in a WiFi network with

more realistic bandwidth conditions based on actual network measurements, and also validated on data collected in an operational mobile network.

5.2.1 Ground-truth data collection methods

Over the past years, different client-side apps that monitor YouTube's performance have been developed, both commercial [110, 111], and non-commercial [13, 112, 113]. The purpose of such apps is multifold: besides obtaining ground-truth information for QoE estimation or prediction, they can be used for benchmarking network performance, analysing service behaviour, etc.

For YouTube, client-side monitoring approaches found in literature can be divided into three groups. The first group are apps that embed YouTube videos using the IFrame API (e.g., YoMoApp [112], YouQ_IF [13]). We note that throughout this study, we refer to two versions of the YouQ app as YouQ_IF (the version built on YouTube IFrame API [116]), and YouQ_AA (the version built on YouTube Android API [117]). Although the IFrame API offers a lot of information about playback quality parameters, the question is to what extent do such client-side apps behave in the same way (referring to video streaming and adaptation logic) as the actual YouTube app that are most commonly used by YouTube viewers on smartphones. Another group of YouTube performance monitoring apps are apps that embed YouTube videos using the native YouTube API for Android (e.g., YouQ_AA [15]). However, a native API is only available for Android, and the number of performance metrics that can be retrieved through the API is limited, as compared to the IFrame API. The third approach is to perform monitoring using the YouTube application. An example of such an approach can be found in [125], where the authors captured the screen while playing YouTube videos and used image processing techniques to detect the spinner that indicates stalling or initial buffering.

In the scope of this study, a new methodology for measuring application-level KPIs has been developed. The so called Video Quality Monitor (ViQMon) approach relies on recording the screen while using the actual YouTube app, with the *Stats for Nerds* option enabled to display meta-data regarding video playback. The video recording is processed using Optical Character Recognition (OCR) techniques to derive QoE-relevant KPIs. The approach is platform-independent and enables monitoring on different operating systems. The *Stats for Nerds* option displays video meta-data as an overlay over the video. Information displayed includes video dimension, video resolution, unique ID, framerate, buffer state, etc. Recordings are then analysed using Tesseract OCR engine [205] to read data necessary to derive application-layer KPIs per video viewing session, such as initial delay, average stalling count and duration, and percentage of time played on a given video quality level (Figure 5.19).

The main advantage of the approach is that it enables monitoring of application-level performance while using the YouTube app, as opposed to approaches that rely on embedding YouTube

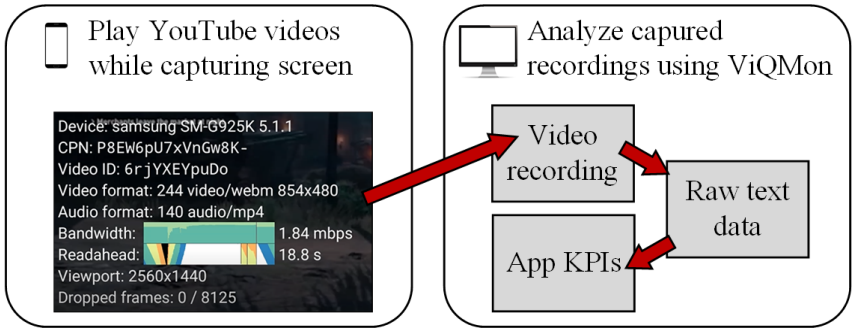


Figure 5.19: ViQMon methodology.

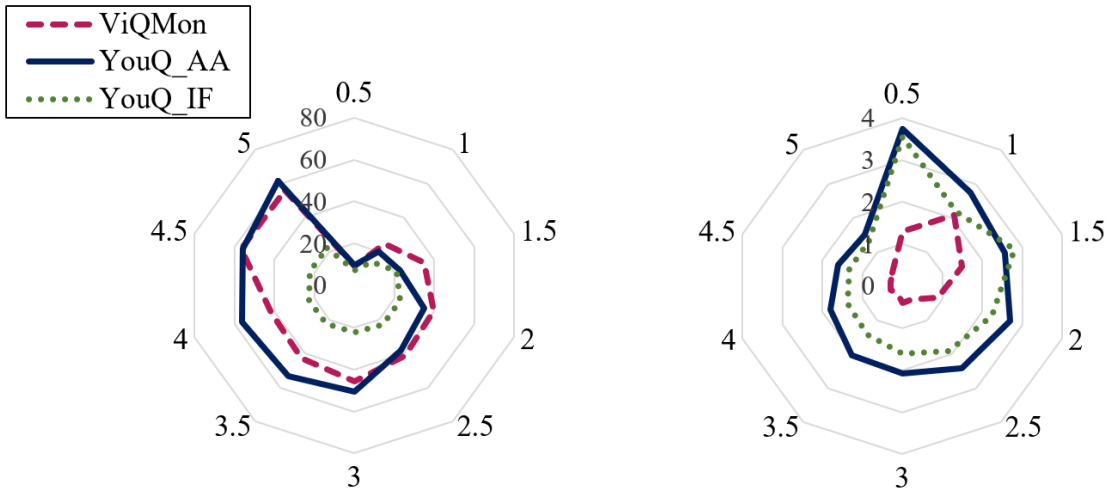
videos. In this way, we are able to collect data in realistic and reliable usage scenarios. Another advantage is that data analysis is done offline, on a desktop computer running ViQMon, while on mobile devices we only capture the screen recording, using any screen recording app. For this reason, ViQMon is applicable for various devices, OS versions, etc. On the other hand, the main challenges concerning the approach are related to collected data size. To reduce the usage of storage space on a mobile device, in our experiments we record the screen with a framerate set to the lowest value available in the screen recording app, bearing in mind that *Stats for Nerds* data is refreshed once per second. Also, lowering screen capturing resolution could reduce ViQMon processing time, but could also result in lowered OCR precision.

To gain insight into the differences in YouTube's behaviour when different client-side QoE monitoring applications are used, we conducted measurements in a lab environment using YouQ_IF, YouQ_AA, and ViQMon. In all three cases, the same client device was used (Samsung S6), the same playlists were played in landscape mode, and the same network conditions were emulated. Using the laboratory setup depicted with Figure 4.3, with the client device connected to the Internet using a WiFi connection, and using IMUNES to emulate bandwidth limitations ranging from 0.5 Mbps to 5 Mbps, with a 0.5 Mbps step. At each bandwidth level, 10 videos were played differing in duration (63 to 435 seconds), popularity (number of likes), and type (music, sports, comedy). At the same time, we captured network traffic passing through a designated router. The final dataset consists of both application- and network-layer data corresponding to 300 played videos (10 videos streamed over 10 different bandwidth conditions, and repeated for 3 different measurement apps).

The aforementioned dataset was used to compare YouTube's behaviour in terms of traffic characteristics and initial buffering delay across three different client-side monitoring approaches. As it was not collected with the purpose of training QoE/KPI estimation models and only used in this study for this particular comparison, this dataset is not listed in Table 4.2. We note the following limitations of the comparison: (1) we could only use videos marked as embeddable, to be able to play them in YouQ apps, (2) YouQ_AA app does not log information about played video resolution, as it is not provided by the YouTube Android API, (3) when using

YouQ_IF, we observe TCP as the underlying transport protocol, whereas in experiments with YouQ_AA and ViQMon QUIC is used, and (4) all ViQMon data is collected with a precision of 1 s, as this is the rate at which *Stats for Nerds* data is refreshed.

Results are summarised in Figures 5.20a and 5.20b. Figure 5.20a shows the average per-video amount of data downloaded by the client application for each experiment (one experiment corresponds to 10 videos played at a given available bandwidth level, with 10 experiments conducted per client app). It can be observed that the amount of data is approximately the same in all of the experiments in which we used YouQ_IF (except on 0.5Mbps). From our logs, it is visible that the highest observed resolution played by the IFrame-based app was 480p, although higher video resolutions were available. On the other hand, the amount of data downloaded in the cases of YouQ_AA and ViQMon was significantly higher, with ViQMon log data showing higher resolutions being played as compared to YouQ_IF. However, as quality level data could not be collected using YouQ_AA, we can only assume that quality levels roughly correspond to those observed by ViQMon.



(a) Average per-video transferred data size [MB] in each experiment (denoted with available bandwidth).

(b) Average initial delay [s] in each experiment (denoted with available bandwidth).

Figure 5.20: Differences in YouTube's streaming behaviour between three QoE monitoring apps.

With Figure 5.20b we depict the differences in initial delay achieved with the three apps. It is clear that apps embedding YouTube videos introduce additional delay, as compared to the ViQMon case, where the YouTube app is used. We note that with ViQMon, data is collected with 1s precision, but even if calculated average initial delays shown in Figure 5.20b are increased by one second, the delay is still shorter than in the embedded video case. As bandwidth was kept constant and stable in a single experiment, we observed only 11 stalling events in total, during the playing of 300 videos. All of the stalling events happened in experiments with ViQMon, and when bandwidth was lower than 2.5Mbps. Another difference visible from the data is that, as opposed to YouQ_IF using the same buffering strategy as described in Section

5.1.2, with ViQMon we observe YouTube consistently using roughly 120 and 125 seconds as low and high buffer thresholds respectively, regardless of quality levels being played.

We conclude that obtained results indicate significant differences in streaming behaviour, and consequently traffic characteristics, across different monitoring approaches. Researchers and practitioners interested in monitoring app-level data should be aware of these differences, in particular when using this "ground-truth" data for benchmarking network performance, or building QoE estimation models relying on encrypted traffic features. Given the widespread use of the YouTube app on smartphones (Android and iOS), we further base our measurements and results on this case.

5.2.2 Data collection and processing

Lab setup and measurement campaign

The And18L dataset was collected to build ML–based models that estimate application-level performance from network-level metrics, given the new findings with respect to changes YouTube introduced. For obtaining ground-truth data, we used the ViQMon approach. Important to consider is the need to observe different quality degradations that might occur at the client, so as to enable ML algorithms to learn from examples, and finally to enable built models to recognise such events when the model is deployed in a network. To emulate conditions that resemble those found in a real network, we scripted the IMUNES emulator to use the 4G/LTE bandwidth logs [172] published in [173]. The logs cover 5 hours of active monitoring, and were collected in different mobility scenarios: on foot, bicycle, tram, train, and car. An excerpt of the logs is depicted in Figure 5.21.

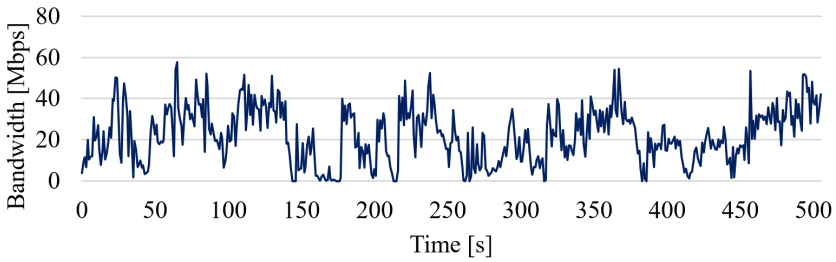


Figure 5.21: 4G/LTE bandwidth log data collected using an Android application on the train (based on data from [173]).

We created playlists on YouTube containing a total of 100 videos, with video duration distribution depicted in Figure 5.22a, and played them on Samsung S6 while connected to the network via a WiFi access point and with the aforementioned bandwidth envelopes imposed (using the lab setup from Figure 4.3). As bandwidth observed in the logs is generally high and short outages would not induce quality degradations (due to content being buffered on the client), we repeated the playing of 100 videos three more times, while dividing the original bandwidth

by factors of 10, 20, and 30. In this way we obtained data from video streaming sessions with quality degradations, while keeping virtually realistic bandwidth fluctuations. As 2 of the 100 videos were deleted from YouTube sometime during the experiments, Dataset And18L contains data corresponding to 394 played videos.

Network traffic features applicable for QUIC traffic

With the introduction of QUIC, information about retransmissions, lost segments, duplicate acknowledgements, etc., became encrypted. At the time that this study was conducted, a study by Mazhar *et al.* [141] also addressed in-network KPI monitoring applicable in the case of adaptive video delivered via QUIC, but focusing on real-time prediction. The feature set used in their study also includes features based on throughput and packet count, but calculated in smaller intervals (given that the focus is on per-10-seconds QoE estimation). Compared to Study S1, where some network traffic features are derived from such transport-layer data, in Study S2, we only use network-layer data (i.e., data from IP packet headers) for the feature calculation. We list the 24 metrics calculated from network traffic, based on which the prediction is performed:

- *averageThroughput*,
- *avgPacketSize*,
- *[mean, hMean, median, lowMedian, highMedian, groupedMedian, min, max, stDev, pStDev, pVar] SizeIn5sIntervals*,
- *[mean, hMean, median, lowMedian, highMedian, groupedMedian, min, max, stDev, pStDev, pVar] SizeIn1sIntervals*.

Most of the features are based on calculating statistics on lists that store the amount of data downloaded by the client in 1- or 5-second periods. For example, *avgSizeIn5sIntervals* refers to the average of the values in bins, where the value in each bin indicates the amount of downlink traffic in 5 seconds of the playback duration. We note that what is aggregated in each bin is the size of the whole frame. Although most of the features' names are straightforward, we further clarify the following: in our feature set "hMean" stands for harmonic mean, "pStDev" for population standard deviation, and "pVar" for population variance.

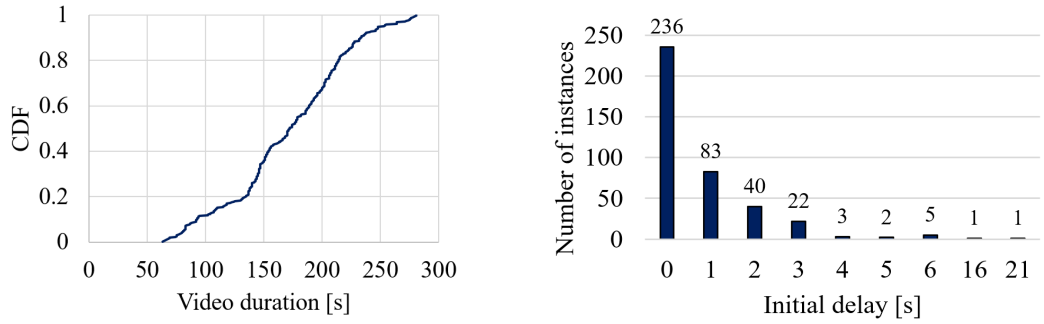
KPI and QoE classes (based on ITU-T Recommendation P.1203)

Following the approach illustrated with Figure 4.2, to train ML–based models that estimate YouTube performance, we calculated a series of metrics (predictors) from network traffic related to the streaming of each video (as described earlier in this section), and labelled these metrics with ground-truth application-level data (targets). The dataset is then provided as input to ML

algorithms to build models that are able to estimate the target variable using only the predictors. Aiming to assess models built to estimate different application-level parameters trained using different ML algorithms, we define the target application-level parameters as follows:

- **MOS2** - 2 MOS classes, based on the output of the model from ITU-T Recomm. P.1203 and its implementation published online [99, 103, 104] ("high", "low"), where instances with MOS $\geq$ 3.5 are labelled as "high", and "low" otherwise,
- **MOS3** - 3 MOS classes, based on the output of P.1203 ("high", "medium", "low"), where instances with MOS $\geq$ 4.0 are labelled as "high", MOS $\geq$ 3.0 and $<$ 4.0 as "medium", and MOS $<$ 3.0 as "low",
- **resolution** - 2 video resolution classes ("hd", "sd"), where "hd" means that the video was played in 720p or a higher resolution for most of its duration, "sd" otherwise, and
- **stalling** - 2 stalling occurrence classes ("yes", "no"), which denote if there were any stalling events (excluding initial delay).

We note that the model from ITU-T Recomm. P.1203 has been validated for H.264 video, while for VP9 (which we often observed in YouTube) we used the Non-Standard Codec Extension for ITU-T P.1203 [105]. The extension uses the outputs and the inputs of the standardised P.1203 model and corrects the outputs using a correction/mapping function defined in [106], making the model suitable for H.265, VP9, and AV1 codecs as well.



(a) Video duration distribution.

(b) Initial delay distribution.

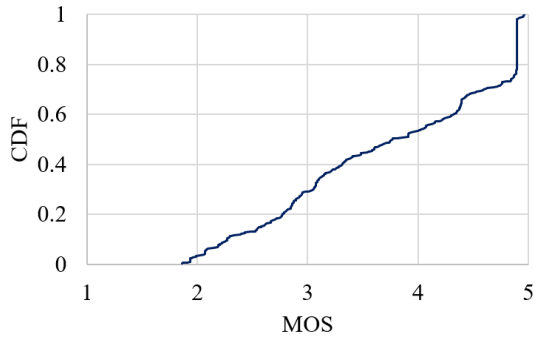
Figure 5.22: Selected specifics of Dataset And18L.

5.2.3 Model training and performance

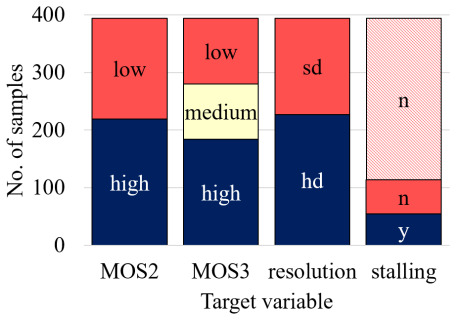
As different application-level metrics, measured at different granularities, may be relevant for use in QoE management strategies, the idea of this study was to assess to what extent different application-level KPIs and overall MOS can be estimated from IP-level data. Although the number of models we wanted to build was initially higher, after inspecting the collected dataset,

it became clear that building some models is redundant. For example, initial delays observed in the dataset were short enough not to impact QoE (according to existing QoE models from literature) in almost all of the video streaming instances (Figure 5.22b). Classifying, for example, videos with 2 s initial delay into a “long” initial delay class would not make sense. Finally, we decided to train models based on different algorithms that 1) classify MOS into 2 classes, 2) classify MOS into 3 classes, 3) classify resolution into 2 classes, and 4) classify stalling in 2 classes. Using Dataset And18L we built 12 different ML models.

The dataset And18L, used for building the models is presented in Figure 5.23. Figure 5.23a shows the distribution of MOS values in the dataset, as calculated using the ITU-T P.1203 model with the codec extension. In Figure 5.23b we show the number of instances that belong to the classes of each type of classification model. Given the distribution of stalling classes, we subset the dataset to balance the number of instances in each class. For each of the target variables and each of the used ML algorithms, namely OneR, J48, and Random Forest (RF), we first selected the relevant features using WEKA’s Wrapper method, and then trained the models and assessed them through cross-validation. The results in terms of accuracy, precision, and recall are listed in Table 5.8.



(a) MOS distribution.



(b) Distribution of classes for each type of classification model. The instances omitted due to subsampling are shaded.

Figure 5.23: Distribution of MOS and target variables in Dataset And18L.

Table 5.8: Performance comparison of different ML based models considering different targets.

Target variable	Alg.	Selected attributes (predictors)	Acc. [%]	Prec.	Rec.
MOS2 {high:h, low:l}	OneR	*stDevSizeIn1sIntervals*	80.203	h: 0.811 l: 0.790	h: 0.840 l: 0.754
	J48	*stDevSizeIn1sIntervals, avgPacketSize*	82.7411	h: 0.830 l: 0.824	h: 0.868 l: 0.777
	RF	*avgSizeIn5sIntervals, hMeanSizeIn5sIntervals, minSizeIn5sIntervals, maxSizeIn5sIntervals, hMeanSizeIn1sIntervals, lowMedianSizeIn1sIntervals*	81.2183	h: 0.828 l: 0.792	h: 0.836 l: 0.783
MOS3 {high:h, medium:m, low:l}	OneR	*maxSizeIn1sIntervals*	70.3046	h: 0.826 m: 0.489 l: 0.673	h: 0.853 m: 0.479 l: 0.649
	J48	*avgPacketSize, pStDevSizeIn5sIntervals, stDevSizeIn1sIntervals*	70.8122	h: 0.847 m: 0.494 l: 0.639	h: 0.870 m: 0.427 l: 0.684
	RF	*avgSizeIn5sIntervals, minSizeIn5sIntervals, maxSizeIn5sIntervals, pVarSizeIn5sIntervals, medianSizeIn1sIntervals, maxSizeIn1sIntervals*	71.8274	h: 0.839 m: 0.568 l: 0.625	h: 0.875 m: 0.438 l: 0.702
resolution {hd, sd}	OneR	*maxSizeIn1sIntervals*	81.4721	hd: 0.863 sd: 0.758	hd: 0.806 sd: 0.826
	J48	*maxSizeIn1sIntervals, medianSizeIn1sIntervals*	83.5025	hd: 0.849 sd: 0.815	hd: 0.868 sd: 0.790
	RF	*averageThroughput, avgSizeIn5sIntervals, lowMedianSizeIn5sIntervals, pStDevSizeIn5sIntervals*	82.2335	hd: 0.855 sd: 0.780	hd: 0.833 sd: 0.808
stalling {yes, no}	OneR	*medianSizeIn5sIntervals*	69.2982	y: 0.625 n: 0.853	y: 0.909 n: 0.492
	J48	*highMedianSizeIn1sIntervals*	70.1754	y: 0.640 n: 0.821	y: 0.873 n: 0.542
	RF	*hMeanSizeIn1sIntervals, highMedianSizeIn1sIntervals, minSizeIn1sIntervals, stDevSizeIn1sIntervals, pVarSizeIn1sIntervals*	69.2982	y: 0.672 n: 0.714	y: 0.709 n: 0.678

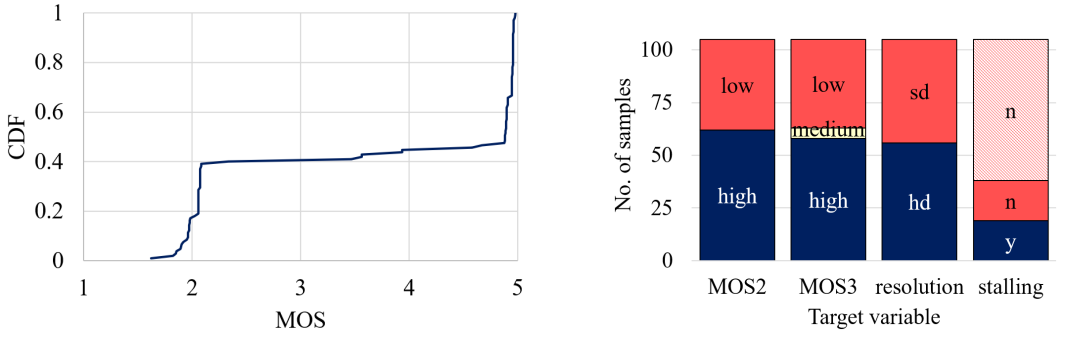
From the Table, it is visible that classifying MOS and video resolution into 2 classes achieves high accuracy (up to 83%). When classifying MOS into 3 classes, classification errors usually happen between classes “medium” and “low”, while instances from class “high” seem easily distinguished by the model. Stalling occurrence is not easily recognised in IP-level data, and we believe that the model does not really detect patterns related to stalling, but stalling can be detected to some extent due to the fact that it occurs in low-throughput conditions. Regarding the algorithms used to build models, we observe that more complex models introduce only a slight increase in performance, as compared to the OneR rule-based model. Complex models require more resources when utilised in a real network, and it seems questionable if it is worth using more resources for only a slight increase in accuracy.

5.2.4 Model performance on mobile network data

To address the issue of generalisation of the models presented in the previous section and trained in a lab environment using a WiFi connection, we test model applicability on data collected using probes in an operational mobile network. Dataset And17M was collected in September 2017 in the network of a large European ISP, with traffic traces obtained from a probe in the mobile network, and videos viewed on a Sony Xperia X device. The dataset contains application- and network-layer data corresponding to the streaming of 105 YouTube videos, with app-layer KPIs derived using ViQMon. As the network coverage was very good in the city area covered by the probe used to capture network traffic, we limited network type in certain scenarios to 2G or 3G on the smartphone so as to invoke degradations. However, due to the fact that the smartphone used for testing either connected to 4G or 2G and a 3G connection could not be established, calculated MOS (based on app-level measurements and the ITU-T P.1203 model implementation) was observed to be either very high or very low (Figure 5.24a). Ground-truth classes in Dataset And17M for each of the model types are shown in Figure 5.24b. Because of such distributions of MOS and QoE-influencing factors, the results when classifying the data from dataset And17M using models built on dataset And18L may be too optimistic (Table 5.9). While these results provide initial insights, a larger dataset collected in a mobile network, covering a wider variety of QoE degradations, is needed to obtain a more realistic picture of model applicability.

Table 5.9: Machine learning results - mobile.

Target variable	Accuracy [%]
MOS2 {high:h, low:l}	up to 100
MOS3 {high:h, medium:m, low:l}	up to 95.2381
resolution {hd, sd}	up to 94.2857
stalling {yes, no}	up to 76.3158



(a) MOS distribution.

(b) Distribution of classes for each type of classification model.

Figure 5.24: Distribution of MOS and target variables in Dataset And17M.

5.3 Session-level QoE/KPI estimation for iOS

Given the gap in research addressing QoE/KPI monitoring of YouTube delivered to iOS devices, in Study S3, performed in 2017 – 2018 and published in [17], we apply the methodology from Study S2 (focused on Android), to train the models for the iOS platform. Similarly to Study S2, the models trained on a dataset collected in the lab (iOS18L) are also validated on data collected in an operational mobile network (iOS17M and iOS18M). Besides the training of models focused exclusively on iOS, we investigate the need for training separate models for different platforms. For this purpose we validate the models trained for the Android platform (based on dataset And18L) on a dataset collected on iOS (iOS18L). Further investigation of the potential for the training of general models and applicability across different platforms is investigated in Section 7.1.

5.3.1 Study methodology

Measurement procedure and used datasets

For conducting the experiments to collect iOS18L, we used the same laboratory setup as depicted in Figure 4.3, with an iOS device (iPhone 6s) running YouTube and a screen recording app. As in Study S2, to emulate realistic bandwidth limitations, we scripted the IMUNES emulator to use the 4G/LTE bandwidth logs [172] published in [173]. As bandwidth was generally high in the logs, we divided the values in the log with the factors of 10, 20 and 30, thus creating four bandwidth envelopes in total. We played 100 YouTube videos over each of the four bandwidth envelopes (as was done with And18L in Study S2, and using the same set of videos). The exact number of videos in the final dataset is slightly smaller than 400, due to videos being removed or country restrictions changed over the course of the data collection period.

Our mobile network testbed consisted of an iPhone 6s running a screen recording app while playing YouTube videos, and a network probe running on the Gn interface for WCDMA, and

S11 and S1U interfaces for LTE. As there was no way to limit the bandwidth in this scenario and the area covered by the probe has a very good 4G coverage, to induce quality degradations, we resorted to limiting the client device to only connect to 2G or 3G (dataset iOS17M, collected in the network of an European ISP) and to limiting signal strength by using a mechanical barrier (dataset iOS18M, collected in the network of an Asian ISP).

Screen recordings captured at client devices during the aforementioned measurement campaigns were later processed using the ViQMon tool, which, as mentioned, uses OCR techniques to read data from the *Stats for Nerds* overlay, available in the YouTube app. From these parameters we can calculate various KPIs, and also MOS (Mean Opinion Score, corresponding to "overall QoE") according to ITU-T Recommendation P.1203 [97] using the implementation available online [99, 103, 104]. We again note that the model from ITU-T Recomm. P.1203 has been validated for H.264 video only, while for VP9 (which we often observed in YouTube) we used the Non-Standard Codec Extension for ITU-T P.1203 [105, 106].

Each of the datasets, for each of the video streaming instances contains calculated network traffic features and target labels i.e., ground truth classes. Network traffic features were calculated for the whole playback duration and include:

- averageThroughput{DL/UL} - average throughput,
- avgPacketSize{DL/UL} - average packet size,
- percOfUsedTransTimeDL - percentage of used transmission time (time window is considered to be "used" for transmission if all interarrival times are shorter than 0.1 s) [182],
- numOfPacketsLarger100BUL - number of uplink packets larger than 100 Bytes (the condition eliminates ACKs, thus counting segment requests) [182],
- avgSizeLarger100BUL, stdSizeLarger100BUL - average and standard deviation of uplink packet sizes, counting only the packets larger than 100 Bytes [182],
- {avg/hMean/median/min/max/stDev} SizeIn{5/3/2/1s} Intervals{DL/UL} - average, harmonic mean, median, min, max, standard deviation of volumes of downlink and uplink traffic in 5,3,2 and 1s-intervals.

While most of the features were used in Study S2 as well, several features were added based on the findings in related work that were published during the time between which we conducted Studies S2 and S3, with references given next to the new features' description.

We also list the 11 target variables used to classify each of the video instances, with corresponding class values:

1. MOS3: "high" $\geq 4 >$ "medium", $\geq 3 >$ "low",

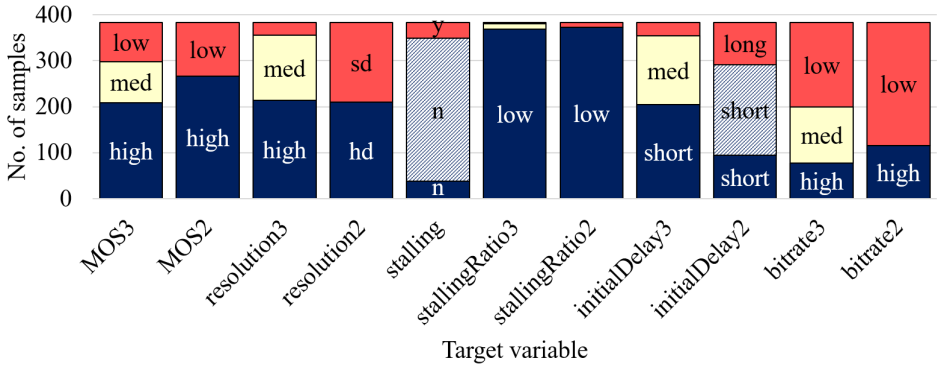
2. MOS2: "high" $\geq 3.5 >$ "low",
3. resolution3 (longest played resolution): "high" $\geq 720p >$ "medium" $\geq 360p >$ "low",
4. resolution2: "hd" $\geq 720p >$ "sd",
5. stalling: "y" (yes), "n" (no),
6. stallingRatio3 (ratio of stall time and overall video playback time): "low" $\leq 0.05 <$ "medium" $\leq 0.2 <$ "high",
7. stallingRatio2: "low" $\leq 0.1 <$ "high",
8. initialDelay3: "short" $< 1\ s <$ "medium" $\leq 5\ s <$ "long",
9. initialDelay2: "short" $\leq 1\ s <$ "long",
10. bitrate3 (average video bitrate): "high" $\geq 2000\ kbps >$ "medium" $\geq 1000\ kbps >$ "low",
11. bitrate2: "high" $\geq 1500\ kbps >$ "low".

We note that initial delays measured to be zero are due to the fact that *Stats for Nerds* data is refreshed at a 1-second granularity level, thus this value is in fact less than 1 second.

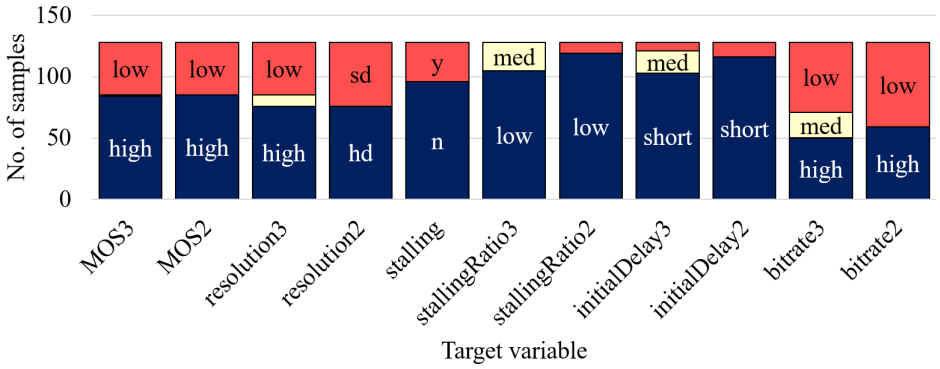
QoE/KPI classification methodology

We briefly analyse collected datasets, as certain observations are relevant for assessing the performance of ML models. Charts in Figure 5.25 show the number of instances classified into each of the classes, for each of the target variables, and for each of the four datasets. Additionally, we provide CDFs of MOS values for all of the datasets, to depict the range of QoE degradations captured in our datasets (Figure 5.26).

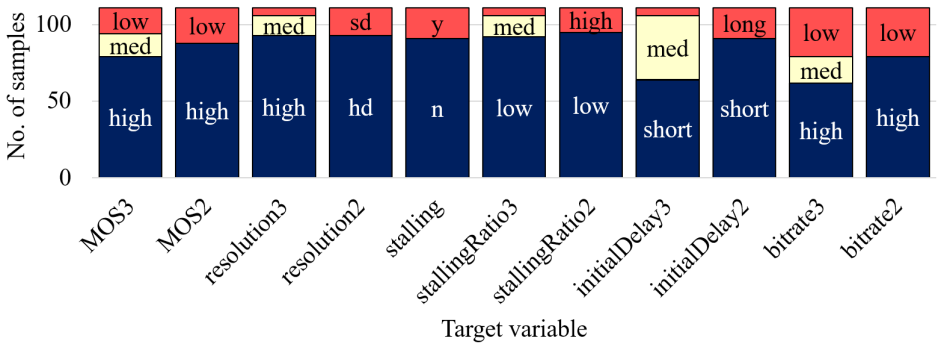
In datasets iOS18L and And18L (collected in the lab WiFi network), we observe a variety of MOS values, which resulted in a moderately balanced dataset with respect to MOS classifications. On the other hand, iOS17M is rather imbalanced. We see that MOS was either very low or very high, which results in the lack of instances in the MOS class "medium", but also in potentially too optimistic results when testing the models on this dataset. With iOS18M we captured a more balanced dataset in terms of MOS classification. In both datasets collected in the lab, "low" resolution (144p, 240p) was rarely observed. This is why we focused only on binary classification of resolution in model training. In dataset iOS17M, we observe lack of middle resolution instances, while in dataset iOS18M resolution was generally high.



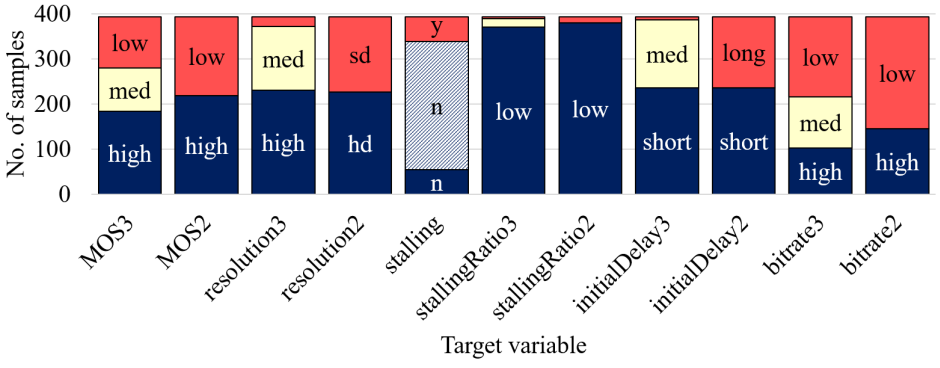
(a) Classes distribution in iOS18L.



(b) Classes distribution in iOS17M.



(c) Classes distribution in iOS18M.



(d) Classes distribution in And18L.

Figure 5.25: Distribution of video instances into classes. Given that for some target variables datasets were imbalanced, models were either not trained for these target variables, or instances were subset to balance out the dataset (pattern filled bars).

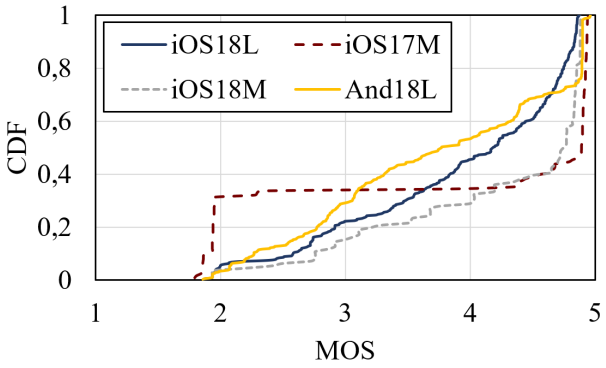


Figure 5.26: MOS distributions across datasets.

Stalling events were rare in all of the datasets. Due to the lack of samples with stalling events, we only trained the models that predict if there was any stalling (yes/no). Even in that case, we had to balance out the dataset by undersampling the dominant class before model training. Initial delays were generally short across all of the datasets, and when labelling instances into initial delay classes, we set rather low values of initial delay to be the boundaries between classes. We therefore classify initial delay only into 2 classes to test the feasibility of initial delay classification, but these classes do not necessarily represent good and bad quality. With respect to bitrate, all of the datasets had a fair distribution of instances among classes.

We used dataset iOS18L to train 7 types of classification models (out of 11 listed before in this section) using 3 ML algorithms. Some classifications were omitted due to dataset characteristics (low number of instances in certain classes). The 3 used algorithms are OneR (simple rule-based algorithm), J48 (decision tree), and Random Forest (a number of decision trees "voting" for a class). Model performance was assessed through 10-fold cross validation.

Referring back to Figure 1.3, we note that data collection requires the collection of traces involving various QoE degradation scenarios. Given that instrumenting this process in a lab environment requires less effort when compared to field data collection in a mobile network, we tested to what extent models trained on data from a lab WiFi network are applicable to traffic collected from operational mobile networks. For this purpose we tested iOS18L-trained models on iOS17M and iOS18M.

Finally, we tested the applicability of And18L–based models on iOS18L. Besides the obvious motivation for this being the convenience of using the same models across different platforms, we clarify another motivational point. ViQMon as a method for application-level performance measurement relies on extraction of information from video frames, which requires time and processing power. There is an alternative for Android which avoids video processing by using a Wrapper application [124] that copies *Stats for Nerds* info into a file. Being able to use the same models for iOS and Android would thus reduce time and effort needed for model training and re-evaluation.

5.3.2 Models focused on iOS platform

We trained 42 different ML–based classifiers: using 2 separate datasets (iOS18L and And18L), 7 different prediction targets (MOS classified into 2 and 3 classes, longest resolution into 2 classes, stalling occurrence into 2 classes, initial delay into 2 classes, and video bitrate into 2 and 3 classes), and 3 different ML algorithms (OneR, J48 and RF). We test the model performance through 4 scenarios.

We first assess the performance of iOS18L-trained models using 10-fold cross-validation. Table 5.10 summarises the results of this test. We provide information about the attributes selected in the feature selection procedure, achieved global accuracy, and precision and recall per class. From the Table, it can be observed that video bitrate is the most accurately predicted target (up to 91.91%), which seems logical, as it directly impacts the throughput. Most of the other targets can also be classified with more than 80% accuracy. With MOS classified into 3 classes, accuracy is just slightly lower (77.02%), which is expected due to a finer classification. The lowest is the prediction accuracy of initial delay, 70.43%.

When considering an ISP perspective, the aim may be to take action in cases when significant QoE degradations are estimated. In such cases, models with higher recall values in “low” and “medium” classes, and higher precision values in the class “high” are more appropriate.

Given the fact that initial delay depends only on the service behaviour at the beginning of the session, in following studies we added throughput–based network traffic features focused on the first part of the video, besides the features calculated for the whole playback time (Table 6.1). However, the lower performance of models classifying initial delays can also be attributed to the fact that initial delays were generally low in the dataset. The following studies demonstrate even shorter initial delays, thus we refrained from training initial delay classification models further on. To illustrate the statistical properties of traffic features with the most significant model impact, in Figure 5.27 we plot the distributions of the traffic feature used by the OneR algorithm across classes for selected KPIs. It can be seen that distributions are more separated in some cases (e.g., bitrate) and more overlapping in others (e.g., initial delay), which also impacts the accuracy of the classifier.

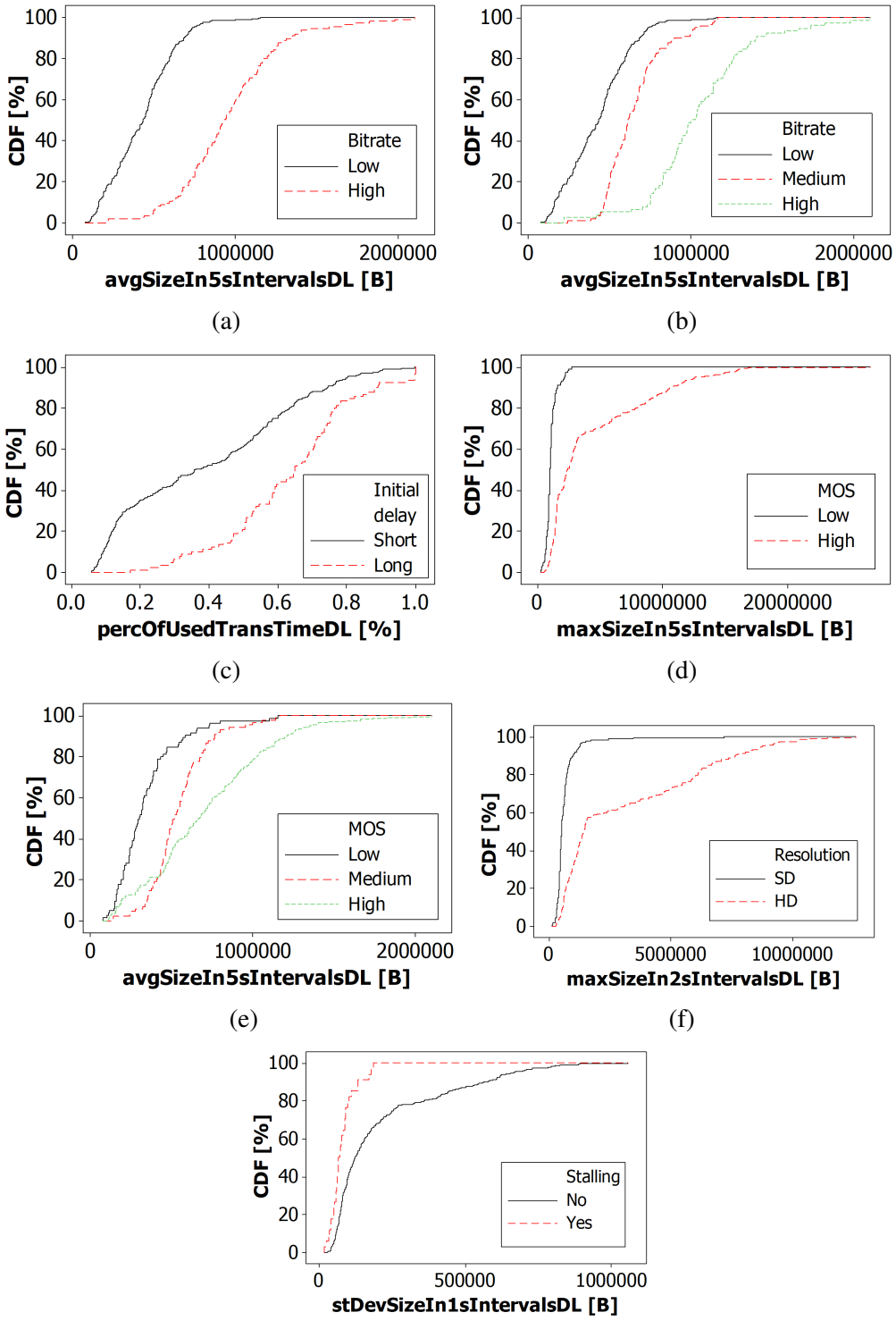


Figure 5.27: CDFs of traffic features for OneR algorithm for bitrate split into 2 a), and 3 classes b), initial delay c), MOS into 2 classes d), and 3 classes, e) resolution f), and stalling occurrence g).

Table 5.10: Performance comparison of different ML based models considering different targets. 10-fold cross-validation on dataset iOS18L.

Target variable	Alg.	Selected attributes (predictors)	Acc. [%]	Prec.	Rec.
MOS3 {high:h, medium:m, low:l}	OneR	*avgSizeIn5sIntervalsDL*	77.0235	h: 0.922 m: 0.645 l: 0.708	h: 0.766 m: 0.752 l: 0.808
	J48	*percOfUsedTransTimeDL, hMeanSizeIn5sIntervalsDL, stDevSizeIn5sIntervalsDL, maxSizeIn3sIntervalsDL, hMeanSizeIn2sIntervalsDL, maxSizeIn2sIntervalsDL*	76.5013	h: 0.866 m: 0.663 l: 0.643	h: 0.837 m: 0.618 l: 0.741
	RF	*avgPacketSizeDL, avgSizeIn3sIntervalsDL, hMeanSizeIn3sIntervalsDL, maxSizeIn3sIntervalsDL, avgSizeIn2sIntervalsDL, stDevSizeIn1sIntervalsDL, minSizeIn3sIntervalsUL*	76.5013	h: 0.855 m: 0.630 l: 0.69	h: 0.847 m: 0.652 l: 0.682
MOS2 {high:h, low:l}	OneR	*maxSizeIn5sIntervalsDL*	82.7676	h: 0.900 l: 0.692	h: 0.846 l: 0.786
	J48	*stDevSizeIn5sIntervalsDL, maxSizeIn3sIntervalsDL, maxSizeIn2sIntervalsDL, avgSizeIn5sIntervalsUL*	84.3342	h: 0.902 l: 0.724	h: 0.868 l: 0.786
	RF	*hMeanSizeIn2sIntervalsDL, maxSizeIn2sIntervalsDL, hMeanSizeIn1sIntervalsDL, avgPacketSizeUL, minSizeIn5sIntervalsUL, hMeanSizeIn3sIntervalsUL, minSizeIn3sIntervalsUL*	84.5953	h: 0.885 l: 0.754	h: 0.895 l: 0.735

Table 5.10: Performance comparison of different ML based models considering different targets. 10-fold cross-validation on dataset iOS18L.

Target variable	Alg.	Selected attributes (predictors)	Acc. [%]	Prec.	Rec.
resolution2 {hd, sd}	OneR	*maxSizeIn2sIntervalsDL*	78.8512	hd: 0.860 sd: 0.725	hd: 0.733 sd: 0.855
	J48	*percOfUsedTransTimeDL, numOfPacketsLarger100BUL, medianSizeIn5sIntervalsDL, medianSizeIn3sIntervalsDL, medianSizeIn1sIntervalsDL, maxSizeIn1sIntervalsDLm medianSizeIn5sIntervalsUL, minSizeIn5sIntervalsUL*	83.28	hd: 0.880 sd: 0.785	hd: 0.805 sd: 0.867
	RF	*medianSizeIn5sIntervalsDL, avgSizeIn2sIntervalsD, stDevSizeIn1sIntervalsDL*	81.4621	hd: 0.836 sd: 0.790	hd: 0.824 sd: 0.803
stalling {yes, no}	OneR	*stDevSizeIn1sIntervalsDL*	70.8333	y: 0.810 n: 0.667	y: 0.500 n: 0.895
	J48	*percOfUsedTransTimeDL, hMeanSizeIn3sIntervalsDL, maxSizeIn3sIntervalsUL*	80.5556	y: 0.833 n: 0.786	y: 0.735 n: 0.868
	RF	*avgPacketSizeDL, percOfUsedTransTimeDL, avgSizeIn2sIntervalsDL, avgSizeIn1sIntervalsDL, maxSizeIn1sIntervalsDL, stDevSizeIn1sIntervalsDL*	79.1667	y: 0.771 n: 0.811	y: 0.794 n: 0.789
initialDelay2 {short:s, long:l}	OneR	*percOfUsedTransTimeDL*	68.8172	s: 0.776 l: 0.639	s: 0.547 l: 0.835
	J48	*medianSizeIn1sIntervalsDL*	67.7419	s: 0.973 l: 0.604	s: 0.379 l: 0.989
	RF	*avgPacketSizeDL, percOfUsedTransTimeDL, avgSizeIn2sIntervalsDL, avgSizeIn1sIntervalsDL, maxSizeIn1sIntervalsDL, stDevSizeIn1sIntervalsDL*	70.4301	s: 0.700 l: 0.709	s: 0.737 l: 0.670

Table 5.10: Performance comparison of different ML based models considering different targets. 10-fold cross-validation on dataset iOS18L.

Target variable	Alg.	Selected attributes (predictors)	Acc. [%]	Prec.	Rec.
bitrate3 {high:h, medium:m, low:l}	OneR	*avgSizeIn5sIntervalsDL*	77.0235	h: 0.708 m: 0.645 l: 0.922	h: 0.808 m: 0.752 l: 0.766
	J48	*avgPacketSizeDL, avgSizeLarger100BUL, avgSizeIn5sIntervalsDL, hMeanSizeIn3sIntervalsDL, medianSizeIn3sIntervalsDL, maxSizeIn3sIntervalsDL, hMeanSizeIn2sIntervalsDL, medianSizeIn2sIntervalsDL, maxSizeIn1sIntervalsDL, avgSizeIn5sIntervalsUL, stDevSizeIn5sIntervalsUL*	86.423	h: 0.857 m: 0.819 l: 0.895	h: 0.846 m: 0.785 l: 0.924
	RF	*avgPacketSizeDL, stdSizeLarger100BUL, avgSizeIn5sIntervalsDL, medianSizeIn5sIntervalsDL, avgSizeIn3sIntervalsDL, medianSizeIn2sIntervalsDL, minSizeIn2sIntervalsDL, maxSizeIn2sIntervalsDL, maxSizeIn1sIntervalsDL, maxSizeIn5sIntervalsUL, hMeanSizeIn2sIntervalsUL, minSizeIn2sIntervalsUL, maxSizeIn2sIntervalsUL, maxSizeIn1sIntervalsUL*	87.9896	h: 0.863 m: 0.821 l: 0.925	h: 0.808 m: 0.835 l: 0.940
bitrate2 {high:h, low:l}	OneR	*avgSizeIn5sIntervalsDL*	89.0339	h: 0.870 l: 0.898	h: 0.750 l: 0.951
	J48	*avgSizeIn5sIntervalsDL, medianSizeIn2sIntervalsDL, stDevSizeIn1sIntervalsDL*	90.8616	h: 0.893 l: 0.914	h: 0.793 l: 0.959
	RF	*medianSizeIn5sIntervalsDL, maxSizeIn3sIntervalsDL, avgSizeIn2sIntervalsDL, stDevSizeIn1sIntervalsDL*	91.906	h: 0.897 l: 0.928	h: 0.828 l: 0.959

5.3.3 Model performance on mobile network data

In the second and the third test, we assessed the performance of iOS18L-trained models on datasets from mobile networks (iOS17M and iOS18M). The results are summarised in Tables 5.11 and 5.12. As we are using the same iOS18L-trained models as in the first test, selected attributes are the same as in the first test (as listed in Table 5.10) and thus omitted in these two tables. We note that tables 5.11 and 5.12 only report on the results achieved with the best performing model, with the exact algorithm used indicated in the table. On iOS17M we achieved excellent results, which can, to some extent, be attributed to iOS17M not being very balanced (the quality was either very good or very bad). We however note two problematic results. When classifying MOS into 3 classes, precision and recall on the class "medium" are equal to 0. This is because there was only one instance in MOS class "medium" in iOS17M, and it was misclassified. Another point is lowered precision in the classification of initial delay, which may be improved by considering only traffic in an initial time window.

Table 5.11: Performance of iOS18L-trained models on iOS17M.

Target variable	Alg.	Acc. [%]	Prec.	Rec.
MOS3 {high:h, medium:m, low:l}	RF	98.4375	h: 0.988 m: 0.000 l: 0.977	h: 0.988 m: 0.000 l: 1.000
MOS2 {high:h, low:l}	J48	99.2188	h: 1.000 l: 0.977	h: 0.988 l: 1.000
resolution2 {hd, sd}	J48	91.4063	hd: 0.882 sd: 0.977	hd: 0.987 sd: 0.808
stalling {yes, no}	OneR	90.625	y: 0.727 n: 1.000	y: 1.000 n: 0.875
initialDelay2 {short:s, long:l}	OneR	67.1875	s: 0.987 l: 0.212	s: 0.647 l: 0.917
bitrate3 {high:h, medium:m, low:l}	J48	87.5	h: 0.875 m: 0.667 l: 0.944	h: 0.980 m: 0.571 l: 0.895
bitrate2 {high:h, low:l}	J48	93.750	h: 0.932 l: 0.942	h: 0.932 l: 0.942

As dataset iOS18M is more balanced, the results of testing iOS18L-trained models on iOS18M may be more realistic (Table 5.12). Video bitrate is very accurately predicted, while with other targets we observe that the model accurately detects high-quality instances, while detection of low-quality instances should be improved. As collecting data for model training in a mobile environment requires significantly more effort, we are motivated to adjust the models trained on data from a lab network to work on data from mobile network. To see why recall was

Table 5.12: Performance of iOS18L-trained models on iOS18M.

Target variable	Alg.	Acc. [%]	Prec.	Rec.
MOS3 {high:h, medium:m, low:l}	OneR	75.6757	h: 0.875 m: 0.000 l: 0.452	h: 0.886 m: 0.000 l: 0.824
MOS2 {high:h, low:l}	RF	79.2793	h: 0.816 l: 0.500	h: 0.955 l: 0.174
resolution2 {hd, sd}	RF	86.4865	hd: 0.868 sd: 0.800	hd: 0.989 sd: 0.222
stalling {yes, no}	RF	86.4865	y: 0.868 n: 0.800	y: 0.989 n: 0.222
initialDelay2 {short:s, long:l}	J48	75.6757	s: 0.848 l: 0.316	s: 0.857 l: 0.300
bitrate3 {high:h, medium:m, low:l}	OneR	86.4865	h: 0.841 m: 0.636 l: 1.000	h: 0.935 m: 0.412 l: 0.969
bitrate2 {high:h, low:l}	RF	91.8919	h: 0.986 l: 0.795	h: 0.899 l: 0.969

low on low-quality instances in this test, future work should focus on collecting more data in a lab under low bandwidth conditions, adding it to iOS18L, and redoing the test. Additionally, if the problem is not in the lack of low-quality instances used for training, another idea is to further analyse the distributions of network traffic features with respect to target variables in WiFi and mobile scenarios and see if the model parameters can be adjusted to address the difference in service behaviour.

5.3.4 Cross-testing of Android-data–trained models on iOS data

Finally, in the fourth test we assess the applicability of And18L-trained models on iOS18L (Table 5.13). The table includes the results for the best-performing model, with the used algorithm indicated in the table. For most of the targets, results show only a slight decrease in prediction accuracy, when compared to the results from the first test. We detect significantly worse performance in the classification of stalling and initial delay, which may be due to the differences in YouTube's buffering behaviour on the two platforms, as they rely on different client logic and HAS standards – DASH and HLS. The exact reasons and possible mitigations will be addressed in future work. However, the test shows that most of the models trained on data collected using an Android device could also be applied on iOS data, thus reducing the effort in terms of collecting separate datasets for these two platforms. What is also interesting to notice is that, as compared to the features used in Study S2, where the same dataset was used for model traning,

here it can be observed that in a lot of cases the feature stdSizeLarger100BUL introduced after Study S2 and prior to conducting Study S3 was utilised (Table 5.13).

Table 5.13: Performance of And18L-trained models on iOS18L.

Target variable	Alg.	Selected attributes (predictors)	Acc. [%]	Prec.	Rec.
MOS3 {high:h, medium:m, low:l}	RF	*stdSizeLarger100BUL, maxSizeIn5sIntervalsDL, avgSizeIn2sIntervalsDL, medianSizeIn2sIntervalsDL, avgSizeIn1sIntervalsDL, medianSizeIn1sIntervalsDL, stDevSizeIn1sIntervalsDL, avgPacketSizeUL, medianSizeIn5sIntervalsUL, maxSizeIn5sIntervalsUL, maxSizeIn3sIntervalsUL, medianSizeIn1sIntervalsUL*	72.0627	h: 0.848 m: 0.568 l: 0.595	h: 0.804 m: 0.472 l: 0.776
MOS2 {high:h, low:l}	RF	*percOfUsedTransTimeDL, stdSizeLarger100BUL, maxSizeIn5sIntervalsDL, medianSizeIn1sIntervalsDL, averageThroughputUL, avgPacketSizeUL, avgSizeIn3sIntervalsUL*	83.5509	h: 0.939 l: 0.678	h: 0.816 l: 0.880
resolution2 {hd, sd}	RF	*stdSizeLarger100BUL, maxSizeIn5sIntervalsDL, avgSizeIn3sIntervalsDL, stDevSizeIn3sIntervalsDL, avgSizeIn2sIntervalsDL, stDevSizeIn2sIntervalsDL, avgSizeIn5sIntervalsUL, stDevSizeIn2sIntervalsUL*	78.8512	hd: 0.770 sd: 0.819	hd: 0.876 sd: 0.682
stalling {yes, no}	RF	*stDevSizeIn1sIntervalsDL, avgSizeIn2sIntervalsUL, maxSizeIn1sIntervalsUL, stDevSizeIn1sIntervalsUL*	73.8903	y: 0.170 n: 0.940	y: 0.500 n: 0.762
initialDelay2 {short:s, long:l}	J48	*highMedianSizeIn2sIntervalsDL*	76.2402	s: 0.762 l: 0.000	s: 1.000 l: 0.000
bitrate3 {high:h, medium:m, low:l}	OneR	*avgSizeIn1sIntervalsDL*	75.4569	h: 0.683 m: 0.620 l: 0.949	h: 0.910 m: 0.727 l: 0.707

bitrate2 {high:h, low:l}	RF	*averageThroughputDL, numOfPacketsLarger100BUL, avgSizeIn5sIntervalsDL, medianSizeIn1sIntervalsDL, maxSizeIn1sIntervalsDL, stDevSizeIn5sIntervalsUL*	88.7728	h: 0.779 l: 0.944	h: 0.879 l: 0.891

5.4 Real-time KPI estimation

While the studies presented in the previous three sections (Studies S1–S3) focused on QoE/KPI estimation on a per-video session level, a prerequisite for employing dynamic QoE-aware network management and invoking real-time QoE-centric resource allocation mechanisms would require 'real-time' KPI estimation. Such solutions would allow network operators to track video streaming KPIs in real-time, and act accordingly (e.g., allocate additional resources, re-route traffic) when certain thresholds are detected (e.g., reduced video resolution, stalling occurrences). We note that an important line of research is related to the actual root cause analysis (RCA) of video quality degradations, as they may result for various reasons (e.g., insufficient network resources, problems on the client device, CDN related problems). However, the RCA of quality degradations is considered out of the scope of this thesis, where our focus is on monitoring and QoE/KPI estimation. This section describes Study S5, conducted in 2019 – 2020 and published in [16], aimed at developing a methodology for the training of real-time KPI estimation models. The section presents results in terms of model performance on a selected use-case involving dataset And19L, collected in the scope of Study S4 (Chapter 6).

5.4.1 Methodology

Based on ideas presented in related work on buffer estimation and stalling prediction [127, 141, 182], we defined a network traffic feature set containing 228 features. These features include various network-level statistics calculated using only the IP addresses and packet sizes from the traffic trace, making the methodology applicable for various services, platforms, protocols, etc. The core idea is to train models that estimate KPIs for each second of the video streaming session, based on network traffic features calculated on the traffic exchanged during that second, but also wider time-windows, including traffic exchanged in intervals preceding the observed 1 s interval. The problem of KPI estimation is cast as a classification problem, described further on. We note that other studies addressing real-time KPI estimation [126, 127, 128, 129, 139, 140, 141], emerging at the same time as this study was conducted, were focused on YouTube accessed via the browser, whereas our focus was on YouTube accessed via the native YouTube app.

The description of used network traffic statistics is given in Table 5.14. Each of the statistics is calculated for both downlink and uplink traffic, and on window sizes of 1, 2, 3, 5, 10, and 20 seconds. To clarify the notion of the window, consider the following example: A 5 s window feature is calculated on a 5 s interval of network traffic, where the most recent of the 5 seconds is the one we are trying to classify into a target KPI class. To indicate the exact statistic, direction, and window size, we use the following convention for naming the features: `<direction>_<statistic>_<window_size>`. For example, `dl_max_iat_w20` refers to a feature

calculated as maximum downlink packet interarrival time, calculated based on traffic captured in the last 20 seconds.

Table 5.14: Condensed list of network traffic features used for real-time KPI classification. Each feature is calculated for both downlink and uplink traffic, and on windows of 1, 2, 3, 5, 10, and 20 seconds, constituting the full set of 228 features.

Feature name	Description
pckt_count	Packet count.
pckt_count_gt100	Packet count where all the packets smaller than 100B are ignored (mostly acknowledgements).
throughput	Average throughput.
active_time	Percentage of the window that is used for transmission; interval is considered as used if interarrival time between two consecutive packets in one direction is not longer than 100ms.
mean_pckt_size_gt100	Mean packet size in the window; packets smaller than 100B are ignored.
median_pckt_size_gt100	Median packet size in the window; packets smaller than 100B are ignored.
max_pckt_size_gt100	Maximum packet size in the window; packets smaller than 100B are ignored.
min_pckt_size_gt100	Minimum packet size in the window; packets smaller than 100B are ignored.
stdev_pckt_size_gt100	Standard deviation of the packet size in the window; packets smaller than 100B are ignored.
mean_pckt_size	Mean packet size in the window.
median_pckt_size	Median packet size in the window.
max_pckt_size	Maximum packet size in the window.
min_pckt_size	Minimum packet size in the window.
stdev_pckt_size	Standard deviation of the packet size in the window.
mean_iat	Mean interarrival time in the window.
median_iat	Median interarrival time in the window.
max_iat	Maximum interarrival time in the window.
min_iat	Minimum interarrival time in the window.
stdev_iat	Standard deviation of the interarrival time in the window.

The dataset used for KPI estimation model training (And19L), includes both features corresponding to each interval, as well as ground-truth labels. As stalling was rarely observed, we focused on resolution and video bitrate classification to demonstrate our approach. Each 1-second interval instance (row in the dataset) consists of 228 feature values and 2 labels: resolution classified into 2 classes (“sd”/“hd”), and bitrate classified into 2 classes (“low”/“high”). Class “hd” corresponds to intervals in which the played video resolution was 720p or higher,

while in the case of bitrate, class “high” corresponds to instances with played video bitrate higher or equal to 1000 kbps. The dataset contains 49825 instances, each sample corresponding to 1 second of video playback. Figure 5.28 shows the distribution of instances across classes.

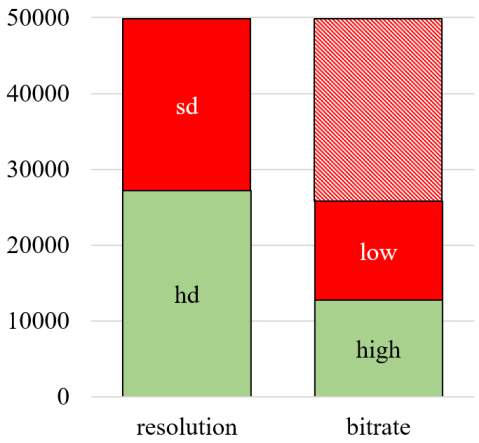


Figure 5.28: Distribution of instances across real-time KPI classes in dataset And19L. Instances omitted to balance out the number of samples are shaded.

To train real-time resolution and bitrate classification models, we have taken the following steps: (1) balancing out the number of samples per class by subsampling the dominant class (only in the bitrate classification case), (2) splitting the dataset into training (67%) and validation (33%) set, (3) selecting the 10 most relevant features for each classification by using 2 common approaches: Feature-Importance–based selection (FI) [166] and Sequential Feature Selection (SFS) [181], and (4) training the models by using 2 algorithms: Decision Tree (DT) and Random Forest (RF). We train and test 8 models in total, addressing 2 aforementioned classifications using 2 feature selection methods and 2 ML algorithms.

The results in terms of model performance are often dependent on the complexity of the feature selection process and on the complexity of the algorithm used to train the model. We compare model performances while relying on two commonly used feature selection methods that differ greatly in the amount of resources needed – selection based on feature importance and sequential feature selection. We also compare results achieved with two commonly used algorithms, Decision Tree, and Random Forest consisting of 1000 trees. We note that all median features were omitted, as their calculation is memory-intensive and thus not preferred for potential in-network utilisation.

5.4.2 Results

Table 5.15 summarises the results in terms of resolution classification models’ performance on the validation set. The table shows precision, recall, and accuracy values for all four cases depending on whether FI or SFS was used for feature selection, and on whether DT or RF was used for model training. SFS is significantly more computationally-intensive than FI (for

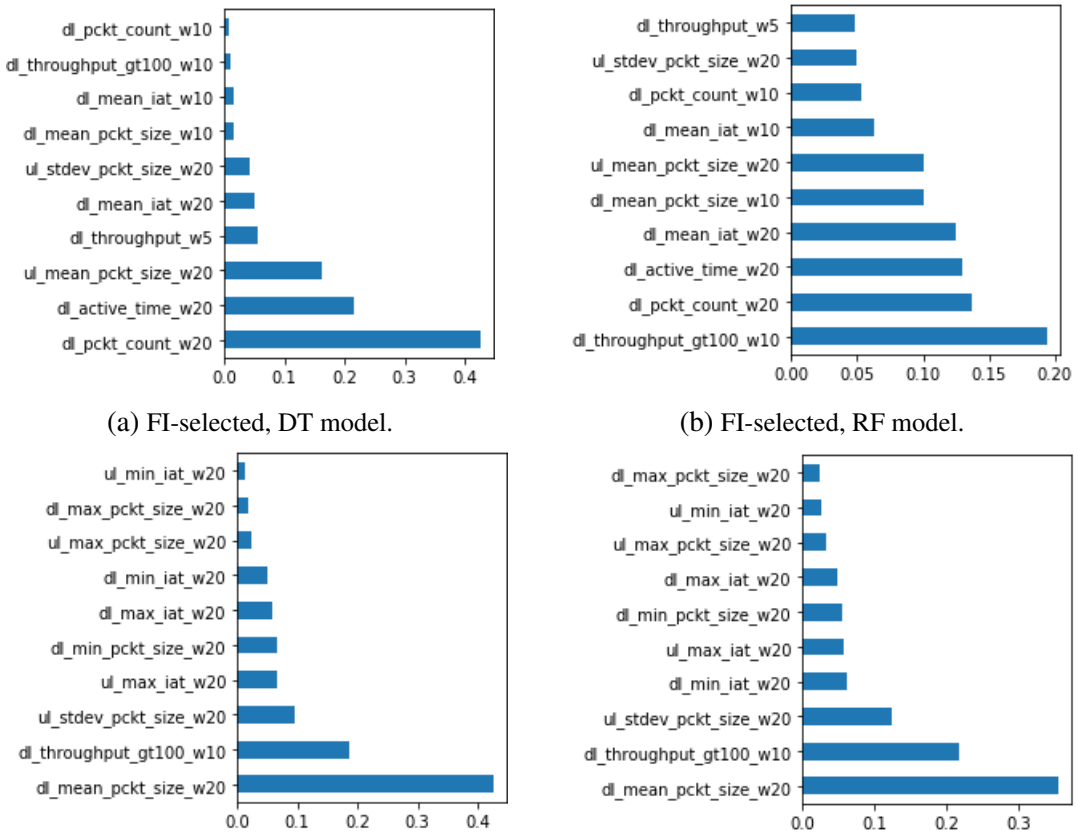
comparison purposes only, taking a few hours to complete, as compared to a second for FI run on the same PC). However, as feature selection is done only in the model training phase, and models with SFS–selected features perform significantly better, it makes sense to favour SFS. On the other hand, the performance of RF–trained models is comparable to that of DT–trained models. Given that we are dealing with models that need to be utilised and deployed in the network and run in real-time, simpler models are preferred, thus yielding the conclusion that DT may be a better option.

Table 5.15: Performance of real-time resolution classification models trained and tested using the And19L dataset.

		Algorithm		
		DT	RF	
Feat. selection	FI	hd: 0.80 sd:0.74	hd: 0.82 sd:0.74	Prec.
		hd: 0.78 sd:0.76	hd: 0.77 sd:0.79	Rec.
		0.77	**0.78**	Acc.
	SFS	hd: 0.87 sd:0.83	hd: 0.89 sd:0.81	Prec.
		hd: 0.86 sd:0.84	hd: 0.83 sd:0.87	Rec.
		0.85	**0.85**	Acc.

Figure 5.30 shows the importance of features in each trained model. In subfigures 5.30a and 5.30b, the feature list is the same, given that in both cases the same FI-based selection was used. However, the importance of these features in trained models differs. The same is true in case of SFS-based selection (subfigures 5.30c and 5.30d). It can be observed that the most important features in all four cases mostly are the same, but less important features are different depending on whether FI or SFS was used. This indicates that models based on SFS benefited from the exhaustive search of the feature space through employing less important but still very relevant features.

In case of bitrate classification (Table 5.16), there are no major differences in performance of the four models, regardless of the feature selection method and training algorithm. The most important features are mostly the same for all four models, while other features differ (Figure 5.29). However, it may be concluded that these less important features, in case of bitrate classification, do not contribute much to the model performance anyway, as the relationship between video bitrate and traffic volume is more straightforward than in the case of resolution.



(a) FI-selected, DT model.

(b) FI-selected, RF model.

(c) SFS-selected, DT model.

(d) SFS-selected, RF model.

Figure 5.29: Feature importances in real-time video bitrate classification models.

5.5 Practical challenge: Session delimitation

Going back to session-level QoE/KPI estimation, this section addresses the practical challenge of session delimitation, i.e., detection of the *start* and the *end* of the video playback in the traffic, which is a prerequisite for applying the models described in Sections 5.1 – 5.3. This problem has been investigated in [139], where the authors delimit sessions based on (1) a spike of non-video traffic in downlink at the start of the session, and (2) no video traffic at the end of the session. The authors attribute the aforementioned spike to the download of the player code or to the download of the catalogue web page. However, this solution is only viable in the case when a user is using Web–based streaming applications, and refreshing the web page to access the catalogue before watching another video. There are also limitations with regards to the session end detection, as the amount of content in the buffer will greatly depend on the available bandwidth, and thus the amount of time at the end of the playback in which no content is downloaded can vary.

As an initial step towards finding a more generic and robust solution, we tried applying ML on 1-second intervals of network traffic for the purpose of detecting whether a start or an end of the video has occurred, and doing so on a dataset collected in a variety of network

Table 5.16: Performance of real-time video bitrate classification models trained and tested using the And19L dataset.

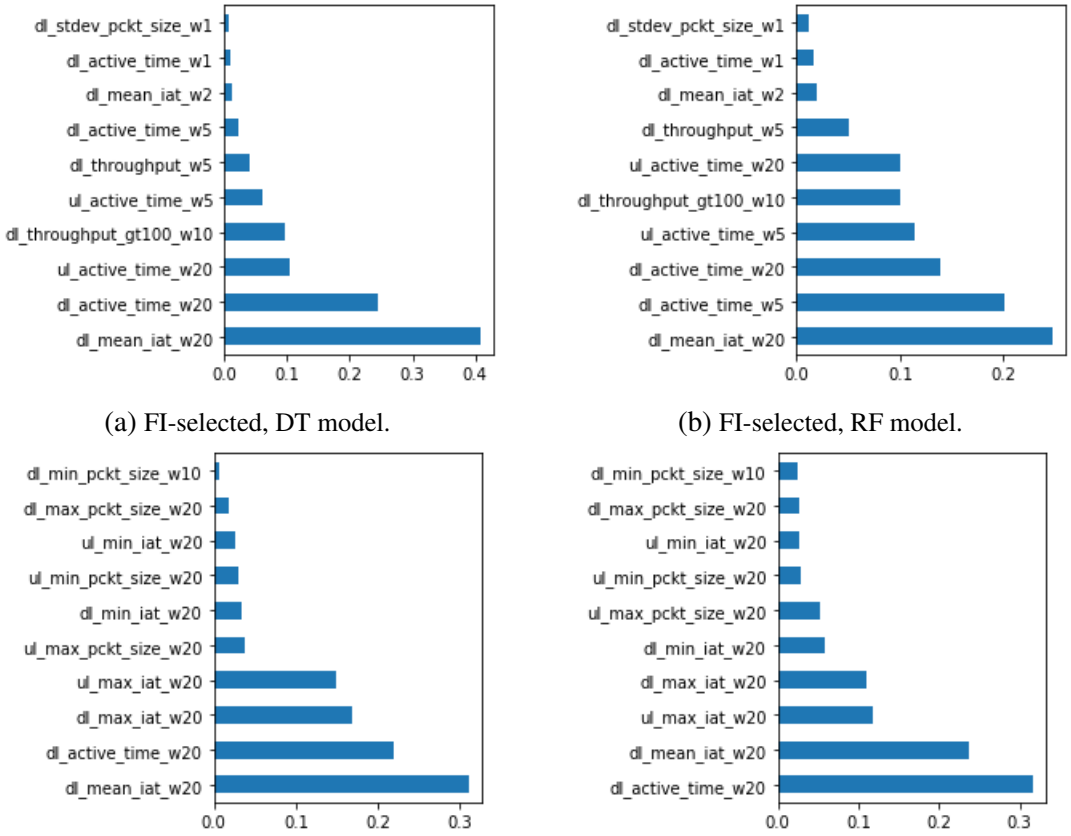
		Algorithm		
		DT	RF	
Feat. selection	FI	high: 0.84 low:0.89	high: 0.85 low:0.89	Prec.
		high: 0.89 low:0.84	high: 0.89 low:0.85	Rec.
		0.86	**0.87**	Acc.
	SFS	high: 0.87 low:0.89	high: 0.88 low:0.89	Prec.
		high: 0.88 low:0.87	high: 0.89 low:0.88	Rec.
		0.88	**0.89**	Acc.

conditions. This analysis was performed later during the thesis research, and uses the dataset And19L (described in more detail in Chapter 6), as opposed to Studies S1 – S3, which used datasets collected in 2018.

We labelled the real-time features (as listed in Table 5.14) of the And19L dataset with classes "start", "end", and "other". As all data was collected with possible time-shift of up to 1 second (due to the precision of data collection methods), we labelled the first three intervals as "start", the last three as "end", and otherwise as "other". We note that for session-level QoE/KPI estimation, a delimitation error of up to a few seconds is likely not to be considered critical. We split the dataset in half, balanced out the number of instances per class in the first half, selected traffic features using Sequential Feature Selection (SFS) [181], and trained the models using Random Forest. The models were tested on other half as a whole, without balancing.

We first list the selected features as follows:

- ul_min_pckt_size_gt100_w2,
- ul_max_pckt_size_gt100_w3,
- ul_min_pckt_size_gt100_w3,
- dl_mean_pckt_size_w5,
- ul_min_pckt_size_gt100_w5,
- ul_max_pckt_size_w5,
- dl_active_time_w10,
- ul_min_pckt_size_gt100_w10,
- ul_max_pckt_size_w10,



(a) FI-selected, DT model.

(b) FI-selected, RF model.

(c) SFS-selected, DT model.

(d) SFS-selected, RF model.

Figure 5.30: Feature importances in real-time resolution classification models.

- ul_throughput_w20.

Using these features, we achieved an accuracy of 83 %. Although this would mean that in a 2-minute-long video, over 20 one-second intervals would be misclassified, further inspection of the predictions shows that it is still a viable approach. The results showed some typical classification errors that can be addressed by simple postprocessing of the predictions:

- More than 3 “start” or “end” intervals labelled as “start”/“end” – the algorithm detects the initial burst and the depleting phase – this does not affect the ability of detecting the *start* or *end*.
- Instances between classes “start” and “end” are confused – in our measurements, videos were played one after the other, which may be the reason for this confusion – the predictions still offer enough knowledge to split between the videos.

We also inspected the benefits of adding feature windows that succeed the interval window, as session-level QoE/KPI estimation does not necessarily need to happen right after the session is finished. We added the same features calculated on windows surrounding the instance interval (the instance interval is in the centre of the window), and the same features calculated

on windows succeeding the instance interval (the instance interval is the first interval in the window). The accuracy increased to up to 92%, with misclassifications following the same patterns as described earlier. However, we note that the performance increase may originate from the model detecting spikes after the end of the video, thus making it only applicable in the case when videos are played one after the other. While these are only initial results, they show the potential of using ML for session delimitation, and present challenges that are yet to be addressed.

5.6 Chapter summary

In this chapter, we have presented four studies aimed at developing ML–based models for in-network YouTube QoE/KPI estimation, addressing different platforms (Android, iOS), network types (WiFi, mobile), at different temporal granularities (per-video, per-second).

In the initial study (Study S1), we have proven the feasibility of QoE classification of adaptive YouTube video sessions using a ML approach and based on TCP- and IP-level features computed from encrypted network traffic flows. The YouQ system was developed for the purposes of measuring, storing, and processing application- and network-level data. Based on a large dataset collected using the YouQ system, a proof-of-concept solution has been proposed for QoE classification in the case of YouTube content accessed via a Web browser running on an Android device and via a WiFi network. The classification accuracy reached 84% for three classes and up to 91% for two QoE classes, hence leading to very promising results. The study also yielded insights into the YouTube client buffer mechanism, providing a better understanding of the service behaviour. While the study was focused on a single use-case, the proposed methodology is generic, and served as a basis for later improvements and adaptations throughout this thesis' research.

In Study S2 we addressed the challenge of client-side monitoring of app-level KPIs impacting QoE, and showed that different approaches used in existing monitoring apps result in different YouTube application behaviour and traffic characteristics. More specifically, approaches relying on embedding YouTube videos for test purposes and utilising available YouTube APIs exhibit different characteristics as compared to the YouTube app. We therefore proposed the ViQMon monitoring approach that utilises OCR libraries to extract application-level KPIs from video recordings (with the *Stats for Nerds* option enabled) captured while watching YouTube videos with the YouTube app. A further benefit of this approach is that it is applicable across different platforms (Android and iOS).

The study addressed the challenge of session-level QoE/KPI classification, focused on the Android platform, using ML-based models with input relying solely on IP-level network traffic features. As such, the models may be trained using both TLS/TCP and QUIC/UDP traffic. Ad-

ditional improvements of the methodology, with respect to the initial study, include the usage of more realistic bandwidth conditions during the measurement campaign, based on actual network bandwidth measurements found in literature, and also the incorporation of a standardised QoE model published in ITU-T Recomm. P.1203. Results related to trained models' performance showed promising accuracy when classifying MOS in 2 or 3 classes, longest resolution played into 2 classes, and stalling occurrence into 2 classes. Finally, built models were used to classify MOS and KPIs using a dataset collected in an operational mobile network.

The focus of Study S3 was on session-level YouTube QoE/KPI estimation for iOS devices. Using the data collected in a WiFi lab network, we trained ML models that classify QoE and various KPIs of video streaming sessions on the iOS platform. Model performance was first assessed through cross-validation, and then tested on 2 additional datasets collected in mobile networks to evaluate the applicability of WiFi-trained models. We achieved promising results in all 3 of these tests, with prediction accuracies, depending on the classification target, ranging from roughly 70 to 90%. We also test the models trained on data from an Android platform on data from an iOS platform, both collected in a WiFi environment. The results show that some of the models are highly applicable across platforms, while on others we observe limited applicability.

The last study presented in this chapter is Study S5, addressing real-time KPI estimation, on YouTube viewed on an Android device as a use-case. In the study we devised a set of network traffic features applicable for the real-time KPI estimation setting, trained the models that classify video resolution and bitrate on a per-second level, and compared results achieved by using different feature selection methods and different algorithms for model training. The results show high accuracy when classifying both KPIs into two classes – up to 85% for resolution, and up to 89% for bitrate. The usage of SFS in the case of resolution classification yielded better results in comparison to FI–based feature selection. The selection of feature selection method in the case of bitrate did not make a lot of difference. As for used algorithms, the results show comparable results for RF and DT in all test cases for an individual KPI, which suggests that DT may, for its simplicity, be a better choice for in-network utilisation on traffic data streams.

Additionally, we address the practical challenge of session delimitation, which is a prerequisite for performing session-level QoE/KPI estimation. In a brief evaluation, we demonstrate that machine learning could also potentially be employed for solving this issue.

Chapter 6

Impact of the inclusion of context data on QoE/KPI estimation performance

Models presented in the previous chapter clearly indicate that gaining insights into video streaming performance is possible by employing ML–based models on encrypted traffic streams. This chapter builds on top of these findings and investigates the potential gains of including context data as additional predictors. In practice, there is commonly a lack of cooperation between OTT providers and ISPs that would facilitate the exchange of data relevant for performing end user QoE-centric service and/or network management. In this chapter, we thus motivate the data exchange with models that, by utilising simple context information provided by an OTT service provider, bring more accurate in-network QoE/KPI estimation. The initial results of this study (Study S4) were published in [18], but the work on the problem was extended through 2019 – 2020. Section 6.1 explains the motivation behind the study and context data that can potentially be utilised, while the following sections describe the methodology used in this study (Section 6.2), and the results with respect to three classification model types: for classifying MOS into 3 classes, longest played resolution into 2 classes, and average video bitrate into 2 classes (Section 6.3). Section 6.4 summarises the chapter.

6.1 Introduction of context features

While OTT providers have insights into service performance and content characteristics, this data is commonly not shared with ISPs. As meeting end users' QoE is of interest to both service and network providers, the question arises as to what would be the potential incentives and benefits resulting from certain amounts of data shared between the two parties? In practice, there are commonly various regulatory, business, security, and privacy-related issues preventing OTTs from disclosing information about their streams, and/or preventing ISPs from sharing information about their networks. Nevertheless, various research efforts have addressed technical

and business aspects of OTT-ISP cooperation and information exchange, proposing the means for enabling cooperation in terms of architecture, communication [65], and incentives through potential revenue increase [83].

Collaborations among OTTs and ISPs in general are common, but not QoE-centric. ISPs and OTTs often have peering agreements, either through Public Peering Interconnections (PPI) through Internet eXchange Point (IXP) or through Private Network Interconnection (PNI). Peering was traditionally free, when the inbound traffic was roughly balanced with the outbound traffic of an Autonomous System (AS). As this is now not the case for Content Delivery Networks (CDNs) and content-heavy ISPs, operators started offering a paid peering service for those that did not meet their peering prerequisites. Another common practice for OTTs is to provide the ISPs with surrogate servers, to bring the content as close as possible to end users and thus decrease end-to-end latency and decrease the ISPs transit costs. Most of the collaborations of this type are free for both parties and based on mutual benefits [206][159].

When considering QoE-centric collaborative approaches, such can be classified into following classes [146]:

- Application-level optimisation - an application entity receives information about application and/or network performance and decides whether to adjust application parameters or to invoke network-level mechanisms,
- Network-level optimisation - a network entity receives information about application and/or network performance and decides whether to execute network-level mechanisms or to instruct application on how to adapt,
- Policy manager optimisation - a policy manager receives information about applications and network performance and orchestrates control actions for applications and the network based on its "global" system view.

Most of the OTT-ISP cooperation solutions proposed in literature portray the case when OTTs provide the ISPs access to key performance indicators (KPIs), such as information about video stalling. Considering the mode of communication between the OTT and ISP, this can be implemented in three ways [159]:

- the OTT actively sends information to the ISP,
- the ISP actively requests information from an OTT server,
- information is sent along the regular transmissions so that it can be passively monitored.

For the first two options, an API needs to be specified to exchange the information between ISP and OTT, while the last option requires fundamental changes in how HTTPS transmissions work.

The question remains, however, which data should be shared, so as to achieve relevant benefits for all involved stakeholders. This study is a step forward towards bridging this gap, focusing specifically on investigating to what extent the accuracy of ML–based in-network QoE

estimation models could be improved with the availability of additional application-level context data provided by the OTT provider. The underlying assumption is that improved in-network QoE monitoring can lead to improved root cause analysis and more efficient QoE management, which would be beneficial to both service and network providers. We identify four categories of QoE-influencing factors that may be shared by an OTT provider:

- Content-related: video category, average bitrate on all layers, number of views,
- Playback-related: information about stalling, played quality levels, initial delay,
- User-related: interaction (e.g., skipping to another video), player settings (e.g., autoplay on/off),
- System-related: end user device, end-to-end delay.

While playback-related and user-related data would provide clear insights into users' QoE by utilising existing QoE models (such as ITU-T P.1203) [97], in the context of the issues that need to be alleviated to enable the data exchange, exchanging content- and system-related data may be a more realistic start. Another concern for OTTs would clearly be the complexity of collecting and sharing data. For example, playback-related data, such as changes in played quality levels or the occurrence of stalling events, has a dynamic and real-time aspect to it, while content-related data is in its nature static per certain video. Following this rationale, we decided to assess the benefits of sharing platform and video encoding bitrate information, as these parameters are static in their nature and do not require information related to monitoring of the entire video streaming session.

Again using YouTube as a use-case, we enrich the features of datasets with the following information:

- video encoding bitrate for each H.264–encoded video in resolutions 144p, 240p, 360p, 480p, 720p, and 1080p (**6 new features**) – these parameters, obtained by querying the YouTube service, are different for each video (each video is identified with a unique ID in the context of YouTube), and may indicate the spatial and temporal complexity of the content,
- iOS/Android platform indicator (**1 new feature**) – the information about the client platform can be used in the network so as to enable the selection of an appropriate platform-specific model, or can be introduced as an additional feature.

For the purpose of testing the impact of these features on QoE/KPI estimation models, we collected two datasets, And19L and iOS19L, evaluating the benefits from introducing context features for both platforms. The following section provides a detailed description of the collected datasets, and specifies the used methodology.

6.2 Methodology

6.2.1 Data collection and analysis

We collected two datasets (Feb/March 2019), And19L and iOS19L, consisting of network traffic and application-level performance measurements (based on ViQMon) corresponding to the playout of 100 different videos played using the official YouTube app on both Android and iOS platforms, in 3 differently defined bandwidth conditions. This sums up to 300 videos per dataset (100 videos at 3 bandwidths), but due to video properties being changed over the course of data collection, or videos being deleted by owners, each dataset consists of 299 videos. Selected YouTube videos included a range of content types, from dynamic music videos, static music videos (e.g., album covers, lyrics videos, etc.), entertainment videos (e.g., gaming, movie trailers, comedy), lifestyle videos (e.g., howto and style, people and blogs), and informative videos (e.g., news and politics, sports). Videos also varied in duration, as depicted in Figure 6.1.

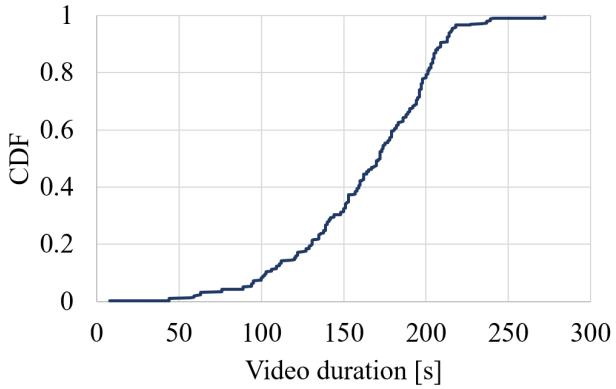


Figure 6.1: The cumulative distribution function of the durations of 100 videos included in And19L and iOS19L.

To emulate network conditions and induce QoE degradations that may occur in mobile networks, as in previous studies, we used the lab setup depicted with Figure 4.3, and scripted the IMUNES [169] network emulator and simulator to limit the bandwidth in our lab environment according to 3 bandwidth envelopes:

- 3G bandwidth [174, 175],
- 4G bandwidth [172, 173], and
- bandwidth capped at 1.5 Mbps, as observed in zero-rated streaming services offered by some network operators.

We emphasise the evolution of the overall research methodology in the data collection aspect, going from datasets collected in 39 different bandwidth envelopes in 2016 – 2017, to shaping the bandwidth in accordance with 4G bandwidth measurements in 2018 – 2019 and

inducing degradations by dividing the bandwidth values by factors of 10, 20 and 30, to finally using 4G, 3G, and 1.5 Mbps bandwidth scenarios, thus making all scenarios completely realistic.

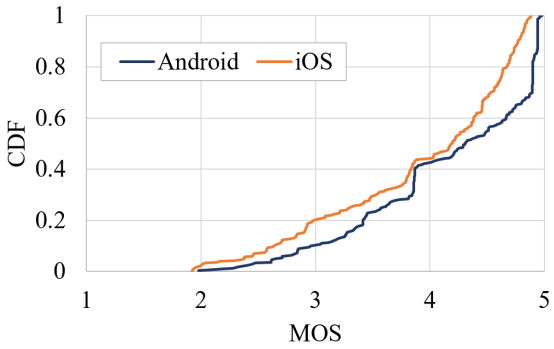
From the network traffic, we calculate traffic features as listed in Table 6.1. Compared to the previous study (Study S3), we introduced a new group of 7 throughput features: *dl_avg_throughput_first5s*, *dl_avg_throughput_first10s*, *dl_avg_throughput_first_half*, *dl_avg_throughput_second_half*, *dl_avg_throughput_first20perc*, *dl_avg_throughput_last20perc*, and *dl_avg_throughput_mid20perc*, aiming to gain more information from the amount of traffic that is being downloaded in different parts of the session.

Table 6.1: Condensed list of network traffic features used for session-level QoE/KPI classification. The full set consists of 62 features.

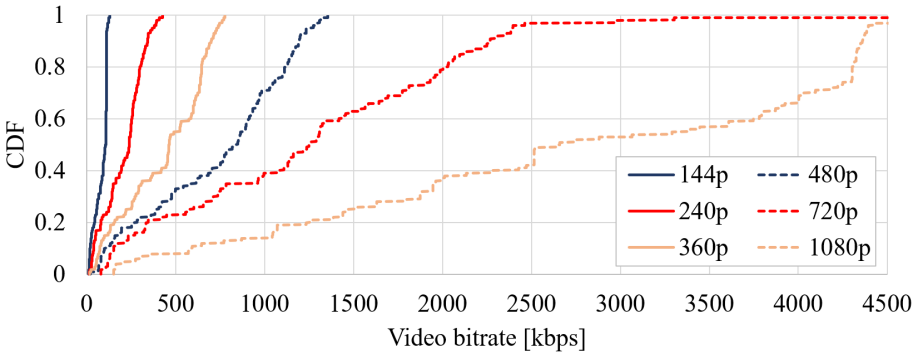
Feature name	**Description**
[dl, ul]_ avg_throughput	Average downlink and uplink throughputs in the whole session.
dl_avg_throughput_ [first5s, first10s, first_half, second_half, first20perc, last20perc, mid20perc]	Average downlink throughput in first 5 and 10 seconds of the session, in first and last 50% of the session, first, last and middle 20% of the session.
[dl, ul]_ mean_pckt_size	Average downlink and uplink packet size in the whole session.
dl_active_time	Percentage of the session duration that is used for transmission; interval is considered as used if interarrival time between two consecutive packets in one direction is not longer than 100ms.
ul_pckt_count_gt100	Session packet count, where all the packets smaller than 100B are ignored (mostly acknowledgements).
ul_mean_pckt_size_gt100	Mean packet size in the session; packets smaller than 100B are ignored.
ul_stdev_pckt_size_gt100	Standard deviation of the packet size in the session; packets smaller than 100B are ignored.
[dl, ul]_ [mean, hmean, median, min, max, stdev]_ data_in_ [5s, 3s, 2s, 1s]_intervals	Mean, harmonic mean, median, minimum, maximum and the standard deviation of data amounts bucketed in 5–, 3–, 2–, and 1–s intervals, for downlink and uplink traffic.

To compare the accuracy of ML models in cases with and without access to additional static application-level context data, we enriched the set of features (predictors) for each video session with: 1) platform info ("Android", "iOS"), and 2) video encoding bitrate *per quality level* denoted by YouTube with identifiers (itags) 160, 133, 134, 135, 136, and 137 (corresponding to H.264 video with resolutions 144p, 240p, 360p, 480p, 720p, and 1080p, respectively), obtained

using *youtube-dl* [207]. The distribution of encoding bitrates across the datasets is given in Figure 6.2b.



(a) MOS distribution for 299 played video sessions per each dataset (And19L, iOS19L).

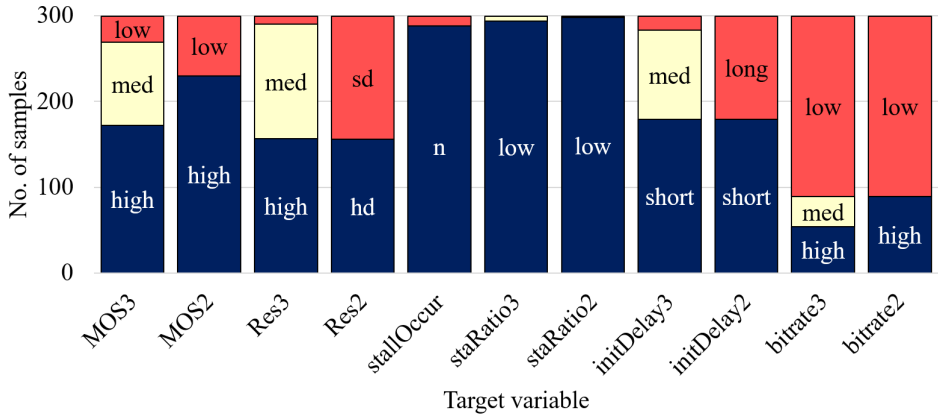


(b) Encoding bitrate distribution for the 100 different videos included in datasets And19L and iOS19L.

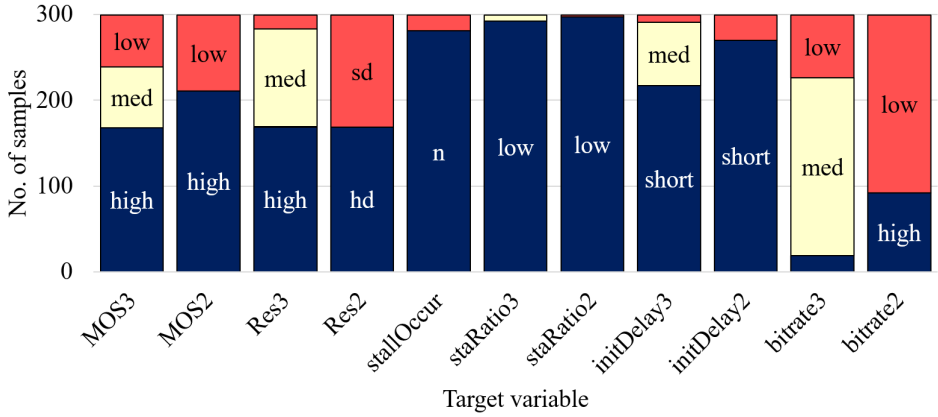
Figure 6.2: Selected application-level characteristics of the datasets And19L and iOS19L.

Finally, we labelled each of the played videos with QoE/KPI classes based on ViQMon application-level performance measurements and MOS values calculated using the previously mentioned publicly available implementation of the QoE model from ITU-T Recomm. P.1203 [97]. To illustrate the range of quality degradations incorporated in the datasets, we provide MOS distributions for both datasets in Figure 6.2a. The ground-truth labelling was done in the same way as in Studies S2 and S3, with the exception being bitrate classified into 2 classes, where we shifted the boundary between the "low" and "high" class to 1000 kbps instead of 1500 kbps, as the distribution of samples across classes in datasets collected in 2019 differed from the one observed in datasets from 2018 (possibly due to more efficient codecs being used). The distribution of samples across classes is given in Figure 6.3.

Both datasets exhibit a low stalling count, 11 in And19L, and 18 in iOS19L. Observed initial delays are generally very short, and, as commented in earlier studies as well, the boundaries between what we consider "short", "medium", and "long" initial delays were set very low for the purpose of testing the feasibility of distinguishing between initial delays of different duration. However, given the circumstances in which the YouTube service adaptation algorithm has obviously been designed to avoid stalling and long initial delays, in this study we focus



(a) Classes distribution in And19L.



(b) Classes distribution in iOS19L.

Figure 6.3: Distribution of video instances into classes. Given that initial delays are generally short, and stalling events were very rare, classes related to these influence factors are rather imbalanced, and thus these classifications are omitted.

on longest played resolution, average video bitrate, and MOS. More specifically, given the distributions of samples in these three types of classifications, we focus on resolution and bitrate classified into 2 classes, and MOS classified into 3 classes. For readibility purposes, we repeat the definition of classes for these three targets:

- MOS: "high" $\geq 4 >$ "medium", $\geq 3 >$ "low",
- Resolution: "hd" $\geq 720p >$ "sd",
- Video bitrate: "high" $\geq 1000\ kbps >$ "low".

6.2.2 Model training and assessment

To measure the extent to which platform indicator and encoding bitrate information provided as additional predictors could potentially improve YouTube QoE classification, we assess 8 model types for each of the 3 QoE/KPI classifications, and for two ML algorithms used for training:

Decision Tree (DT) and Random Forest (RF). The model type depends on the data each model was trained on, as follows:

- M_0: Android and iOS data, no app-level context info,
- M_p: Android and iOS data, with platform indicator,
- A_0: Android only data,
- I_0: iOS only data,
- M_b: Android and iOS data, with encoding bitrate info,
- $M_{p,b}$: Android and iOS data, with platform indicator and encoding bitrate info,
- A_b: Android only data, with encoding bitrate info,
- I_b: iOS only data, with encoding bitrate info.

The symbol “M”, “A”, or “I” indicates the used dataset: merged (And19L + iOS19L), And19L, or iOS19L respectively. The indices indicate the additional context features used, “0” meaning none, “b” indicating encoding bitrates (6 features), and “p” being the platform indicator (1 feature).

Prior to the training of each model type for a target (QoE/KPI), instances were up/subsampled to balance out the instance count in each of the classes (Figure 6.3), and features were subset to reduce computational cost and noise. In the case of MOS, we balance the per-class sample count by upsampling the “low” class using SMOTE [180] and randomly subsampling the “high” class, both in line with the sample count of the “medium” class. For resolution classification we do not perform any up/subsampling, while for bitrate classification we subsample the “low” class. For feature selection, we used the SFS algorithm [181], taking the 10 best features into the model training step. The models were trained using DT and RF algorithms, and tested through 10-fold cross-validation, as well as on a dataset split (67% training, 33% validation), with the results reported in Section 6.3.

With A_0 and I_0 we assess the baseline performance of models trained for a specific platform using the newly collected dataset, while by merging the Android and iOS data and training the M_0 model, we assess the feasibility of using the same model for both platforms. Building on top of that, through M_p, we assess if adding the platform indicator ("Android", "iOS") as a feature into the merged dataset improves performance of the combined model.

Features of models M_b, $M_{p,b}$, A_b, and I_b are enriched with information about bitrates of H.264-encoded videos in resolutions 144p, 240p, 360p, 480p, 720p, and 1080p (6 additional features). This information was added into the process of training separate Android/iOS models (A_b, I_b), and combined models, with ($M_{p,b}$) or without the platform indicator (M_b).

6.3 Results

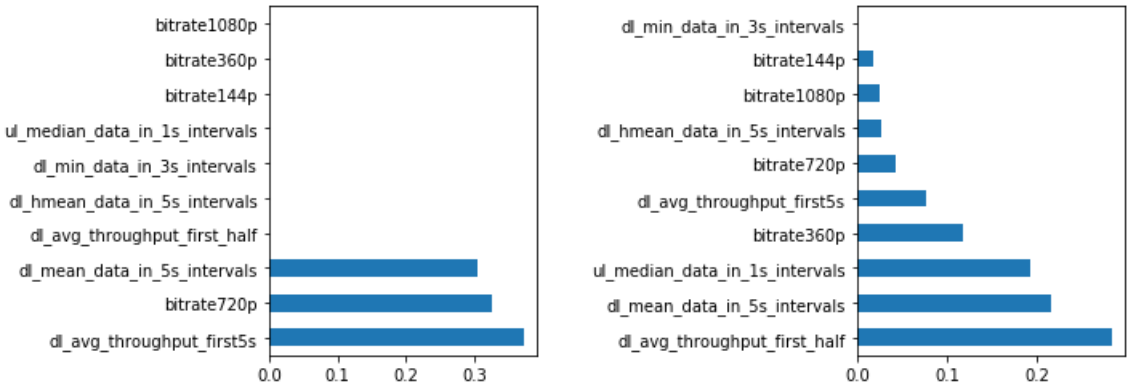
Results in terms of the performance of all 48 models are summarised in Table 6.2 (MOS classification models), Table 6.3 (longest played resolution classification models), and Table 6.4 (average video bitrate classification models). Results show that platform indicator does not have an impact on improving the performance of any of the classification models. Our results show that the platform indicator was not selected by SFS for any of the models. As platform-specific models perform better than the models trained on the merged dataset, it is clear that the platform indicator does carry a certain amount information relevant for the model. However, due to the greedy nature of SFS, it was not recognised. In case the platform indicator is available, it may be utilised by the network provider so as to enable the selection of a proper separate Android or iOS model.

Table 6.2: Performance of models that classify **MOS** into 3 classes. A, I, and M indicate that the dataset used for model training was And19L, iOS19L, and merged (And19L + iOS19L) respectively. Indices denote additional information used as predictors: 0:none, p:platform, b:bitrate.

	DT			**RF**		
Model	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**
M_0	**0.61**	h: 0.57 m: 0.55 l: 0.77	h: 0.61 m: 0.68 l: 0.55	**0.68**	h: 0.81 m: 0.60 l: 0.68	h: 0.61 m: 0.68 l: 0.74
M_p	**0.57**	h: 0.90 m: 0.54 l: 0.51	h: 0.32 m: 0.61 l: 0.74	**0.57**	h: 0.90 m: 0.54 l: 0.51	h: 0.32 m: 0.61 l: 0.74
A_0	**0.70**	h: 0.76 m: 0.66 l: 0.71	h: 0.57 m: 0.66 l: 0.87	**0.73**	h: 0.89 m: 0.69 l: 0.69	h: 0.61 m: 0.66 l: 0.94
I_0	**0.65**	h: 0.67 m: 0.70 l: 0.56	h: 0.83 m: 0.55 l: 0.56	**0.59**	h: 0.62 m: 0.64 l: 0.47	h: 0.75 m: 0.55 l: 0.44
M_b	**0.70**	h: 0.68 m: 0.61 l: 0.84	h: 0.82 m: 0.61 l: 0.68	**0.71**	h: 0.84 m: 0.62 l: 0.70	h: 0.75 m: 0.65 l: 0.74
$M_{p,b}$	**0.73**	h: 0.92 m: 0.66 l: 0.68	h: 0.79 m: 0.74 l: 0.68	**0.70**	h: 0.77 m: 0.63 l: 0.71	h: 0.71 m: 0.61 l: 0.77
A_b	**0.72**	h: 0.74 m: 0.76 l: 0.43	h: 0.78 m: 0.74 l: 0.38	**0.80**	h: 0.76 m: 0.81 l: 1.00	h: 0.81 m: 0.88 l: 0.38
I_b	**0.72**	h: 0.76 m: 0.75 l: 0.64	h: 0.92 m: 0.52 l: 0.78	**0.73**	h: 0.65 m: 0.73 l: 0.93	h: 0.92 m: 0.55 l: 0.78

On the other hand, video coding bitrate information significantly improved the performance of MOS and resolution classification, for all models that include it. Looking into Table 6.2, it is clear that MOS classification models without the bitrate context have the accuracy of roughly

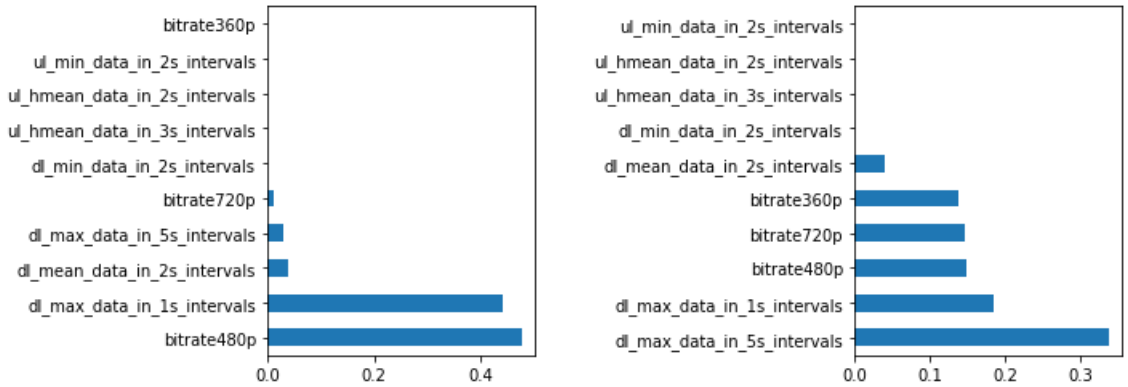
60% (differences may occur due to the stochastic nature of employed algorithms), except for A_0, where the accuracy is higher. When bitrate context is introduced, the accuracy raises above 70% for all MOS classification models. With Figure 6.4, showing feature importances for M_b models, we illustrate that even simple DT–based models improve MOS classification, clearly using at least one of the bitrate features. As opposed to DT, RF–based models employed 9 out of 10 SFS-selected features (and 1000 trees) without significant improvement.



(a) DT model. (b) RF model.

Figure 6.4: Feature importances in M_b MOS classification models enriched with coding bitrate information.

The impact of introducing bitrate context is even larger for resolution classification. The accuracy of models prior to the inclusion of the bitrate context roughly ranges between 75% and 80%, while the introduction of bitrate features raises the accuracy of all models above 90%. Similarly as with MOS, the performance of DT and RF models is comparable, although DT mostly relies on only 2 features, one of which is a bitrate context feature (Figure 6.5).



(a) DT model. (b) RF model.

Figure 6.5: Feature importances in M_b resolution classification models enriched with coding bitrate information.

We illustrate the interpretation of why bitrate context benefits the MOS and resolution classification to such large extent with Figures 6.6, 6.7, and 6.8. The first figure shows all instances

Table 6.3: Performance of models that classify **longest played resolution** into 2 classes. A, I, and M indicate that the dataset used for model training was And19L, iOS19L, and merged (And19L + iOS19L) respectively. Indices denote additional information used as predictors: 0:none, p:platform, b:bitrate.

	DT			RF		
Model	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**
M_0	**0.77**	hd: 0.91 sd: 0.69	hd: 0.64 sd: 0.92	**0.79**	hd: 0.85 sd: 0.74	hd: 0.74 sd: 0.85
M_p	**0.73**	hd: 0.75 sd: 0.71	hd: 0.75 sd: 0.71	**0.75**	hd: 0.89 sd: 0.67	hd: 0.60 sd: 0.91
A_0	**0.82**	hd: 0.84 sd: 0.80	hd: 0.81 sd: 0.83	**0.80**	hd: 0.92 sd: 0.72	hd: 0.67 sd: 0.94
I_0	**0.75**	hd: 0.77 sd: 0.72	hd: 0.81 sd: 0.67	**0.82**	hd: 0.85 sd: 0.77	hd: 0.82 sd: 0.81
M_b	**0.91**	hd: 0.87 sd: 0.96	hd: 0.97 sd: 0.84	**0.91**	hd: 0.87 sd: 0.96	hd: 0.97 sd: 0.84
$M_{p,b}$	**0.91**	hd: 0.87 sd: 0.96	hd: 0.97 sd: 0.84	**0.91**	hd: 0.87 sd: 0.96	hd: 0.97 sd: 0.84
A_b	**0.92**	hd: 0.88 sd: 0.98	hd: 0.98 sd: 0.85	**0.93**	hd: 0.91 sd: 0.95	hd: 0.96 sd: 0.89
I_b	**0.90**	hd: 0.89 sd: 0.92	hd: 0.95 sd: 0.83	**0.89**	hd: 0.87 sd: 0.92	hd: 0.95 sd: 0.81

from both datasets in a 2-dimensional feature space: avg. downlink throughput vs. H.264-encoded video bitrate on resolution 360p (the bitrate feature with the greatest information gain). Videos with lower bitrates (more static content) can have high MOS despite low throughput. As the majority of the network traffic features in our work and related work are throughput-based, it is expected that model performance improves if the coding bitrate information is present, as it provides an indication of the complexity of the content. Due to MOS degradations in our datasets mostly originating from resolution degradations, conclusions are applicable for both. Figure 6.7 confirms these expectations by depicting predictions made by resolution classification M_0 type model (which does not use any additional app-level context info) with respect to bitrate. It can be observed that most of the classification errors occur on low-bitrate videos. A number of these errors are corrected once the bitrate information is present (Figure 6.8).

As for bitrate classification models, none of the context information improved the classification performance in any of the models. In case of platform-specific models, a single coding bitrate feature was selected by SFS, but later not used in the trained model. The explanation may be that average video bitrate highly correlates with throughput, and additional context data cannot improve the performance any further.

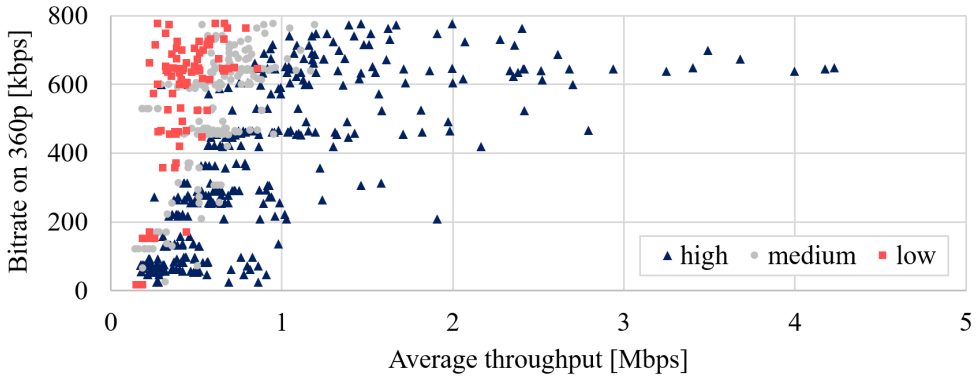


Figure 6.6: Samples in the dataset colored with their ground truth QoE classes. Features such as average downlink throughput may be less informative in cases of low-bitrate content.

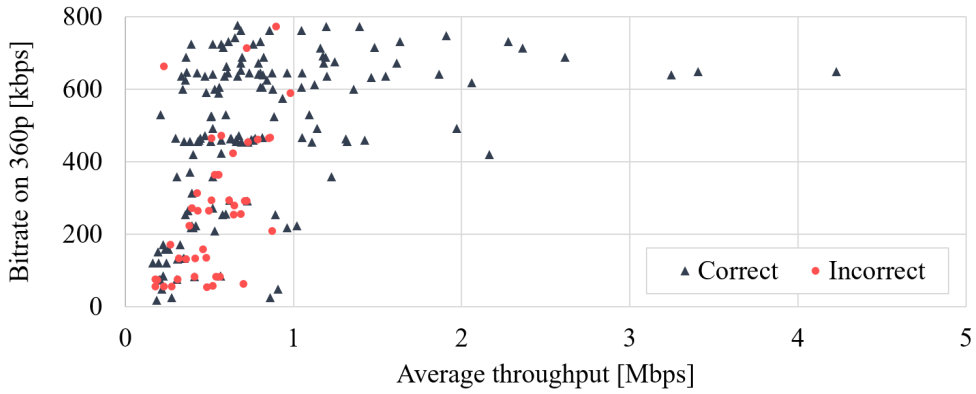


Figure 6.7: Resolution predictions made by M_0 type model in 2-dimensional feature space. Low-bitrate videos are more likely to be misclassified. (Note: bitrate was NOT used when training M_0.)

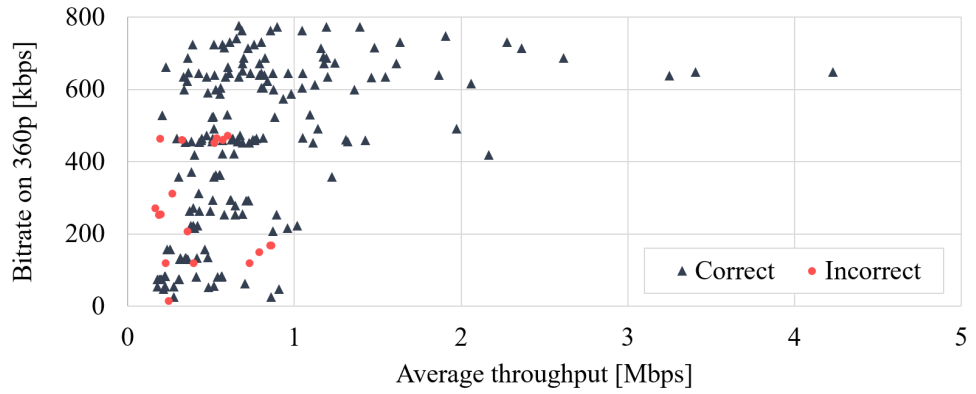


Figure 6.8: Resolution predictions made by M_b type model in 2-dimensional feature space. There are significantly less misclassified low-bitrate instances, as compared to M_0.

6.4 Chapter summary

While there is a series of issues to be resolved to enable QoE-centric information exchange between ISPs and OTTs, primarily related to privacy, lack of business models, and regulations, technical solutions have already been proposed to enable such exchange [65]. We note that in the scope of this thesis we do not investigate the actual technical realisation of information

Table 6.4: Performance of models that classify **bitrate** into 2 classes. A, I, and M indicate that the dataset used for model training was And19L, iOS19L, and merged (And19L + iOS19L) respectively. Indices denote additional information used as predictors: 0:none, p:platform, b:bitrate.

	DT			RF		
Model	**Acc. [%]**	**Prec.**	**Rec.**	**Acc. [%]**	**Prec.**	**Rec.**
M_0	**0.89**	h: 0.88 l: 0.89	h: 0.88 l: 0.89	**0.83**	h: 0.95 l: 0.76	h: 0.67 l: 0.97
M_p	**0.86**	h: 0.82 l: 0.90	h: 0.90 l: 0.82	**0.83**	h: 0.82 l: 0.83	h: 0.82 l: 0.83
A_0	**0.95**	h: 0.92 l: 0.97	h: 0.96 l: 0.94	**0.97**	h: 1.00 l: 0.95	h: 0.92 l: 1.00
I_0	**0.87**	h: 0.81 l: 0.92	h: 0.88 l: 0.86	**0.90**	h: 0.85 l: 0.94	h: 0.92 l: 0.89
M_b	**0.79**	h: 0.78 l: 0.80	h: 0.78 l: 0.80	**0.79**	h: 0.84 l: 0.76	h: 0.70 l: 0.88
$M_{p,b}$	**0.87**	h: 0.85 l: 0.89	h: 0.88 l: 0.86	**0.83**	h: 0.93 l: 0.77	h: 0.68 l: 0.95
A_b	**0.93**	h: 0.92 l: 0.94	h: 0.92 l: 0.94	**0.92**	h: 0.88 l: 0.94	h: 0.92 l: 0.91
I_b	**0.87**	h: 0.77 l: 0.97	h: 0.96 l: 0.80	**0.92**	h: 0.92 l: 0.92	h: 0.88 l: 0.94

exchange, however possible solutions include APIs offered by the OTT provider, or the in-band signalling of non-encrypted select context data.

To motivate resolution of challenges on the way, we demonstrate that, by having access to static information for a played video indicating video encoding bitrates used on different quality levels, the accuracy of in-network QoE/KPI estimation models can be significantly improved. Such information could be provided by OTT providers without infringing upon user privacy, and result in improved QoE. While bitrate classification models perform very highly without any context information, and thus do not make use of it, our results show that MOS classification models increase in accuracy from roughly 60% – 65% to 70% – 75%. As for longest played resolution classification, the accuracy increases from 75% – 80% to 90% – 95%.

Chapter 7

Model generalisation and adaptation

Building on the knowledge gained from the Studies S1 – S5, in this chapter we address a series of practical challenges related to the in-network QoE/KPI estimation problem. First, in Section 7.1, we investigate the potential of training general models able to address QoE/KPI estimation across end user scenarios differing in used device platform. Section 7.2 explores the potential of utilising real-time KPI estimates (as presented in Section 5.4) in session-level QoE/KPI prediction. Finally, as models may need to be updated over time in response to service implementation changes (e.g., changes in adaptation algorithm and buffering strategy, changes in used protocols), Section 7.3 investigates methods for model re-evaluation and adaptation. The chapter is concluded with a summary in Section 7.4.

7.1 Towards general models

Given the vast variety of video streaming services and usage scenarios differing in used client device platform, application used for accessing the content, access network, underlying transport protocol, etc., an important question is whether in-network QoE/KPI estimation for some of these scenarios could be addressed at once. The problem of generalisation can also, in that sense, be viewed from multiple perspectives and levels of abstraction: devising generic methodologies for the training of models, implementing methodology procedures in a generic way, and training models applicable for multiple usage-scenarios. The higher level of detail clearly makes it harder to generalise, as different services may have significantly different behaviour and usage patterns (e.g., YouTube and Netflix), which commonly drives the choice of the adaptation strategy employed by service providers, thus affecting the traffic patterns as well. Certain services and usage-scenarios, however, exhibit similar behaviour and may be addressed with a single QoE/KPI estimation model, potentially with an insignificant drop in model performance.

Throughout this thesis, we have aimed at generalising the overall approach and defining the procedures applied on the case of YouTube in such a way that is applicable to other services as

well. In Section 4.1 we defined a generic framework and all the key steps towards implementing procedures presented in this work. The traffic features defined and refined through Studies S2 – S4 are calculated on IP-level, and thus applicable for both TCP and QUIC traffic. QoE and KPI classes are also applicable to other services as well (beyond YouTube), with a limitation being that ITU-T Recomm. P.1203 model is validated on videos up to 5 minutes long, without user interactions and for certain codecs. The process of training the models is also generic, despite being demonstrated only on the case of YouTube, and can be applied for additional use-cases.

Previously, in Section 5.2 (Study 2) and Section 5.3 (Study 3), we already addressed model applicability across different scenarios. In Study 2, we apply the models trained on data collected in the laboratory setting on data collected in an operational mobile network, achieving promising results. Such a finding is very relevant, having in mind how challenging it may be to collect a sufficiently large dataset in an operational mobile network. Moreover, in Study 3, which focuses on videos streamed using the iOS platform, we test whether models trained using data from Android devices in Study 2 are applicable on iOS data. Here we discover limited generalisation capabilities, with models being applicable with a decrease in performance. The study also considers applying the models to real network data, again yielding promising results.

While the notion of generalisation is obviously interweaved through the thesis, in this section we analyse the case when a model is trained on data collected in multiple use-cases in order for the model to be applicable for those use-cases. This case as such does not ease the process of data collection and processing, but potentially reduces the complexity of a QoE monitoring architecture when deployed in the network. For this purpose, we first establish a baseline by assessing the performance of platform-specific models (for Android and iOS separately) for MOS, resolution and bitrate classification, and then compare results with the performance of models trained on data collected on both platforms.

The datasets used in this evaluation are And19L and iOS19L, described in Section 6.2.1. The performance of the models is also listed in Section 6.3, but not analysed in the context of generalisation. We note that model types A_0 (Android), I_0 (iOS), and M_0 (merged) from Chapter 6 correspond to models in this section. Thus, the methodology with regards to traffic feature extraction, QoE/KPI labelling, feature selection and model training is omitted here, and can be found in Section 6.2.1.

We trained 12 baseline models: using two datasets, two ML algorithms for training purposes (DT and RF), and focusing on MOS, longest played resolution, and video bitrate classification. The results in terms of model performance are presented in Table 7.1 for Android, and in Table 7.2 for iOS. The results show that, both for Android and iOS, average video bitrate is predicted with highest performance. Looking into features that were used in bitrate classification models, it is clear that these models are also very simple, as video bitrate highly correlates with downlink throughput. We demonstrate this with Figure 7.1c, depicting the feature importance in

DT–based bitrate classification for Android. Resolution classification models also heavily rely on downlink throughput features, but result in lower performance, when compared to bitrate classification models. This may be emphasised due to the fact that used datasets include around 20% of static content (such as music with album covers). In these cases, high resolutions do not necessarily result in high bitrates and consequently downlink throughput. Features used in DT–based resolution classification for Android are shown in Figure 7.1b.

Table 7.1: Performance of session-level YouTube QoE/KPI classification models trained and tested using the And19L dataset.

	Algorithm		
	DT	RF	
MOS: high/medium/ low	h:0.76 m:0.66 l:0.71	h:0.89 m:0.69 l:0.69	Prec.
	h:0.57 m:0.66 l:0.87	h:0.61 m:0.66 l:0.94	Rec.
	0.70	**0.73**	Acc.
Longest played resolution: hd/sd	hd:0.84 sd:0.80	hd:0.92 sd:0.72	Prec.
	hd:0.81 sd:0.83	hd:0.67 sd:0.94	Rec.
	0.82	**0.80**	Acc.
Average video bitrate: high/low	h:0.92 l:0.97	h:1.00 l:0.95	Prec.
	h:0.96 l:0.94	h:0.92 l:1.00	Rec.
	0.95	**0.97**	Acc.

MOS classification proves to perform the worst, ranging from 59% to 65%. This is expected, as MOS is already a complex construct influenced by a variety of KPIs that reflect in different ways on the network-level. Features used in RF–based MOS classification for Android are shown in Figure 7.1a. Aiming to improve the performance of MOS classification models, we propose the hybrid approach in Section 7.2, which uses real-time KPI predictions as additional features. However, we stress that the MOS classification models presented in this section may be sufficient in certain applications, given that most misclassifications occur between classes "low" and "medium" or "high" and "medium".

Due to a lack of guidelines available in literature and standards in terms of network conditions in which the data should be collected, what kind of content the dataset should include, etc., data collection methods differ across related work. Moreover, related work mostly focuses on YouTube viewed in the browser, rather the native YouTube app. We thus cannot perform direct comparisons with related work in terms model performance. We are aware that introducing estimated chunk–based features, such as in [139, 140] would potentially improve the performance of the models presented in this section. However, chunk detection requires significantly more processing, and may be impossible in cases when multiple chunks are downloaded in parallel.

Table 7.2: Performance of session-level YouTube QoE/KPI classification models trained and tested using the iOS19L dataset.

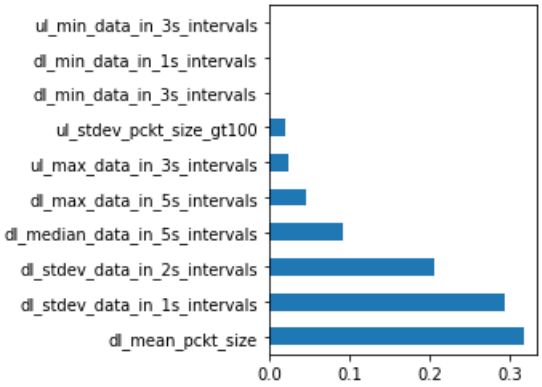
	Algorithm		
	DT	RF	
MOS: high/medium/ low	h:0.67 m:0.70 l:0.56	h:0.62 m:0.64 l:0.47	Prec.
	h:0.83 m:0.55 l:0.56	h:0.75 m:0.55 l:0.44	Rec.
	0.65	**0.59**	Acc.
Longest played resolution: hd/sd	hd:0.77 sd:0.72	hd:0.85 sd:0.77	Prec.
	hd:0.81 sd:0.67	hd:0.82 sd:0.81	Rec.
	0.75	**0.82**	Acc.
Average video bitrate: high/low	h:0.81 l:0.92	h:0.85 l:0.94	Prec.
	h:0.88 l:0.86	h:0.92 l:0.89	Rec.
	0.87	**0.90**	Acc.

This is why we only focus on network-layer features to keep our models simple and robust.

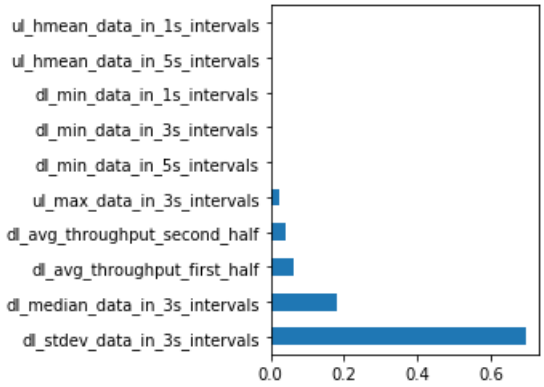
Following the establishment of a baseline we train 6 models – for classifying MOS, resolution and bitrate using DT and RF – on the merged dataset that includes both Android and iOS data (And19L and iOS19L). We follow the same procedure, as described in Section 6.2.1, including splitting into train and test set, subsetting the merged dataset to balance out the number of samples in line with the least populated class and SFS–based feature selection. Trained models' performance is summarised in Table 7.3.

Table 7.3: Performance of session-level YouTube QoE/KPI classification models for Android and iOS (trained and tested using the merged And19 and iOS19 dataset).

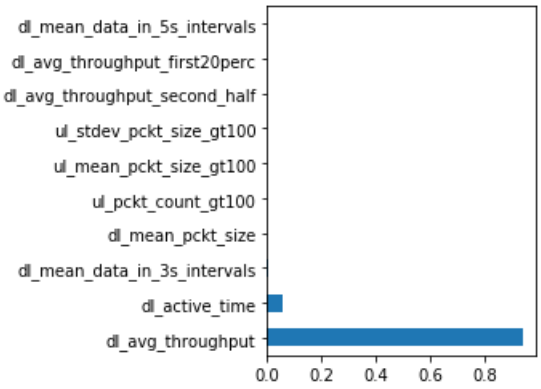
	Algorithm		
	DT	RF	
MOS: high/medium/ low	h:0.57 m:0.55 l:0.77	h:0.81 m:0.60 l:0.68	Prec.
	h:0.61 m:0.68 l:0.55	h:0.61 m:0.68 l:0.74	Rec.
	0.61	**0.68**	Acc.
Longest played resolution: hd/sd	hd:0.91 sd:0.69	hd:0.85 sd:0.74	Prec.
	hd:0.64 sd:0.92	hd:0.74 sd:0.85	Rec.
	0.77	**0.79**	Acc.
Average video bitrate: high/low	h:0.88 l:0.89	h:0.95 l:0.76	Prec.
	h:0.88 l:0.89	h:0.67 l:0.97	Rec.
	0.89	**0.83**	Acc.



(a) MOS, 3 classes, RF model.



(b) Resolution, 2 classes, DT model.



(c) Bitrate, 2 classes, DT model.

Figure 7.1: Feature importance in session-level QoE/KPI classification models trained for the Android platform.

The results are evidently comparable with the results achieved with separate models. This means that a single model can potentially address multiple use-cases and there is no need to first recognise the platform once the model is deployed. This does not necessarily mean that the model itself can be trained on a single-platform dataset and applied to another platform, as has been investigated in Study 3 [17], due to service implementation differences on the two platforms [18]. Similar conclusions, but focused on different services and not platforms, have been found in [139, 208]. The authors show that developing well–performing general models is feasible if the training set included data from all services. Applying the model trained on Amazon, YouTube, and Twitch data to Netflix data resulted in a significant drop in model performance. In Study 3 [17] we found that the drop for different platforms is not that significant, but still existent.

7.2 Utilisation of real-time KPI estimates as additional predictors in session-level QoE classification

With the aim of improving the performance of MOS classification models, we enrich them with predictions made by real-time resolution and bitrate classification models. Once the per-second predictions for the session are available, they are aggregated into two additional features (on top of the ones defined in Table 6.1) – percHD and percHBr – percentage of intervals with high definition and percentage of intervals with high bitrate. As in previous section, we balance out the dataset, split it into train and validation set (67% : 33%), select relevant features, and train DT and RF models. The approach is demonstrated on a platform-specific model for Android, with the results given in Table 7.4.

Table 7.4: Performance of session-level YouTube QoE classification models enriched with real-time predictions as additional features. Models were trained and tested using the And19L dataset.

	Algorithm		
	DT	RF	
MOS: high/medium/ low	h:0.93 m:0.89 l:0.73	h:1.00 m:0.85 l:0.72	Prec.
	h:0.93 m:0.66 l:0.97	h:0.89 m:0.74 l:0.90	Rec.
	0.84	**0.84**	Acc.

The performance of MOS classification models was significantly improved, while the complexity of the model is reduced. We illustrate this statement with Figure 7.2 that shows feature importance in the DT–based MOS classification model. The percentage of intervals with HD resolution was confirmed as the most important feature, followed by one of the commonly used throughput–based features. PercHBr was not considered relevant by the SFS algorithm. The reason for that may be that bitrate highly correlates with certain throughput–based features, and introducing percHBr carries no additional value to the model. On the other hand, as throughput–based features can be misleading for videos with static content, percHD carries additional value in identifying such cases.

7.3 Model re-evaluation and adaptation

Streaming services can occasionally apply updates of the adaptation and buffering logic and thus change the typical network traffic patterns that QoE/KPI estimation models are dependent on. While the changes can be observed on an application-level as for example changes in the amount of data that is preloaded (buffered), changed segment size, more or less frequent quality switches, etc., the question arises as to how such changes can be detected in order to apply an appropriate update to QoE/KPI estimation models. In this section we test to what extent do

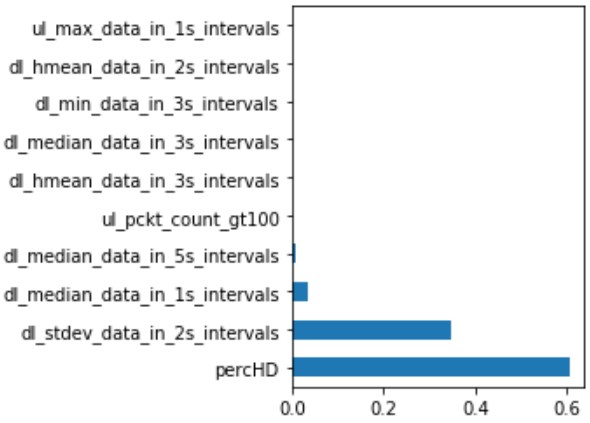
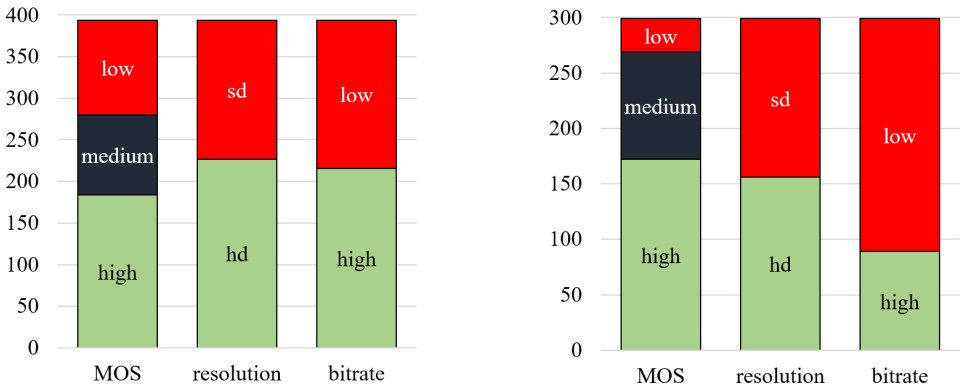


Figure 7.2: Feature importance in the hybrid MOS classification model.

such changes reflect on the performance of QoE/KPI classification models (Section 7.3.1), and how such changes can be addressed to mitigate model under-performance (Sections 7.3.2 and 7.3.3).

7.3.1 Testing models trained with old data on new data

We train the models on the dataset And18L, collected on an Android device in early 2018 and test it on the dataset collected on Android in early 2019 (And19L). Figure 7.3a depicts the number of instances in each of the classes in dataset And18L. The same is displayed for dataset And19L in Figure 7.3b. Prior to model training, we randomly subsample the And18L dataset to balance out the number of instances in accordance with the least populated class. We note that the dataset And19L was not subsampled prior to model testing. Relevant features were selected using the SFS algorithm, and models trained using DT and RF. Models' performance is reported in Table 7.5.



(a) Dataset And18L.

(b) Dataset And19L.

Figure 7.3: Distribution of instances across session-level QoE/KPI estimation classes.

In comparison to models trained and tested on splits originating from the same dataset (models trained on And18L in study S2, and models trained on And19L in this study), the results

Table 7.5: Performance of session-level YouTube QoE/KPI classification models trained on Android dataset from 2018 (And18L) and tested on a newer Android dataset from 2019 (And19L).

	Algorithm		
	DT	RF	
MOS: high/medium/ low	h:0.99 m:0.57 l:0.27	h:0.95 m:0.53 l:0.20	Prec.
	h:0.48 m:0.65 l:0.93	h:0.48 m:0.35 l:1.00	Rec.
	0.58	**0.49**	Acc.
Longest played resolution: hd/sd	hd:0.85 sd:0.68	hd:0.60 sd:0.95	Prec.
	hd:0.62 sd:0.88	hd:0.60 sd:0.95	Rec.
	0.74	**0.77**	Acc.
Average video bitrate: high/low	h:0.64 l:0.99	h:0.70 l:0.96	Prec.
	h:0.98 l:0.77	h:0.92 l:0.83	Rec.
	0.83	**0.86**	Acc.

show a slight decrease in performance of resolution– and bitrate–classification models, but significant decrease in MOS–classification performance. We explain this as follows. The KPIs we aim to predict, longest played resolution and average video bitrate, are averaged across the whole session and do not carry temporal information. Thus, a lot of information regarding the adaptation strategy is lost. On the other hand, the ITU-T Recomm. P.1203 model, based upon which MOS is calculated, takes into account all the QoE-influencing data on a per-second level, which is where the adaptation strategy changes come into play. Based on our observations, YouTube's buffering strategy has not changed in terms of observed low and high thresholds. However, other changes in video delivery mechanisms, such as the frequency and amplitude of quality switches, may influence the traffic patterns the models rely upon. Thus, there is a need to investigate potential solutions for model re-evaluation and adaptation, at the same time aiming at minimising the required data collection effort.

7.3.2 Adaptive learning strategies

In real-life applications, data used as input for prediction models often changes over time, making the performance of the models degrade as new data is presented to them. In general, this phenomenon is known as the *dataset shift* or *dataset drift* [209, 210]. For example, a model predicting the sales of a product trained on data collected during November may not be applicable during Christmas time. Also, the same model may, for example, not be applicable in case of a disruption caused by a similar product appearing on the market. Dataset shift occurs not only with changes in data over time, but also depending on the specifics of the dataset used for model training, when compared to the test set. For example, if a certain model was trained on

data containing information about people of younger age, that model may not perform well on data containing information about people of older age [211].

Dataset shift can be divided into three categories: 1) *covariate shift*, 2) *prior probability shift*, and 3) *concept drift* [212, 213]. The example with datasets containing information about people from different age groups is an example of a covariate shift – there is a shift in independent variables, i.e., the distribution of age differs between the two datasets. The case of the sales prediction is the example of a prior probability shift – the shift in the target variable. When the cause of the change is hidden, i.e., not explicitly given in the form of a predictive feature or a target variable, we are dealing with a more complex type of a dataset shift known as the *concept drift*. In concept drift, the target concept rather depends on hidden contexts that are not explicitly provided to the learner algorithm.

Depending on the domain, concept drift can appear suddenly (sudden, abrupt, instantaneous concept drift), gradually (gradual concept drift), and temporarily disappear and reappear (reoccurring concept drift) [214]. In [215] sudden drift is explained with an example of someone graduating from college and suddenly having completely different monetary concerns. Gradual drift, on the other hand, is described with an example of a slowly wearing piece of factory equipment causing a gradual change in the quality of output parts. When concepts reappear in different versions, we are dealing with the reoccurring concept drift. A good example for that is seasonality, e.g., an increase in sales of ice-cream in summer [216].

The strategies of handling the drift differ among these types of concept drift. For sudden drifts, the most common approach is to re-train the model on a time-window of instances. The windows can be fixed or variable. For example, if 100 new instances labelled with ground truth are collected per week, and the model is re-trained on a weekly basis using the newest 300 samples (i.e., samples from last three weeks) – this is referred to as a *fixed window strategy*. If a sudden change occurred at some point in those three weeks, the model will include data from before and after the occurrence of the change. In case a more precise model is needed, it can be re-trained more often, such as daily. In the *variable window strategy*, the change first needs to be detected, then old data is dropped, and the model trained on new data only. The change is detected as the under-performance of the model on new data, with common strategies discussed in [209, 217]. However, in that case a more significant amount of data needs to be collected after the drift is detected.

A common approach for handling gradual drift is employing a dynamic ensemble in which different models trained during the period of change vote for target predictions of instances in the test set. In case of reoccurring drift, a contextual approach can be used, such as selecting an appropriate model depending on the season [214]. These approaches are not further discussed, as they are not relevant in the context of video streaming application changes, which can be characterised as sudden drifts, and possibly handled with windows.

7.3.3 Testing models trained with enriched old data on new data

Focusing on sudden concept drifts and on window-based approaches to handling the drift, the first step is deciding whether fixed or variable windows would be more appropriate for the specific application. Aiming at keeping the solution as simple as possible, fixed windows, if applicable, seem as a more appropriate choice, as there is no need for change detection or consequent larger dataset collection. Video streaming data can be collected using test devices periodically, e.g., 10 videos per week, and models re-trained on a weekly basis using the latest 300 samples. In the absence of continuously collected datasets, we refrain from making any conclusions on the schedule that should be employed – the schedule itself would need to be adjusted based on experience.

We illustrate the process of model re-evaluation and adaptation with Figure 7.4. We envision the solution as a hybrid between window approaches defined in previous section: while the focus is on fixed windows, in the solution, we assume that the model can be tested on a small sample of new instances prior to replacing the oldest instances with new ones. Starting from (1) the "Model training" step in which the initial model is trained on n instances (n is the size of the window), the solution proposes the use of either (2) periodic updating – automated collection of a small batch of new instances that replace the same number of oldest instances in the window, or (3) model testing – automated collection of a small batch of new instances and testing of the existing model, where the next step will depend on testing results. In case that (2) periodic updating is employed, the instances that are in the window after the update are taken back into the model training step. If (3) model testing is employed, thresholds can be defined which would lead to either (4) an enrichment of the existing set of instances with new ones or (5) initiation of a new measurement campaign. These processes can be performed simultaneously.

For example, a network operator may decide to train a session-level QoE estimation model on 400 video streaming instances. The data collection instrumentation is set to collect 40 samples per week and, referring back to the conceptual framework (Figure 4.1), the model training component is set to train the model weekly on 400 newest samples. While deployed model is not tested weekly on each new 40 samples, the operator might want to test the model performance on a monthly basis, possibly on a slightly larger number of samples. For the sake of the example, the data collection instrumentation automatically, independently of the periodic updating, collects 80 samples once a month to test the currently deployed model. If the model performs better than is defined with the "enrichment" threshold, the 80 samples used for testing can replace the oldest 80 samples and the updated window of features is used for model training (the case of $m = 0$ in step (4)). If the model performs worse than the "enrichment" threshold, but better than the "new campaign" threshold, additional samples can be collected and model trained on the newest 400 samples that include the 80 used for testing and the samples collected after the test. In the worst case scenario, the model performs worse than the "new campaign"

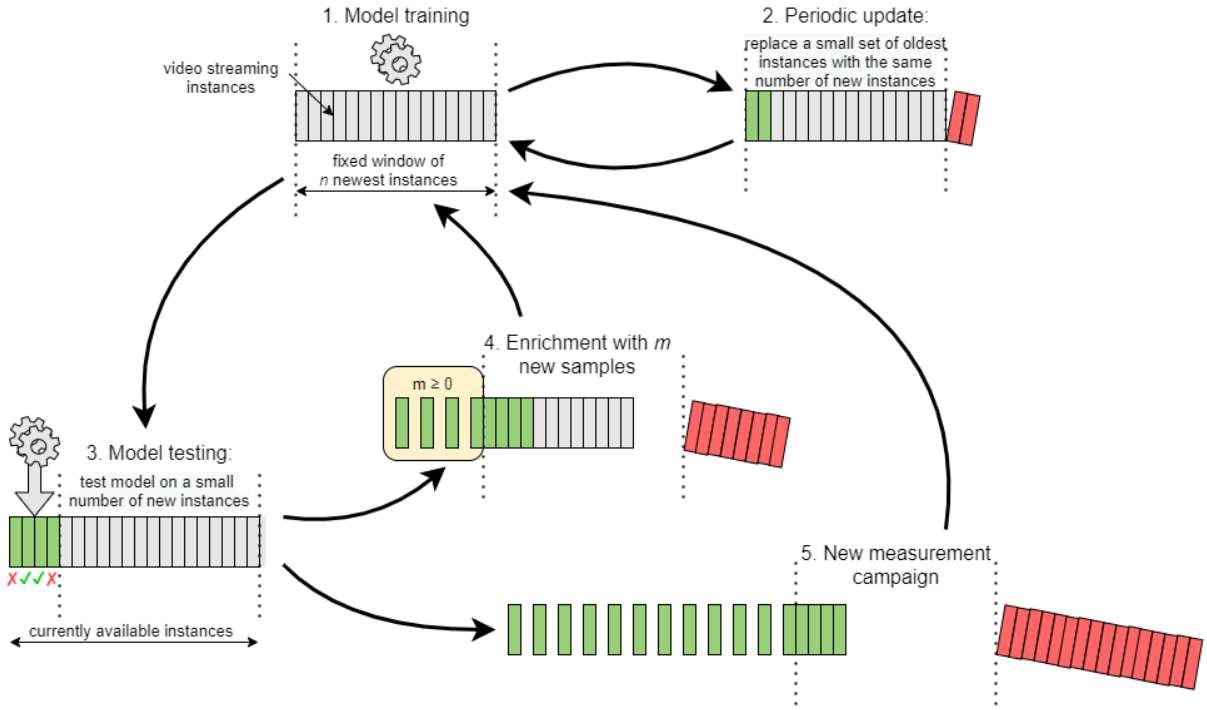


Figure 7.4: Proposed approach for automated model re-evaluation and adaptation.

threshold, and a collection of new 400 samples is initiated. We note that the used numbers are just used to illustrate the proposed solution. Data collected over longer periods of time is needed to determine the configuration (in terms of periodicity and thresholds), which may be fine-tuned based on experience.

To see to what extent can different amounts of new data contribute to the model performance, we run a simple test. Given the lack of datasets collected consistently and periodically, we base the test on And18L and And19L datasets. For each KPI, the And18L dataset was subsampled with respect to the class with the smallest number of samples (Figure 7.3), and then a number of samples from And19L was added, forming the train set. The rest of the samples from And19L was used for testing. The exact number of newer samples enriching the train test varied among different KPI classifications. For example, as there are only 30 "low" MOS samples in And19L, the only test performed in case of MOS is when 15 samples from each MOS class are added to And18L. For resolution and bitrate classification, we test the model performance when the old dataset is enriched with 15, 30, and 50 samples. We note that this evaluation is not used to demonstrate the fixed window approach (as the number of instances is not fixed in the test), rather to test whether model performance can be improved with a small number of new samples. After adding the new samples, we used SFS to select the relevant features and DT and RF for model training. The performance of all trained models is summarised in Table 7.6.

Results show that for resolution and bitrate classification the performance of the models increases as we increase the number of new samples included in the training set. In comparison to results obtained when testing And18L–trained models on And19L without including new data

Table 7.6: Performance of session-level YouTube QoE/KPI classification models trained on Android dataset from 2018 (And18L) enriched with samples from a newer Android dataset from 2019 (And19L), tested on the rest of the data from And19L.

	Algorithm		
	DT	RF	
MOS: high/medium/low **15** new samples per class	h:0.80 m:0.48 l:0.17	h:0.99 m:0.54 l:0.15	Prec.
	h:0.63 m:0.49 l:0.53	h:0.50 m:0.55 l:0.93	Rec.
	0.58	**0.54**	Acc.
Longest played **resolution**: hd/sd **15** new samples per class	hd:0.89 sd:0.71	hd:0.96 sd:0.69	Prec.
	hd:0.67 sd:0.91	hd:0.60 sd:0.97	Rec.
	0.78	**0.78**	Acc.
Longest played **resolution**: hd/sd **30** new samples per class	hd:0.78 sd:0.77	hd:0.91 sd:0.72	Prec.
	hd:0.80 sd:0.75	hd:0.68 sd:0.93	Rec.
	0.78	**0.80**	Acc.
Longest played **resolution**: hd/sd **50** new samples per class	hd:0.89 sd:0.78	hd:0.91 sd:0.71	Prec.
	hd:0.78 sd:0.89	hd:0.67 sd:0.92	Rec.
	0.83	**0.79**	Acc.
Avg. video **bitrate**: high/low **15** new samples per class	h:0.60 l:0.97	h:0.59 l:0.96	Prec.
	h:0.95 l:0.76	h:0.91 l:0.67	Rec.
	0.81	**0.80**	Acc.
Avg. video **bitrate**: high/low **30** new samples per class	h:0.64 l:0.97	h:0.64 l:0.97	Prec.
	h:0.92 l:0.83	h:0.92 l:0.83	Rec.
	0.85	**0.85**	Acc.
Avg. video **bitrate**: high/low **50** new samples per class	h:0.61 l:0.99	h:0.64 l:0.99	Prec.
	h:0.95 l:0.85	h:0.95 l:0.87	Rec.
	0.87	**0.88**	Acc.

into the training set (Table 7.5), the model performance slightly increases even for a very small number of newly added instances. Comparing the results to the ones obtained on a single dataset (And18L: Table 5.8, And19L: Table 7.1), it can be seen that with 50 added new samples, the resolution classification performance is in line, which means that the model already performs as well as if it was fully trained on new data. Bitrate classification performance does not reach the performance of models trained and tested on the same dataset in these tests, but new samples clearly do make the model more appropriate for usage on new data. With a small number of "low" MOS in And19L, the same cannot be concluded from exact numbers, but as MOS

depends on aforementioned KPIs, we can assume that the same increase in model performance would be observed with the increased number of new samples in training.

7.4 Chapter summary

In this chapter we addressed several challenges that may be of interest when considering the deployment of QoE/KPI estimation models. Aiming at reducing the complexity of QoE monitoring, the chapter examined the notion of general models, able to address multiple video streaming usage scenarios. The chapter provided references to previous sections of the thesis where the idea is initially evaluated, and also presented results achieved with models trained on data from multiple use-cases (and thus trained to be general). Moreover, the chapter proposed the idea of combining different outputs of QoE/KPI monitoring deployments in order to improve overall model performance, and presents results achieved with models that use real-time predictions as additional predictors for session-level QoE/KPI estimation. Finally, the chapter proposed ideas for maintaining the deployed models in light of potential changes that may occur in video service adaptation/buffering behaviour or used protocols. Various adaptive learning techniques are discussed. Considering the nature of the problem, we propose a solution for automated re-evaluation and adaptation based on updating a fixed window of video streaming instances. In a practical test, we show that even a small number of newly collected instances added to the existing dataset can improve model performance.

Chapter 8

Conclusions and future work

8.1 Summary of contributions

Due to the widespread use of encryption in OTT traffic, network operators, for the most part, lack insight into video streaming performance at the client. With the increasing amounts of video traffic passing through global networks, there is a lot of interest from network operators for finding solutions to estimate streaming performance in terms of both application-level KPIs and QoE. Such insights are further needed in order to detect QoE degradations and mitigate underlying issues (if these are rooted in network), and efficiently manage the traffic with respect to limited resource availability, while keeping the customers satisfied.

While there is a considerable amount of research relying on ML for various traffic classification problems, at the beginning of this research there were very few papers addressing the problem of QoE/KPI classification of video streaming sessions with ML–based traffic analysis. The Prometheus approach [10] was one of the first to rely on ML for this problem. However, as compared to our initial work (Study S1) [12, 13], it employs application-level features, and is thus not applicable in the context of encryption. Roughly at the same time as our initial study was published, Dimopoulos *et al.* published a paper on YouTube KPI classification [11]. While their approach is applicable for encrypted traffic, the model training methodology relies on non-encrypted traffic collected at a Web proxy. To the best of our knowledge, our feasibility study [12, 13] was the first to present an approach fully applicable for encrypted traffic. The resulting publications provide an in-depth description of the methodology, thus also encouraging reproducibility.

The introduction of the QUIC protocol motivated the adaptation of the devised methodology, and so did the realisation that APIs used in developed YouTube performance measurement apps did not switch to employing QUIC nor do they behave in the same way as the official YouTube app. This was analysed in Study S2, published in [14, 15], where we presented our advancements with regards to measurement methodology and new QoE/KPI estimation models,

applicable for QUIC, while also employing the newly published ITU-T P.1203 [97] QoE model. The findings with regards to the measurement apps were confirmed in [124], where the authors present a similar alternative approach for performing client-side measurements. However, while a lot of papers published after our Study S2 employ IP-level features for (mostly real-time) KPI estimation for YouTube viewed in the browser [126, 127, 128, 129, 139, 140, 141], ours remains a rare study focused on the official YouTube Android app and, besides KPIs, on estimating QoE based on a standardised model.

With the lack of studies on QoE/KPI estimation focused on iOS, we addressed this gap with Study S3 [17]. Besides the methodology refinement and trained models, the novelty of the study is also in first attempts at cross-testing the QoE/KPI estimation models against datasets collected in different use-cases (platforms and access networks).

As service providers are generally not keen on sharing service performance data with network operators, although there are incentives for cooperation for both sides in terms of revenue generation [83] and technical solutions [65], we aimed to further motivate the cooperation and resolution of regulatory issues from a data perspective. The unique contribution of the Study S4 [18] is in the investigation of potential parameters that can be shared by OTTs, and the evaluation of model performance, given the availability of such parameters.

Keeping track with the emerging research on real-time KPI estimation [126, 127, 128, 129, 139, 140, 141], in Study S5 [16], we develop our own methodology focused on the use-case concerning the official YouTube Android app, as opposed to all the aforementioned studies, which rely on data collection in the browser (most of them on laptops).

Finally, in Study S6 [16], we address additional problems related to QoE/KPI estimation. The problem of applicability of models across multiple use-cases has, to some extent, been addressed in [139], where the authors perform cross-tests and train generic models for multiple services. Our study contributes to the body of knowledge by assessing the performance of models applied across multiple platforms and access networks. The study further addressed the potential of combining the outputs of real-time and session-level QoE/KPI estimation, and explored and evaluated solutions for model re-evaluation and adaptation, which, to the best of our knowledge, has not been addressed in related work thus far.

The six studies described in this thesis have yielded a large number of models, targeted at classifying QoE and KPIs such as average bitrate and longest played resolution, addressing both Android and iOS platforms, data collected in the lab and in an operational mobile network, and on different time scales – both on a session-level and in real-time. With each study, the overall methodology has been refined or enhanced with new elements, to finally take the shape presented in Chapter 4, depicted with a conceptual framework. The framework and the methodology it represents are generic, although its components are demonstrated in practice on the use-cases concerning YouTube. We emphasise that, to date, related work has been focused

on addressing individual use-cases related to the QoE/KPI estimation problem, and there is no similar framework proposed in literature, that integrates all key steps towards the deployment of QoE/KPI monitoring architectures in the network.

8.2 Challenges and future work

In-network QoE/KPI estimation is a very active research area, with our work also adding to the body of knowledge. However, the conducted research has certain limitations and there are open research questions that would need to be addressed to alleviate these limitations. In the following text, we highlight a number of open research issues with respect to the limitations of our studies.

Automate the application-level performance measurements. Most of our datasets collected in the lab (Table 4.2) contain approximately 300 – 400 videos. Having a setup for conducting measurements with higher level of automation would enable running large-scale data collection campaigns and consequently the training of more robust models. So far, the available literature has proposed the usage of automation frameworks such as Selenium for browser-based measurements [126, 128, 129, 141]. However, such automated client-side measurement tools can also be instrumented on mobile devices using frameworks as Appium or Selendroid.

Conduct further longitudinal studies on model re-evaluation and adaptation. While the thesis proposes a solution for automated model re-evaluation and adaptation based on active learning, and evaluates its applicability on a simplified example involving two datasets collected in 2018 and 2019, there is a lack of datasets collected regularly over a longer period of time. A dataset collected in periodical small batches would enable a more comprehensive assessment of the proposed automated process.

Determine the dataset size and characteristics needed for the training of robust models. In the studies presented in this thesis, we assume that the datasets are large enough, based on the model performance on a dataset split. We also assume that, by imposing a variety of bandwidth limitations, the datasets are varied enough to be applicable on data collected in conditions that were not explicitly set during the training data collection. However, there is a lack of guidelines with respect to dataset size and characteristics. To bridge this gap, a comprehensive study is needed, assessing the performance of models trained on datasets of different sizes. A valuable input for such a study would be in a comprehensive set of measurements of real network conditions, which would enable realistic shaping, and thus the training of models that would potentially perform better in the real-network setting. A set of guidelines on running dataset collection campaigns for the purpose of QoE/KPI estimation model training would also enable researchers to compare the results of their studies in terms of model performance.

Include user interactions. While the models presented in this thesis show promising per-

formance, Bartolec *et al.* in [142, 143] show that their performance is lowered once tested on datasets with user interactions. Pending the availability of QoE models that take playback-related interactions into account, future work can include user interactions in the training of KPI estimation models, especially given the automation of the data collection procedure. What is currently unclear is what are the interactions most commonly performed by real users of video streaming services, and how and when during playback they should be introduced in the measurement procedure.

Address different types of services. The practical studies presented throughout this thesis demonstrate the generic methodology, but were focused exclusively on YouTube VoD streaming. An interesting avenue for future research is QoE/KPI monitoring for live streaming, which is lately gaining popularity. Live streaming has higher demands towards the network, as the buffer is typically much smaller, thus making QoE degradations more likely to occur. Moreover, users' behaviour when using such services is different, with typically longer sessions and the option to communicate with the streamer and viewers. Another interesting use-case is 360-degree video, streamed to either smartphones or VR headsets, which is typically more bandwidth-intensive, especially if high resolutions are required (such as for VR headsets streaming videos in 2K and more).

Train and evaluate models that work on fixed intervals. While most of the models in this thesis work on a session-level, these models require the corresponding traffic to be detected (i.e., delimitated). Session delimitation is a challenge on its own, and certainly adds up to the complexity of solutions deployed in the network. An alternative approach is to train the models on fixed intervals of video traffic (not in real-time, but rather closer to session-level, e.g., per-minute). If such models perform well, it may be reasonable to choose such models over models that require an additional step, thus accumulating the errors originating from multiple estimations.

Investigate potential deployments of QoE/KPI monitoring architectures. There are also numerous challenges related to the deployment of devised models, especially in the context of 5G networks and 5G-specific network functions that support network data analytics [144]. One of the problems related to the actual deployment is also the computational complexity. While in this thesis we scratch the surface of this question by addressing the trade-off between model complexity and accuracy, it remains unclear how much resources these models would require if deployed in the network, and what portion of the traffic in the network should and could be analysed. An interesting avenue for future research is also exploration of the potential for utilising P4 and deploying QoE estimations directly in the data plane [163, 164, 165].

Further study the applicability of QoE/KPI estimation models across multiple use-cases. Undoubtedly, a big challenge for in-network QoE/KPI estimation lies in the dimensionality of the problem. With regards to the services with different streaming adaptation logic,

related work [139] has shown that it is possible to train well-performing generic models that infer QoE/KPIs for multiple services. However, this has proven to be the case only if the model was trained on data from all services. The question remains whether different techniques, for example those from the domain of transfer learning and semi-supervised learning, could make training data collection less exhaustive. Such methods could be evaluated in the context of multiple services, platforms, and networks. While we have, to some extent, addressed cross-platform model applicability, and tested the models on new data collected in a mobile network, we are currently taking part in a study aimed at quantifying the effects of different networks on the performance of the models. The study uses the same setup, with same bandwidth limitations, and same videos being played in a WiFi network at different locations in Europe, in order to test whether models trained at one location would be applicable on other, and, if not, whether transfer learning could be exploited to make them applicable.

Bibliography

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2017-2022", Cisco, White paper, 2019, Available at: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.pdf (Accessed: 28/08/2020).

[2] Skorin-Kapov, L., Varela, M., Hoßfeld, T., Chen, K.-T., "A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Vol. 14, No. 2s, 2018, pp. 1–29.

[3] Sandvine, "The Mobile Internet Phenomena Report", Sandvine, Tech. Rep., 2020, Available at: https://www.sandvine.com/download-report-mobile-internet-phenomena-report-2020-sandvine (Accessed: 28/08/2020).

[4] Ericsson, "Ericsson Mobility Report", Ericsson, Tech. Rep., 2019, Available at: https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf (Accessed: 28/08/2020).

[5] "ISO/IEC 23009-1:2019; Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats", International Organization for Standardization, Standard, 2019, Available at: https://www.iso.org/standard/79329.html (Accessed: 28/08/2020).

[6] Pantos, R., May, W., "HTTP Live Streaming", IETF, RFC 8216, 2017, Available at: https://datatracker.ietf.org/doc/rfc8216/ (Accessed: 28/08/2020).

[7] Google, "HTTPS Encryption on the Web", Google, Tech. Rep., 2020, Available at: https://transparencyreport.google.com/https/overview (Accessed: 28/08/2020).

[8] Iyengar, J., Thomson, M., "QUIC: A UDP-Based Multiplexed and Secure Transport", IETF, Internet Draft, 2020, Available at: https://datatracker.ietf.org/doc/draft-ietf-quic-transport/ (Accessed: 28/08/2020).

[9] Bishop, M., “Hypertext Transfer Protocol Version 3 (HTTP/3)”, IETF, Internet Draft, 2020, Available at: https://datatracker.ietf.org/doc/draft-ietf-quic-http/ (Accessed: 28/08/2020).

[10] Aggarwal, V., Halepovic, E., Pang, J., Venkataraman, S., Yan, H., “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. ACM, 2014, p. 18.

[11] Dimopoulos, G., Leontiadis, I., Barlet-Ros, P., Papagiannaki, K., “Measuring Video QoE from Encrypted traffic”, in Proceedings of the 2016 Internet Measurement Conference, 2016, pp. 513–526.

[12] Orsolic, I., Pevec, D., Suznjevic, M., Skorin-Kapov, L., “YouTube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning”, in 2016 IEEE Globecom Workshops (GC Wkshps). IEEE, 2016, pp. 1–6.

[13] Orsolic, I., Pevec, D., Suznjevic, M., Skorin-Kapov, L., “A Machine Learning Approach to Classifying YouTube QoE Based on Encrypted Network Traffic”, Multimedia Tools and Applications, Vol. 76, No. 21, 2017, pp. 22 267–22 301.

[14] Orsolic, I., Suznjevic, M., Skorin-Kapov, L., “YouTube QoE Estimation from Encrypted Traffic: Comparison of Test Methodologies and Machine Learning Based Models”, in 2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2018, pp. 1–6.

[15] Orsolic, I., Skorin-Kapov, L., Suznjevic, M., “Towards a Framework for Classifying YouTube QoE Based on Monitoring of Encrypted Traffic”, in International Young Researcher Summit on Quality of Experience in Emerging Multimedia Services (QEEMS 2017), 2017.

[16] Orsolic, I., Skorin-Kapov, L., “A Framework for In-Network QoE Monitoring of Encrypted Video Streaming”, IEEE Access, Vol. 8, 2020, pp. 74 691–74 706.

[17] Orsolic, I., Rebernjak, P., Suznjevic, M., Skorin-Kapov, L., “In-Network QoE and KPI Monitoring of Mobile YouTube Traffic: Insights for Encrypted iOS Flows”, in 2018 14th International Conference on Network and Service Management (CNSM). IEEE, 2018, pp. 233–239.

[18] Orsolic, I., Skorin-Kapov, L., Hoßfeld, T., “To Share or Not to Share? How Exploitation of Context Data Can Improve In-Network QoE Monitoring of Encrypted YouTube

Streams", in 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2019, pp. 1–3.

[19] Licciardello, M., Grüner, M., Singla, A., "Understanding Video Streaming Algorithms in the Wild", in International Conference on Passive and Active Network Measurement. Springer, 2020, pp. 298–313.

[20] Gill, P., Arlitt, M., Li, Z., Mahanti, A., "YouTube Traffic Characterization: a View From the Edge", in Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, 2007, pp. 15–28.

[21] Zink, M., Suh, K., Gu, Y., Kurose, J., "Characteristics of YouTube Network Traffic at a Campus Network–Measurements, Models, and Implications", Computer networks, Vol. 53, No. 4, 2009, pp. 501–514.

[22] Adhikari, V. K., Guo, Y., Hao, F., Varvello, M., Hilt, V., Steiner, M., Zhang, Z.-L., "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery", in 2012 Proceedings IEEE INFOCOM. IEEE, 2012, pp. 1620–1628.

[23] Böttger, T., Cuadrado, F., Tyson, G., Castro, I., Uhlig, S., "Open Connect Everywhere: A Glimpse at the Internet Ecosystem Through the Lens of the Netflix CDN", ACM SIGCOMM Computer Communication Review, Vol. 48, No. 1, 2018, pp. 28–34.

[24] Torres, R., Finamore, A., Kim, J. R., Mellia, M., Munafo, M. M., Rao, S., "Dissecting Video Server Selection Strategies in the YouTube CDN", in 2011 31st International Conference on Distributed Computing Systems. IEEE, 2011, pp. 248–257.

[25] Guillemin, F., Kauffmann, B., Moteau, S., Simonian, A., "Experimental Analysis of Caching Efficiency for YouTube Traffic in an ISP Network", in Proceedings of the 2013 25th International Teletraffic Congress (ITC). IEEE, 2013, pp. 1–9.

[26] Laterman, M., Arlitt, M., Williamson, C., "A Campus-Level View of Netflix and Twitch: Characterization and Performance Implications", in 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS). IEEE, 2017, pp. 1–8.

[27] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", IETF, RFC 3550, 2003, Available at: https://datatracker.ietf.org/doc/rfc3550/ (Accessed: 28/08/2020).

[28] Schulzrinne, H., Rao, A., Lanphier, R., "Real Time Streaming Protocol (RTSP)", IETF, RFC 2326, 1998, Available at: https://datatracker.ietf.org/doc/rfc2326/ (Accessed: 28/08/2020).

[29] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., Stiemerling, M., "Real-Time Streaming Protocol Version 2.0", IETF, RFC 7826, 2016, Available at: https://datatracker.ietf.org/doc/rfc7826/ (Accessed: 28/08/2020).

[30] Bitmovin, "Adaptive Streaming", Available at: https://bitmovin.com/adaptive-streaming/ (Accessed: 28/08/2020).

[31] Pantos, R., May, W., "HTTP Live Streaming", IETF, Internet Draft, 2017, Available at: https://datatracker.ietf.org/doc/rfc8216/ (Accessed: 28/08/2020).

[32] Iyengar, J., "Why Fastly loves QUIC and HTTP/3", Available at: https://www.fastly.com/blog/why-fastly-loves-quic-http3 (Accessed: 28/08/2020).

[33] Ghedini, A., "Experiment with HTTP/3 using NGINX and quiche", Available at: https://blog.cloudflare.com/experiment-with-http-3-using-nginx-and-quiche/ (Accessed: 28/08/2020).

[34] Crilly, L., "Introducing a Technology Preview of NGINX Support for QUIC and HTTP/3", Available at: https://www.nginx.com/blog/introducing-technology-preview-nginx-support-for-quic-http-3/ (Accessed: 28/08/2020).

[35] Timmerer, C., "HTTP Adaptive Streaming: State of the Art and Challenges Ahead", https://www.slideshare.net/christian.timmerer/http-adaptive-streamingstate-of-the-art-and-challenges-ahead (Accessed: 28/08/2020). 2018.

[36] Fisher, Y. *et al.*, "An Overview of HTTP Adaptive Streaming Protocols for TV Everywhere Delivery", IEEE Trans. on Neural Networks, 2014.

[37] Añorga, J., Arrizabalaga, S., Sedano, B., Goya, J., Alonso-Arce, M., Mendizabal, J., "Analysis of YouTube's Traffic Adaptation to Dynamic Environments", Multimedia Tools and Applications, Vol. 77, No. 7, 2018, pp. 7977–8000.

[38] Wamser, F., Casas, P., Seufert, M., Moldovan, C., Tran-Gia, P., Hossfeld, T., "Modeling the YouTube Stack: From Packets to Quality of Experience", Computer Networks, Vol. 109, 2016, pp. 211–224.

[39] Mondal, A., Sengupta, S., Reddy, B. R., Koundinya, M., Govindarajan, C., De, P., Ganguly, N., Chakraborty, S., "Candid with YouTube: Adaptive Streaming Behavior and Implications on Data Consumption", in Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video, 2017, pp. 19–24.

[40] Zach, O., Slanina, M., “Content Aware Segment Length Optimization for Adaptive Streaming over HTTP”, Radioengineering, Vol. 27, No. 3, 2018, p. 8.

[41] Schwarzmann, S., Zinner, T., Geissler, S., Sieber, C., “Evaluation of the Benefits of Variable Segment Durations for Adaptive Streaming”, in 2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2018, pp. 1–6.

[42] Schwarzmann, S., Hainke, N., Zinner, T., Sieber, C., Robitza, W., Raake, A., “Comparing Fixed and Variable Segment Durations for Adaptive Video Streaming: a Holistic Analysis”, in Proceedings of the 11th ACM Multimedia Systems Conference, 2020, pp. 38–53.

[43] ur Rahman, W., Chung, K., “SABA: Segment and Buffer Aware Rate Adaptation Algorithm for Streaming over HTTP”, Multimedia Systems, Vol. 24, No. 5, 2018, pp. 509–529.

[44] Hoßfeld, T., Moldovan, C., Schwartz, C., “To Each According to his Needs: Dimensioning Video Buffer for Specific User Profiles and Behavior”, in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 1249–1254.

[45] Moldovan, C., Sieber, C., Heegaard, P., Kellerer, W., Hoßfeld, T., “YouTube Can Do Better: Getting the Most out of Video Adaptation”, in 2016 28th International Teletraffic Congress (ITC 28), Vol. 3. IEEE, 2016, pp. 7–12.

[46] Moldovan, C., Loh, F., Seufert, M., Hoßfeld, T., “Optimizing HAS for 360-Degree Videos”, in NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2020, pp. 1–6.

[47] van der Hooft, J., Vega, M. T., Petrangeli, S., Wauters, T., De Turck, F., “Tile-Based Adaptive Streaming for Virtual Reality Video”, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Vol. 15, No. 4, 2019, pp. 1–24.

[48] van der Hooft, J., Vega, M. T., Petrangeli, S., Wauters, T., De Turck, F., “Optimizing Adaptive Tile-Based Virtual Reality Video Streaming”, in 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2019, pp. 381–387.

[49] Nguyen, D. V., Tran, H. T., Pham, A. T., Thang, T. C., “An Optimal Tile-Based Approach for Viewport-Adaptive 360-Degree Video Streaming”, IEEE Journal on Emerging and Selected Topics in Circuits and Systems, Vol. 9, No. 1, 2019, pp. 29–42.

[50] Singla, A., Göring, S., Raake, A., Meixner, B., Koenen, R., Buchholz, T., "Subjective Quality Evaluation of Tile-Based Streaming for Omnidirectional Videos", in Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 232–242.

[51] Schatz, R., Zabrovskiy, A., Timmerer, C., "Tile-Based Streaming of 8K Omnidirectional Video: Subjective and Objective QoE Evaluation", in 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2019, pp. 1–6.

[52] Vega, M. T., van der Hooft, J., Heyse, J., De Backere, F., Wauters, T., De Turck, F., Petrangeli, S., "Exploring New York in 8K: an Adaptive Tile-Based Virtual Reality Video Streaming Experience", in Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 330–333.

[53] Lederer, S., Mueller, C., Grandl, R., "Adaptive Streaming of an Immersive Video Scene", US Patent 10,313,745. June 2019.

[54] van der Hooft, J., Wauters, T., De Turck, F., Timmerer, C., Hellwagner, H., "Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression", in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 2405–2413.

[55] Zhang, X., Toni, L., Frossard, P., Zhao, Y., Lin, C., "Adaptive Streaming in Interactive Multiview Video Systems", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 29, No. 4, 2018, pp. 1130–1144.

[56] Petrangeli, S., Simon, G., Wang, H., Swaminathan, V., "Dynamic Adaptive Streaming for Augmented Reality Applications", in 2019 IEEE International Symposium on Multimedia (ISM). IEEE, 2019, pp. 56–567.

[57] Corbillon, X., De Simone, F., Simon, G., Frossard, P., "Dynamic Adaptive Streaming for Multi-Viewpoint Omnidirectional Videos", in Proceedings of the 9th ACM Multimedia Systems Conference, 2018, pp. 237–249.

[58] Hosseini, M., Timmerer, C., "Dynamic Adaptive Point Cloud Streaming", in Proceedings of the 23rd Packet Video Workshop, 2018, pp. 25–30.

[59] "Microsoft Smooth Streaming", Available at: https://www.microsoft.com/silverlight/smoothstreaming/ (Accessed: 28/08/2020).

[60] "HTTP Live Streaming", Available at: https://developer.apple.com/streaming/ (Accessed: 28/08/2020).

[61] "HTTP Dynamic Streaming", Available at: https://www.adobe.com/products/hds-dynamic-streaming.html (Accessed: 28/08/2020).

[62] "ISO/IEC 23009-2 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 2: Conformance and reference software", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/79107.html (Accessed: 28/08/2020).

[63] "ISO/IEC AWI TR 23009-3 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 3: Implementation guidelines", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/75482.html (Accessed: 28/08/2020).

[64] "ISO/IEC 23009-4:2018 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 4: Segment encryption and authentication", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/73603.html (Accessed: 28/08/2020).

[65] "ISO/IEC 23009-5:2017 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 5: Server and network assisted DASH (SAND)", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/69079.html (Accessed: 28/08/2020).

[66] "ISO/IEC 23009-6:2017 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 6: DASH with server push and WebSockets", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/71072.html (Accessed: 28/08/2020).

[67] "ISO/IEC WD TR 23009-7 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 7: Delivery of CMAF contents with DASH", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/80541.html (Accessed: 28/08/2020).

[68] "ISO/IEC DIS 23009-8 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 8: Session-based DASH operations", International Organization for Standardization, Standard, 2020, Available at: https://www.iso.org/standard/80898.html (Accessed: 28/08/2020).

[69] Sodagar, I., "The MPEG-DASH Standard for Multimedia Streaming Over the Internet", IEEE Multimedia, Vol. 18, No. 4, 2011, pp. 62–67.

[70] "Understanding the HTTP Live Streaming Architecture", Available at: https://developer.apple.com/documentation/http_live_streaming/understanding_the_http_live_streaming_architecture (Accessed: 28/08/2020).

[71] "E.800: Terms and Definitions Related to Quality of Service and Network Performance Including Dependability", International Telecommunication Union, Recommendation, 2008.

[72] "Vocabulary for performance and Quality of Service. Amendment 2: New definitions for inclusion in Recommendation ITU-T P.10/G.100", International Telecommunication Union, Recommendation, 2006.

[73] Le Callet, P., Möller, S., Perkis, A., "Qualinet White Paper on Definitions of Quality of Experience", European Network on Quality of Experience in Multimedia Systems and Services, Tech. Rep. Version 1.2, 2013.

[74] "P.10/G.100: Vocabulary for performance, Quality of Service and Quality of Experience", International Telecommunication Union, Recommendation, 2017.

[75] "P800.2: Mean Opinion Score interpretation and reporting", International Telecommunication Union, Recommendation, 2016.

[76] Schatz, R., Hoßfeld, T., Janowski, L., Egger, S., "From Packets to People: Quality of Experience as a New Measurement Challenge", in Data traffic monitoring and analysis. Springer, 2013, pp. 219–263.

[77] Alreshoodi, M., Woods, J., "Survey on QoE\QoS Correlation Models for Multimedia Services", arXiv preprint arXiv:1306.0221, 2013.

[78] Aroussi, S., Mellouk, A., "Survey on Machine Learning-based QoE-QoS Correlation Models", in Computing, Management and Telecommunications (ComManTel), 2014 International Conference on. IEEE, 2014, pp. 200–204.

[79] Schatz, R., Fiedler, M., Skorin-Kapov, L., "QoE-based Network and Application Management", in Quality of Experience. Springer, 2014, pp. 411–426.

[80] Skorin-Kapov, L., Ivesic, K., Aristomenopoulos, G., Papavassiliou, S., "Approaches for Utility-Based QoE-Driven Optimization of Network Resource Allocation for Multimedia Services.", Data traffic monitoring and analysis, Vol. 7754, 2013, pp. 337–358.

[81] Ivesic, K., Skorin-Kapov, L., Matijasevic, M., "Cross-layer QoE-driven Admission Control and Resource Allocation for Adaptive Multimedia Services in LTE", Journal of Network and Computer Applications, Vol. 46, 2014, pp. 336–351.

[82] Gómez, G., Lorca, J., García, R., Pérez, Q., "Towards a QoE-Driven Resource Control in LTE and LTE-A Networks", Journal of Computer Networks and Communications, Vol. 2013, 2013.

[83] Ahmad, A., Floris, A., Atzori, L., "QoE-Centric Service Delivery: A Collaborative Approach Among OTTs and ISPs", Computer Networks, Vol. 110, 2016, pp. 168–179.

[84] Ahmad, A., Floris, A., Atzori, L., "QoE-Aware Service Delivery: a Joint-Venture Approach for Content and Network Providers", in Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on. IEEE, 2016, pp. 1–6.

[85] Heegaard, P. E., Biczók, G., Toka, L., "Sharing is Power: Incentives for Information Exchange in Multi-Operator Service Delivery", in Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016, pp. 1–7.

[86] Gharakheili, H. H., "Perspectives on Net Neutrality and Internet Fast-lanes", in The Role of SDN in Broadband Networks. Springer, 2017, pp. 5–22.

[87] Ramneek, Hosein, P., Choi, W., Seok, W., "Disruptive Network Applications and Their Impact on Network Neutrality", in Advanced Communication Technology (ICACT), 2015 17th International Conference on. IEEE, 2015, pp. 663–668.

[88] "System architecture for the 5G System, version 15.5.0", European Telecommunications Standards Institute, Standard, 2019.

[89] Baraković, S., Skorin-Kapov, L., "Survey and Challenges of QoE Management Issues in Wireless Networks", Journal of Computer Networks and Communications, Vol. 2013, 2013.

[90] Seufert, M., Egger, S., Slanina, M., Zinner, T., Hoßfeld, T., Tran-Gia, P., "A Survey on Quality of Experience of HTTP Adaptive Streaming", IEEE Communications Surveys & Tutorials, Vol. 17, No. 1, 2015, pp. 469–492.

[91] Casas, P., Seufert, M., Schatz, R., "YOUQMON: A System for On-line Monitoring of YouTube QoE in Operational 3G Networks", ACM SIGMETRICS Performance Evaluation Review, Vol. 41, No. 2, 2013, pp. 44–46.

[92] Hoßfeld, T., Schatz, R., Biersack, E., Plissonneau, L., "Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience", in Data Traffic Monitoring and Analysis. Springer, 2013, pp. 264–301.

[93] Hoßfeld, T., Egger, S., Schatz, R., Fiedler, M., Masuch, K., Lorentzen, C., "Initial Delay vs. Interruptions: Between the Devil and the Deep Blue Sea", in Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on. IEEE, 2012, pp. 1–6.

[94] Ghadiyaram, D., Bovik, A. C., Yeganeh, H., Kordasiewicz, R., Gallant, M., "Study of the Effects of Stalling Events on the Quality of Experience of Mobile Streaming Videos", in

Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on. IEEE, 2014, pp. 989–993.

[95] Juluri, P., Tamarapalli, V., Medhi, D., "Measurement of Quality of Experience of Video-on-Demand Services: A Survey", IEEE Communications Surveys & Tutorials, Vol. 18, No. 1, 2015, pp. 401–418.

[96] Barman, N., Martini, M. G., "QoE Modeling for HTTP Adaptive Video Streaming–A Survey and Open Challenges", IEEE Access, Vol. 7, 2019, pp. 30 831–30 859.

[97] "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport", International Telecommunication Union, Recommendation P.1203, 2017.

[98] "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport - Video quality estimation module", International Telecommunication Union, Recommendation P.1203.1, jan 2019.

[99] Raake, A., Garcia, M.-N., Robitza, W., List, P., Göring, S., Feiten, B., "A Bitstream-based, Scalable Video-Quality Model for HTTP Adaptive Streaming: ITU-T P. 1203.1", in Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on. IEEE, 2017, pp. 1–6.

[100] "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport - Audio quality estimation module", International Telecommunication Union, Recommendation P.1203.2, oct 2017.

[101] "Parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport - Quality integration module", International Telecommunication Union, Recommendation P.1203.3, jan 2019.

[102] Robitza, W., Garcia, M.-N., Raake, A., "A Modular HTTP Adaptive Streaming QoE Model—Candidate for ITU-T P.1203 ("P.NATS")", in Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on. IEEE, 2017, pp. 1–6.

[103] "ITU-T Rec. P.1203 Standalone Implementation", Available at: https://github.com/itu-p1203/itu-p1203 (Accessed: 28/08/2020).

[104] Robitza, W., Göring, S., Raake, A., Lindegren, D., Heikkilä, G., Gustafsson, J., List, P., Feiten, B., Wüstenhagen, U., Garcia, M.-N., Yamagishi, K., Broom, S., "HTTP Adaptive Streaming QoE Estimation with ITU-T Rec. P.1203 – Open Databases and Software", in 9th ACM Multimedia Systems Conference, Amsterdam, 2018.

[105] "Codec Extension for ITU-T P.1203", Available at: https://github.com/Telecommunication-Telemedia-Assessment/itu-p1203-codecextension (Accessed: 28/08/2020).

[106] Ramachandra Rao, R. R., Göring, S., Vogel, P., Pachatz, N., Villarreal, J. J. V., Robitza, W., List, P., Feiten, B., Raake, A., "Adaptive Video Streaming With Current Codecs and Formats: Extensions to Parametric Video Quality Model ITU-T P. 1203", Electronic Imaging, Vol. 2019, No. 10, 2019, pp. 314–1.

[107] "Video quality assessment of streaming services over reliable transport for resolutions up to 4K", International Telecommunication Union, Recommendation P.1204, jan 2020.

[108] Rao, R. R. R., Göring, S., List, P., Robitza, W., Feiten, B., Wüstenhagen, U., Raake, A., "Bitstream-Based Model Standard for 4K/UHD: ITU-T P. 1204.3—Model Details, Evaluation, Analysis and Open Source Implementation", in 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2020, pp. 1–6.

[109] Mushtaq, M. S., Augustin, B., Mellouk, A., "Empirical Study Based on Machine Learning Approach to Assess the QoS/QoE Correlation", in Networks and Optical Communications (NOC), 2012 17th European Conference on. IEEE, 2012, pp. 1–7.

[110] "Ericsson NPT application", Available at: https://play.google.com/store/apps/details?id=com.ericsson.mbbmeasurement (Accessed: 28/08/2020).

[111] "V3D", Available at: https://www.v3d.fr (Accessed: 28/08/2020).

[112] Wamser, F., Seufert, M., Casas, P., Irmer, R., Tran-Gia, P., Schatz, R., "YoMoApp: A Tool for Analyzing QoE of YouTube HTTP Adaptive Streaming in Mobile Networks", in 2015 European Conference on Networks and Communications (EuCNC). IEEE, 2015, pp. 239–243.

[113] Schwind, A., Seufert, M., Alay, Ö., Casas, P., Tran-Gia, P., Wamser, F., "Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed", in Network Traffic Measurement and Analysis Conference (TMA), 2017. IEEE, 2017, pp. 1–6.

[114] Nam, H., Kim, K.-H., Calin, D., Schulzrinne, H., "YouSlow: a Performance Analysis Tool for Adaptive Bitrate Video Streaming", in ACM SIGCOMM Computer Communication Review, Vol. 44, No. 4. ACM, 2014, pp. 111–112.

[115] Casas, P., Seufert, M., Wamser, F., Gardlo, B., Sackl, A., Schatz, R., "Next to You: Monitoring Quality of Experience in Cellular Networks from the End-Devices", IEEE Transactions on Network and Service Management, Vol. 13, No. 2, 2016, pp. 181–196.

[116] "YouTube Player API Reference for IFrame Embeds", Available at: https://developers.google.com/youtube/iframe_api_reference (Accessed: 28/08/2020).

[117] "YouTube Android Player API", Available at: https://developers.google.com/youtube/android/player (Accessed: 28/08/2020).

[118] Seufert, M., Casas, P., Wamser, F., Wehner, N., Schatz, R., Tran-Gia, P., "Application-layer Monitoring of QoE Parameters for Mobile YouTube Video Streaming in the Field", in Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on. IEEE, 2016, pp. 411–416.

[119] Casas, P., D'Alconzo, A., Wamser, F., Seufert, M., Gardlo, B., Schwind, A., Tran-Gia, P., Schatz, R., "Predicting QoE in Cellular Networks Using Machine Learning and In-smartphone Measurements", in Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on. IEEE, 2017, pp. 1–6.

[120] Seufert, M., Wehner, N., Wamser, F., Casas, P., D'Alconzo, A., Tran-Gia, P., "Unsupervised QoE Field Study for Mobile YouTube Video Streaming with YoMoApp", in Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on. IEEE, 2017, pp. 1–6.

[121] Schwind, A., Seufert, M., Alay, O., Casas, P., Tran-Gia, P., Wamser, F., "Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed", in IEEE/IFIP Workshop on Mobile Network Measurement (MNM'17), 2017, pp. 1–6.

[122] Añorga, J., Arrizabalaga, S., Sedano, B., Alonso-Arce, M., Mendizabal, J., "YouTube's DASH Implementation Analysis", in 19th International Conference on Circuits, Systems, Communications and Computers (CSCC), 2015, pp. 61–66.

[123] Añorga, J., Arrizabalaga, S., Sedano, B., Goya, J., Alonso-Arce, M., Mendizabal, J., "Analysis of YouTube's Traffic Adaptation to Dynamic Environments", Multimedia Tools and Applications, 2017, pp. 1–24.

[124] Seufert, M., Zeidler, B., Wamser, F., Karagkioules, T., Tsilimantos, D., Loh, F., Tran-Gia, P., Valentin, S., "A Wrapper for Automatic Measurements with YouTube's Native Android App", in 2018 Network Traffic Measurement and Analysis Conference (TMA). IEEE, 2018, pp. 1–8.

[125] Szabo, G., Rácz, S., Malomsoky, S., Bolle, A., "Potential Gains of Reactive Video QoE Enhancement by App Agnostic QoE Deduction", in Global Communications Conference (GLOBECOM), 2016 IEEE. IEEE, 2016, pp. 1–7.

[126] Seufert, M., Casas, P., Wehner, N., Gang, L., Li, K., "Stream-based Machine Learning for Real-time QoE Analysis of Encrypted Video Streaming Traffic", in 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). IEEE, 2019, pp. 76–81.

[127] Seufert, M., Casas, P., Wehner, N., Gang, L., Li, K., "Features that Matter: Feature Selection for On-line Stalling Prediction in Encrypted Video Streaming", in IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2019, pp. 688–695.

[128] Wassermann, S., Seufert, M., Casas, P., Gang, L., Li, K., "Let me Decrypt your Beauty: Real-time Prediction of Video Resolution and Bitrate for Encrypted Video Streaming", in 2019 Network Traffic Measurement and Analysis Conference (TMA). IEEE, 2019, pp. 199–200.

[129] Wassermann, S., Seufert, M., Casas, P., Gang, L., Li, K., "I See What you See: Real Time Prediction of Video Quality from Encrypted Streaming Traffic", in Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks, 2019, pp. 1–6.

[130] "Selenium", Available at: http://www.seleniumframework.com/ (Accessed: 28/08/2020).

[131] "Selendroid", Available at: http://selendroid.io/ (Accessed: 28/08/2020).

[132] "Appium", Available at: http://appium.io/ (Accessed: 28/08/2020).

[133] Ahmad, A., Floris, A., Atzori, L., "Towards QoE Monitoring at User Terminal: A Monitoring Approach Based on Quality Degradation", in Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on. IEEE, 2017, pp. 1–6.

[134] Ahmad, A., Atzori, L., Martini, M. G., "Qualia: A Multilayer Solution for QoE Passive Monitoring at the User Terminal", in 2017 IEEE International Conference on Communications (ICC). IEEE, 2017, pp. 1–6.

[135] Casas, P., Schatz, R., Hoßfeld, T., "Monitoring YouTube QoE: Is Your Mobile Network Delivering the Right Experience to Your Customers?", in 2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2013, pp. 1609–1614.

[136] Schatz, R., Hoßfeld, T., Casas, P., "Passive YouTube QoE Monitoring for ISPs", in 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. IEEE, 2012, pp. 358–364.

[137] Mangla, T., Halepovic, E., Ammar, M., Zegura, E., "Using Session Modeling to Estimate HTTP-Based Video QoE Metrics From Encrypted Network Traffic", IEEE Transactions on Network and Service Management, Vol. 16, No. 3, 2019, pp. 1086–1099.

[138] Krishnamoorthi, V., Carlsson, N., Halepovic, E., Petajan, E., "BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients", in Proceedings of the 8th ACM on Multimedia Systems Conference, 2017, pp. 76–87.

[139] Bronzino, F., Schmitt, P., Ayoubi, S., Martins, G., Teixeira, R., Feamster, N., "Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience", Proceedings of the ACM on Measurement and Analysis of Computing Systems, Vol. 3, No. 3, 2019, pp. 1–25.

[140] Gutterman, C., Guo, K., Arora, S., Wang, X., Wu, L., Katz-Bassett, E., Zussman, G., "Requet: Real-time QoE Detection for Encrypted YouTube Traffic", in Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 48–59.

[141] Mazhar, M. H., Shafiq, Z., "Real-Time Video Quality of Experience Monitoring for HTTPS and QUIC", in IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018, pp. 1331–1339.

[142] Bartolec, I., Orsolic, I., Skorin-Kapov, L., "In-Network YouTube Performance Estimation in Light of End User Playback-Related Interactions", in 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2019, pp. 1–3.

[143] Bartolec, I., Orsolic, I., Skorin-Kapov, L., "Inclusion of End User Playback-Related Interactions in YouTube Video Data Collection and ML-Based Performance Model Training", in 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX), 2020, pp. 1-6.

[144] Schwarzmann, S., Cassales Marquezan, C., Bosk, M., Liu, H., Trivisonno, R., Zinner, T., "Estimating Video Streaming QoE in the 5G Architecture Using Machine Learning", in Proceedings of the 4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks, 2019, pp. 7–12.

[145] Schwarzmann, S., Marquezan, C. C., Trivisonno, R., Nakajima, S., Zinner, T., "Accuracy vs. Cost Trade-off for Machine Learning Based QoE Estimation in 5G Networks", in ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020, pp. 1–6.

[146] Schwarzmann, S., Zinner, T., Dobrijevic, O., "Towards a Framework for Comparing Application-Network Interaction Mechanisms", in 2016 28th International Teletraffic Congress (ITC 28), Vol. 3. IEEE, 2016, pp. 13–18.

[147] Bentaleb, A., Taani, B., Begen, A. C., Timmerer, C., Zimmermann, R., "A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP", IEEE Communications Surveys & Tutorials, Vol. 21, No. 1, 2018, pp. 562–585.

[148] Nam, H., Schulzrinne, H., Nam, H., Kim, K.-H., Schulzrinne, H., Varela, M., Nam, H., Schulzrinne, H., Mäki, T., Nam, H. *et al.*, "YouSlow: What Influences User Abandonment Behavior for Internet Video?", Columbia University Rcp, 2016, Available at: http://www.cs.columbia.edu/~hn2203/papers/12_youslow_transaction_on_networking.pdf (Accessed: 28/08/2020).

[149] Plakia, M., Tzamousis, E., Asvestopoulou, T., Pantermakis, G., Filippakis, N., Schulzrinne, H., Kane-Esrig, Y., Papadopouli, M., "Should I Stay or Should I Go: Analysis of the Impact of Application QoS on User Engagement in YouTube", ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), Vol. 5, No. 2, 2020, pp. 1–32.

[150] Thomas, E., Van Deventer, M., Stockhammer, T., Begen, A. C., Famaey, J., "Enhancing mpeg dash performance via server and network assistance", SMPTE Motion Imaging Journal, Vol. 126, No. 1, 2017, pp. 22–27.

[151] Thomas, E., van Deventer, M., Stockhammer, T., Begen, A. C., Champel, M.-L., Oyman, O., "Applications and Deployments of Server and Network Assisted DASH (SAND)", 2016, Available at: http://www.employees.org/~acbegen/files/articles/IBC16_SAND.pdf (Accessed: 28/08/2020).

[152] Cofano, G., De Cicco, L., Zinner, T., Nguyen-Ngoc, A., Tran-Gia, P., Mascolo, S., "Design and Experimental Evaluation of Network-Assisted Strategies for HTTP Adaptive Streaming", in Proceedings of the 7th International Conference on Multimedia Systems, 2016, pp. 1–12.

[153] Pham, S., Heeren, P., Schmidt, C., Silhavy, D., Arbanowski, S., "Evaluation of Shared Resource Allocation Using SAND for ABR Streaming", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Vol. 16, No. 2s, 2020, pp. 1–18.

[154] Bentaleb, A., Begen, A. C., Zimmermann, R., "SDNDASH: Improving QoE of HTTP Adaptive Streaming Using Software Defined Networking", in Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 1296–1305.

[155] Liotou, E., Tseliou, G., Samdanis, K., Tsolkas, D., Adelantado, F., Verikoukis, C., "An SDN QoE-Service for Dynamically Enhancing the Performance of OTT Applications", in 2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX). IEEE, 2015, pp. 1–2.

[156] Barakabitze, A. A., Liyanage, M., Hines, A., "QoESoft: QoE Management Architecture for Softwarized 5G Networks", in 2020 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2020, pp. 1–6.

[157] Awobuluyi, O., Nightingale, J., Wang, Q., Alcaraz-Calero, J. M., "Video Quality in 5G Networks: Context-Aware QoE Management in the SDN Control Plane", in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. IEEE, 2015, pp. 1657–1662.

[158] Grigoriou, E., Barakabitze, A. A., Atzori, L., Sun, L., Pilloni, V., "An SDN-Approach for QoE Management of Multimedia Services Using Resource Allocation", in 2017 IEEE International Conference on Communications (ICC). IEEE, 2017, pp. 1–7.

[159] Robitza, W., Ahmad, A., Kara, P. A., Atzori, L., Martini, M. G., Raake, A., Sun, L., "Challenges of Future Multimedia QoE Monitoring for Internet Service Providers", Multimedia Tools and Applications, Vol. 76, No. 21, 2017, pp. 22 243–22 266.

[160] Jarschel, M., Wamser, F., Hohn, T., Zinner, T., Tran-Gia, P., "SDN-based Application-Aware Networking on the Example of YouTube Video Streaming", in 2013 Second European Workshop on Software Defined Networks. IEEE, 2013, pp. 87–92.

[161] Kassler, A., Skorin-Kapov, L., Dobrijevic, O., Matijasevic, M., Dely, P., "Towards QoE-Driven Multimedia Service Negotiation and Path Optimization with Software Defined Networking", in SoftCOM 2012, 20th International Conference on Software, Telecommunications and Computer Networks. IEEE, 2012, pp. 1–5.

[162] "P4", Available at: https://p4.org/ (Accessed: 28/08/2020).

[163] Bhamare, D., Kassler, A., Vestin, J., Khoshkholghi, M. A., Taheri, J., "IntOpt: In-Band Network Telemetry Optimization for NFV Service Chain Monitoring", in ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019, pp. 1–7.

[164] Vestin, J., Kassler, A., Bhamare, D., Grinnemo, K.-J., Andersson, J.-O., Pongracz, G., "Programmable Event Detection for In-Band Network Telemetry", 2019, Available at: https://arxiv.org/pdf/1909.12101.pdf (Accessed: 28/08/2020).

[165] Langlet, J., Kassler, A., Bhamare, D., "Towards Neural Network Inference on Programmable Switches", 2019, Available at: https://kau.app.box.com/s/4k41n6s30d0fh1rawnhmedt0akkdgtnn (Accessed: 28/08/2020).

[166] "scikit-learn: Machine Learning in Python", Available at: https://scikit-learn.org/stable/ (Accessed: 28/08/2020).

[167] Casas, P., Seufert, M., Wehner, N., Schwind, A., Wamser, F., "Enhancing Machine Learning based QoE Prediction by Ensemble Models", in 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018, pp. 1642–1647.

[168] Zec, M., Mikuc, M., "Operating System Support for Integrated Network Emulation in IMUNES", in Workshop on Operating System and Architectural Support for the on demand IT Infrastructure (1; 2004), 2004.

[169] "Integrated Multiprotocol Network Emulator/Simulator", Available at: http://imunes.net/ (Accessed: 28/08/2020).

[170] "Net.Shark", Available at: http://www.albedotelecom.com/pages/fieldtools/src/netshark.php (Accessed: 28/08/2020).

[171] "TCPDUMP/LIBPCAP public repository", Available at: https://www.tcpdump.org/ (Accessed: 28/08/2020).

[172] "4G/LTE Bandwidth Logs", Available at: https://users.ugent.be/~jvdrhoof/dataset-4g/ (Accessed: 28/08/2020).

[173] van der Hooft, J., Petrangeli, S., Wauters, T., Huysegems, R., Alface, P. R., Bostoen, T., De Turck, F., "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks", IEEE Communications Letters, Vol. 20, No. 11, 2016, pp. 2177-2180.

[174] Riiser, H., Endestad, T., Vigmostad, P., Griwodz, C., Halvorsen, P., "Video Streaming Using a Location-based Bandwidth-lookup Service for Bitrate Planning", ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), Vol. 8, No. 3, 2012, p. 24.

[175] "Commute Path Bandwidth Traces from 3G Networks", Available at: https://folk.universitetetioslo.no/paalh/dataset/hsdpa-tcp-logs/ (Accessed: 28/08/2020).

[176] Orsolic, I., "YouTube Performance Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning", Master's thesis, University of Zagreb, Faculty of Electrical Engineering and Computing, 2016.

[177] Pevec, D., “Measurements of YouTube Traffic and Performance Based on Network and Client-Side Data Collection”, Master’s thesis, University of Zagreb, Faculty of Electrical Engineering and Computing, 2016.

[178] “Weka 3: Data Mining Software in Java”, Available at: http://www.cs.waikato.ac.nz/ml/weka/ (Accessed: 28/08/2020).

[179] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P., “SMOTE: Synthetic Minority Over-Sampling Technique”, Journal of artificial intelligence research, Vol. 16, 2002, pp. 321–357.

[180] “Imbalanced Learn Documentation”, Available at: https://imbalanced-learn.org/stable/ (Accessed: 28/08/2020).

[181] “MLxtend Documentation”, Available at: http://rasbt.github.io/mlxtend/ (Accessed: 28/08/2020).

[182] Tsilimantos, D., Karagkioules, T., Valentin, S., “Classifying Flows and Buffer State for YouTube’s HTTP Adaptive Streaming Service in Mobile Networks”, in Proc. of ACM Multimedia Systems Conf. (MMSys 2018), June 2018.

[183] Nguyen, T. T., Armitage, G., “A Survey of Techniques for Internet Traffic Classification Using Machine Learning”, Communications Surveys & Tutorials, IEEE, Vol. 10, No. 4, 2008, pp. 56–76.

[184] Chen, Q. A., Luo, H., Rosen, S., Mao, Z. M., Iyer, K., Hui, J., Sontineni, K., Lau, K., “QoE Doctor: Diagnosing Mobile App QoE with Automated UI Control and Cross-layer Analysis”, in Proceedings of the 2014 Conference on Internet Measurement Conference. ACM, 2014, pp. 151–164.

[185] Archibald, R., Liu, Y., Corbett, C., Ghosal, D., “Disambiguating HTTP: Classifying Web Applications”, in Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International. IEEE, 2011, pp. 1808–1813.

[186] “YouTube Data API”, Available at: https://developers.google.com/youtube/v3 (Accessed: 28/08/2020).

[187] Sieber, C., Blenk, A., Hinteregger, M., Kellerer, W., “The Cost of Aggressive HTTP Adaptive Streaming: Quantifying YouTube’s Redundant Traffic”, in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). IEEE, 2015, pp. 1261–1267.

[188] Sackl, A., Casas, P., Schatz, R., Janowski, L., Irmer, R., "Quantifying the Impact of Network Bandwidth Fluctuations and Outages on Web QoE", in Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on. IEEE, 2015, pp. 1–6.

[189] Roughan, M., Sen, S., Spatscheck, O., Duffield, N., "Class-of-Service Mapping for QoS: a Statistical Signature-based Approach to IP Traffic Classification", in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement. ACM, 2004, pp. 135–148.

[190] Moore, A. W., Zuev, D., "Internet Traffic Classification Using Bayesian Analysis Techniques", in ACM SIGMETRICS Performance Evaluation Review, Vol. 33, No. 1. ACM, 2005, pp. 50–60.

[191] Han, Y.-T., Park, H.-S., "Game Traffic Classification Using Statistical Characteristics at the Transport Layer", ETRI journal, Vol. 32, No. 1, 2010, pp. 22–32.

[192] Callado, A., Kamienski, C., Szabó, G., Gerö, B. P., Kelner, J., Fernandes, S., Sadok, D., "A Survey on Internet Traffic Identification", Comm. Surveys & Tutorials, IEEE, Vol. 11, No. 3, 2009, pp. 37–52.

[193] Moore, A., Zuev, D., Crogan, M., "Discriminators for Use in Flow-Based Classification", 2005, Available at: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf (Accessed: 28/08/2020).

[194] Moorthy, A. K., Choi, L. K., Bovik, A. C., De Veciana, G., "Video Quality Assessment on Mobile Devices: Subjective, Behavioral and Objective Studies", IEEE Journal of Selected Topics in Signal Processing, Vol. 6, No. 6, 2012, pp. 652–671.

[195] Hoßfeld, T., Seufert, M., Sieber, C., Zinner, T., "Assessing Effect Sizes of Influence Factors Towards a QoE Model for HTTP Adaptive Streaming", in Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on. IEEE, 2014, pp. 111–116.

[196] Eckert, M., Knoll, T. M., "ISAAR (Internet Service Quality Assessment and Automatic Reaction) a QoE Monitoring and Enforcement Framework for Internet Services in Mobile Networks", in International Conference on Mobile Networks and Management. Springer, 2012, pp. 57–70.

[197] "Net Promoter", Available at: http://www.netpromoter.com/know/ (Accessed: 28/08/2020).

[198] Hoßfeld, T., Heegaard, P. E., Varela, M., "QoE Beyond the MOS: Added Value Using Quantiles and Distributions", in Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on. IEEE, 2015, pp. 1–6.

[199] Holte, R. C., "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", Machine learning, Vol. 11, No. 1, 1993, pp. 63–90.

[200] Keogh, E., "Naïve bayes classifier", Available at: http://www.cs.ucr.edu/~eamonn/CE/Bayesian%20Classification%20withInsect_examples.pdf (Accessed: 28/08/2020). 2015.

[201] Platt, J. C., "12 Fast Training of Support Vector Machines Using Sequential Minimal Optimization", Advances in kernel methods, 1999, pp. 185–208.

[202] "Data Mining with Weka: Decision trees", Available at: https://www.youtube.com/watch?v=l7R9NHqvI0Y (Accessed: 28/08/2020).

[203] Breiman, L., "Random forests", Machine learning, Vol. 45, No. 1, 2001, pp. 5–32.

[204] RStudio, "Shiny", Available at: http://shiny.rstudio.com (Accessed: 28/08/2020).

[205] "Tesseract OCR", Available at: https://github.com/tesseract-ocr/tesseract (Accessed: 28/08/2020).

[206] Ahmad, A., Floris, A., Atzori, L., "OTT-ISP Joint Service Management: a Customer Lifetime Value Based Approach", in 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, 2017, pp. 1017–1022.

[207] "YouTube DL", Available at: https://ytdl-org.github.io/youtube-dl/index.html (Accessed: 28/08/2020).

[208] Bronzino, F., Schmitt, P., Ayoubi, S., Feamster, N., Teixeira, R., Wassermann, S., Sundaresan, S., "Lightweight, General Inference of Streaming Video Quality from Encrypted Traffic", 2019, Available at: https://hal.inria.fr/hal-02074823/document (Accessed: 28/08/2020).

[209] Gama, J., Medas, P., Castillo, G., Rodrigues, P., "Learning with Drift Detection", in Brazilian symposium on artificial intelligence. Springer, 2004, pp. 286–295.

[210] Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N. D., Dataset Shift in Machine Learning. The MIT Press, 2009.

[211] "Covariate Shift – Unearthing Hidden Problems in Real World Data Science", Analytics Vidhya, Tech. Rep., 2017, Available at: https://www.analyticsvidhya.com/blog/2017/07/covariate-shift-the-hidden-problem-of-real-world-data-science/ (Accessed: 28/08/2020).

[212] Widmer, G., Kubat, M., "Learning in the Presence of Concept Drift and Hidden Contexts", Machine learning, Vol. 23, No. 1, 1996, pp. 69–101.

[213] Tsymbal, A., “The Problem of Concept Drift: Definitions and Related Work”, Computer Science Department, Trinity College Dublin, Vol. 106, No. 2, 2004, p. 58.

[214] “Handling Concept Drift: Importance, Challenges and Solutions”, The University of Waikato, Tech. Rep., 2011, Available at: https://www.cs.waikato.ac.nz/~abifet/PAKDD2011/PAKDD11Tutorial_Handling_Concept_Drift.pdf (Accessed: 28/08/2020).

[215] Stanley, K. O., “Learning Concept Drift with a Committee of Decision Trees”, Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA, 2003.

[216] Žliobaitė, I., Pechenizkiy, M., Gama, J., “An Overview of Concept Drift Applications”, in Big data analysis: new algorithms for a new society. Springer, 2016, pp. 91–114.

[217] Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R., “Early Drift Detection Method”, in Fourth international workshop on knowledge discovery from data streams, Vol. 6, 2006, pp. 77–86.

List of Abbreviations

ABR Adaptive Bitrate
ACR Absolute Category Rating
API Application Programming Interface

CDN Content Delivery Network

DASH Dynamic Adaptive Streaming over HTTP
DPI Deep Packet Inspection
DT Decision Tree

HAS HTTP Adaptive Streaming
HLS HTTP Live Streaming
HTTP HyperText Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure

IETF Internet Engineering Task Force
ISO International Organization for Standardization
ISP Internet Service Provider
ITU International Telecommunication Union

KPI Key Performance Indicator

ML Machine Learning
MOS Mean Opinion Score
MPD Media Presentation Description

NFV Network Function Virtualization

OTT Over-The-Top

QoE Quality of Experience
QoS Quality of Service
QUIC Quick UDP Internet Connections (not used as an acronym within IETF RFC)

RF Random Forest

RFC Request for Comments
RTCP RTP Control Protocol
RTP Real-time Transport Protocol
RTSP Real Time Streaming Protocol

SAND Server and Network Assisted DASH
SDN Software-Defined Networking
SFS Sequential Feature Selection
SLA Service Level Agreement

TCP Transmission Control Protocol
TLS Transport Layer Security

UDP User Datagram Protocol
URI Uniform Resource Identifier
URL Uniform Resource Locator

VoD Video on Demand

XML Extensible Markup Language

List of Figures

List of Tables

Biography

Irena Oršolić was born on the 27th of June, 1991, in Vinkovci, Croatia. She received her B.Sc. degree in the field of Computing in July 2014 and her M.Sc. degree in the field of Information and Communication Technology in July 2016 from Faculty of Electrical Engineering and Computing, University of Zagreb (FER). As of October 2016, she is working as a Research Assistant at the Department of Telecommunications, FER, funded by the Croatian Science Foundation. Same year, she enrolled in the PhD program at the Faculty, field of Computing, under the supervision of Assoc. Prof. Lea Skorin-Kapov. In October 2017 she passed her PhD qualifying exam, and in October 2018 defended the doctoral dissertation topic, entitled "Quality of Experience estimation of encrypted video streaming by using machine learning methods".

The focus of her research is on investigating potential solutions for QoE monitoring based on the analysis of encrypted network traffic using machine learning methods, which could possibly enable the development of mechanisms for improving QoE and efficient use of network resources. Her research is conducted in the scope of activities of The Multimedia Quality of Experience Research Lab (MUEXlab), and projects funded by the Croatian Science Foundation and Ericsson Nikola Tesla. She has authored or co-authored 12 papers published in international journals and conference proceedings, and participated in a number of international conferences, workshops and PhD schools.

List of publications

Journal papers

1. Oršolić, I., Skorin-Kapov, L., "A Framework for In-Network QoE Monitoring of Encrypted Video Streaming", IEEE Access, Vol. 8, 2020, pp. 74 691–74 706.
2. Oršolić, I., Pevec, D., Sužnjevic, M., Skorin-Kapov, L., "A Machine Learning Approach to Classifying YouTube QoE Based on Encrypted Network Traffic", Multimedia Tools and Applications, Vol. 76, No. 21, 2017, pp. 22 267–22 301.

Conference papers

1. Bartolec, I., Oršolić, I., Skorin-Kapov, L., “Inclusion of End User Playback-related Interactions in YouTube Video Data Collection and ML-based Performance Model Training”, 12th International Conference on Quality of Multimedia Experience (QoMEX), 2020, pp. 1–6
2. Oršolić, I., Skorin-Kapov, L., Hoßfeld, T., “To Share or Not to Share? How Exploitation of Context Data Can Improve In-Network QoE Monitoring of Encrypted YouTube Streams”, 11th International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2019, pp. 1–3.
3. Bartolec, I., Oršolić, I., Skorin-Kapov, L., “In-Network YouTube Performance Estimation in Light of End User Playback-Related Interactions”, 11th International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2019, pp. 1–3.
4. Oršolić, I., Rebernjak, P., Sužnjević, M., Skorin-Kapov, L., “In-Network QoE and KPI Monitoring of Mobile YouTube Traffic: Insights for Encrypted iOS Flows”, 14th International Conference on Network and Service Management (CNSM). IEEE, 2018, pp. 233–239.
5. Oršolić, Irena, and Lea Skorin-Kapov. "In-network Quality of Experience Monitoring of HTTP Adaptive Streaming Services." International Conference on Smart Systems and Technologies (SST) – PhD Forum, 2018, pp. 1–2.
6. Oršolić, I., Sužnjevic, M., Skorin-Kapov, L., “YouTube QoE Estimation from Encrypted Traffic: Comparison of Test Methodologies and Machine Learning Based Models”, 10th International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2018, pp. 1–6.
7. Oršolić, Irena, and Lea Skorin-Kapov. "YouTube QoE Classification Based on Monitoring of Encrypted Traffic.", 15th International Conference on Telecommunications (ConTEL) – PhD Forum, 2017, pp. 1–2.
8. Oršolić, I., Skorin-Kapov, L., Sužnjević, M., “Towards a Framework for Classifying YouTube QoE Based on Monitoring of Encrypted Traffic”, International Young Researcher Summit on Quality of Experience in Emerging Multimedia Services (QEEMS), 2017, pp. 1–5.
9. Oršolić, I., Pevec, D., Sužnjević, M., Skorin-Kapov, L., “YouTube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning”, IEEE Globecom Workshops (GC Wkshps). IEEE, 2016, pp. 1–6.
10. Oršolić, Irena, and Lea Skorin-Kapov. "Towards Cooperative QoE Management Schemes for Multimedia Applications." 28th International Conference on Software, Telecommunications and Computer Networks (SoftCOM) – PhD Forum, 2016, pp. 1–2.

Životopis

Irena Oršolić je rođena 27. lipnja 1991. u Vinkovcima. Završila je preddiplomski studij Računarstvo u srpnju 2014., te diplomski studij Informacijska i komunikacijska tehnologija u srpnju 2016. na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu (FER). Od listopada 2016. je zaposlena kao asistentica na Zavodu za telekomunikacije FER-a, financirana od strane Hrvatske zaklade za znanost (HRZZ). Iste godine, upisala je doktorski studij Fakulteta, polje računarstvo, pod mentorstvom izv. prof. dr. sc. Lee Skorin-Kapov. U listopadu 2017. položila je kvalifikacijski doktorski ispit, a u listopadu 2018. održala javni razgovor o očekivanom izvornom znanstvenom doprinosu disertacije, na kojem je odobrena tema "Procjena iskustvene kvalitete za strujanje šifriranog videa primjenom metoda strojnog učenja".

Fokus njenog istraživanja je na razmatranju potencijalnih rješenja za praćenje iskustvene kvalitete temeljenih na analizi šifriranog mrežnog prometa koristeći metode strojnog učenja, koja bi omogućila razvoj mehanizama za poboljšanje iskustvene kvalitete i efikasno korištenje mrežnih resursa. Istraživanje provodi u sklopu aktivnosti Laboratorija za istraživanje iskustvene kvalitete višemedijskih usluga (MUEXlab) i projekata financiranih od strane HRZZ-a i tvrtke Ericsson Nikola Tesla. Autorica je i koautorica 12 članaka objavljenih u međunarodnim časopisima i na konferencijama, te je sudjelovala na brojnim međunarodnim konferencijama, radionicama i doktorskim školama.