Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Telecommunications and Media Informatics

Estimation of encrypted video service quality in mobile networks using machine learning methods

Diploma Thesis Design 1

Author
András Kővári

Supervisors
dr. László Kovács
Ericsson Hungary Ltd.
dr. Attila Vidács
BME-TMIT

May 17, 2018

Contents

Introduction

Nowadays, the majority of the mobile internet traffic consists of video, and this trend is increasing [1], forecasted to reach 75% percent by 2020. The expectation is based not only on the spread of higher resolution videos (HD, UHD, 4K) being presented, but it's also motivated by the consumer habits shifting towards watching this content on mobile devices (reaching 25% of all content being watched on smartphone screens by 2020 [2], 1). Additionally, an increasing amount of the traffic from the OTT (Over-The-Top) video service providers is encrypted ([3], [4]).

Encryption helps to ensure data privacy of the end users, but makes it difficult for the network operators to monitor the quality of the services used via the network. An even greater variety of network conditions exist, that impact the Quality of Experience of the user in mobile networks.

The Quality of Experience is concerned with the end user's satisfaction, that may be, after all, the most important factor for the user when deciding which services to use, but it's the most difficult to (objectively) measure for the operators or service providers. There have been a number of approaches trying to solve the problem by estimating a QoE class or MOS (Mean Opinion Score) for the video sessions, even in the encrypted case. This is useful for a bird's eye view, but when the scores start to drop, finer resolution information about the sessions is desirable, to troubleshoot the problems, correlate the issues with other events in the network, while performing root cause analysis. As mentioned above, with the increase of video traffic in mobile networks, extracting this kind of information is valuable. The goal of this thesis is to provide a method for estimating the video quality on a fine granularity, based on encrypted traffic, by estimating some of the key factors influencing the Quality of Experience. The analysis builts on the network traffic measurements of video sessions provided by Ericsson Hungary. The HTTP (and TCP) based video streaming is in focus, while RTP and RTSP is out of scope.

The task is planned to be accomplished in two semesters, this report of the first semester is organized as follows:

The first section presents the background of the problem, related work and accomplishments in this area. The second section describes the tools and methodology that was used, while the third section shows the preliminary results from this semester. Section 4 summarizes the current status and the planned work for the next semester.

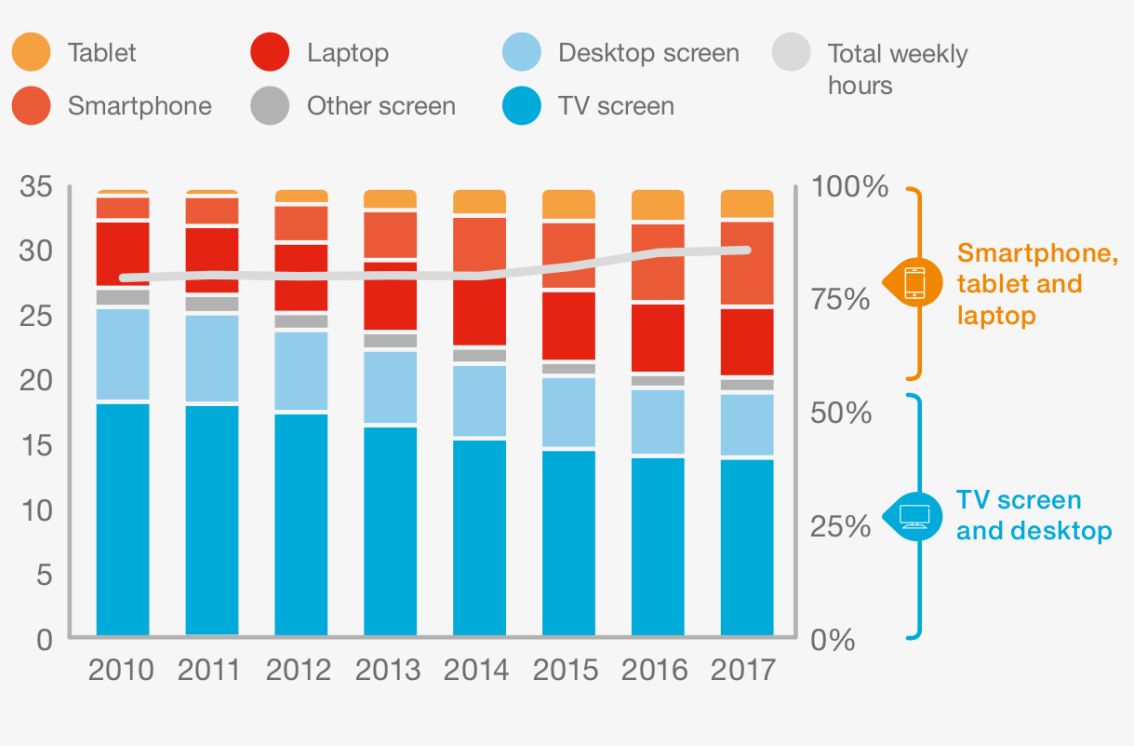


Figure 1: *Share of total TV and video viewing time per device [2]*

Chapter 1

Related Work

1.1 Quality of Service and Quality of Experience

There are several connected terms when it comes to measuring the quality of a service, especially in the telecommunications section.

From a network perspecitve, the Quality of Service (QoS) is the most commonly used term. As defined by the ITU-T, the Quality of Service is "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" [5] It's objectively measureable, but meeting the necessary QoS parameters doesn't imply that the customer's experience will be flawless when using the service. The Quality of Experience approach places the end-user in focus, as the ITU-T definition shows, QoE is: "the degree of delight or annoyance of the user of an application or service." [5] Since QoE is a rather abstract term, quite a few things - influencing factors - could be considered when assessing a user's Quality of Experience. For example "the user's cultural background, socio-economic issues, psychological profiles, emotional state" [5] also impact the QoE, not only the characteristics of the service itself. Nor the network operator, nor the service provider can influence (or even be aware of) all of these factors, but the technical issues (called System Influence Factors by [6]) may be in their domain, especially the network and media related factors. One of the most often used metrics related to QoE is the MOS (Mean Opinion Score) [7], where, during the assessment, the participants score the service (or their experience with it) on a 5-level scale. Averaging these results gives the Mean Opinion Score, a single scalar value describing the QoE. (The assessment of the MOS itself is not strictly based on subjective judgments, its variants include objective (model based) solutions).

1.2 Adaptive HTTP Video Streaming

Without diving into too much detail, it's important to know some basic characteristics of the OTT video streaming services. The adaptive streaming methods are standardised under the MPEG-DASH standard [8], which is adapted by the major service providers. [9] [10] In a nutshell, when using the MPEG-DASH, on the server side, the videos are available in small segments, containing only a few seconds of video, in different quality encodings. During playback, the client (e.g. a player in a web browser or an application on a mobile terminal) can choose from the different representations based on the network conditions, like the available bandwith, and status of the playout buffer containing the already downloaded parts. In other words this means that during the streaming of a video, the quality that it's presented with can vary, altough this can be usually controlled by the user, by asking for a fix representation. Streaming a video usually starts with a larger burst in the traffic, after that - when everything works fine - the traffic is characterized by on-off cycles, downloading the next chunks when the playout buffer starts to deplete, and silently (from the network perspective) playing back the downloaded content, when it's available. [9] This method doesn't require the usage of a certain codec, and it also allows the transmission to be encrypted or unencrypted.

1.3 Estimating Video QoE

1.3.1 The Standard for QoE

The ITU-T P.1203 [11] standard is described by [12] with an example implementation. It's important to note, that the abstract model described by the standard is most suitable in the case of non-encrypted streams. The main idea of the bitstream based assessment is to provide an estimation of what the subjective, perceived quality would be, without the actual involvement of the users. The architecture of the model from the standard can be seen on Figure 1.1.

Based on the video stream, the media parameters are extracted (like resolution, bitrate, frame rate) then the short-term video and audio predictor modules, specific to each codec - estimate the quality for short (second length) pieces of content, and this, along with the output stalling parameter extraction module is fed to the long-term quality predictor module, which produces the estimated MOS for the video stream, along with some optional diagnostic outputs. In the encrypted case however, the inputs required by the model may not be easily extractable - by a

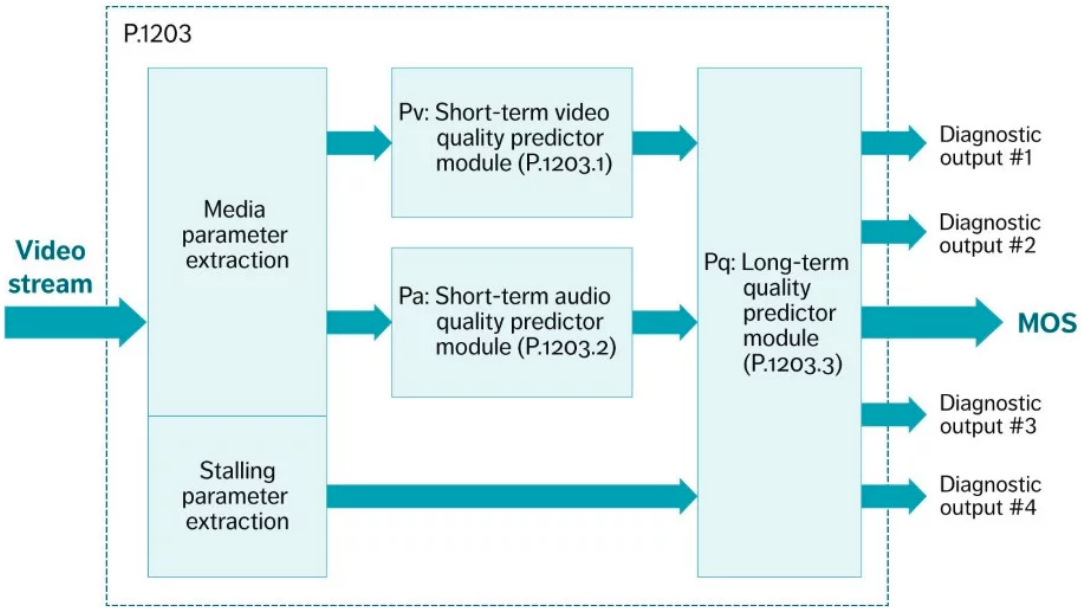


Figure 1.1: *ITU-T P.1203 architecture [12]*

network operator for example. The case is different of course for the service provider, where quality reporting mechanisms from the client are in place in many services. [12]

If active client side involvement is allowed in the evaluation, then the approach described in [13] shows a working example of using the standard model for encrypted video. The system consists of a man-in-the-middle proxy, and active probing on the client side. This enables collecting both the encrypted and the decrypted streams, because on the client side, the whole decrypted video stream is available, and as such, every input that is necessary for the P.1203 model to work can be used.

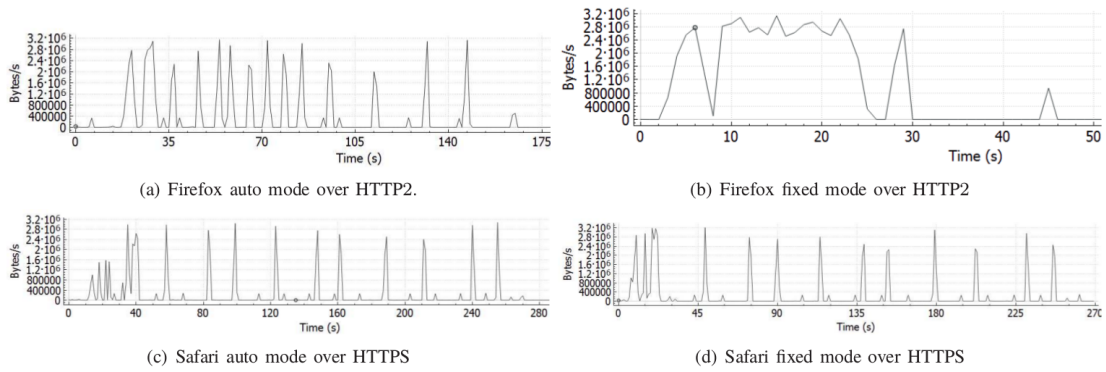
The solution that's described in the standard differs from the proposal in this thesis mainly in it's approach towards the problem, and the goal that it tries to achieve. The standards tries to give an abstract and general framework for the QoE estimation, and in this sense, offers more, than the current goal. On the other hand, my task is about a special subset of these factors, but considering the huge portion of encrypted video streams, offers a machine learning based solution, instead of a mostly analytical solution (altough the long-term predictor is partly machine learning based in the P.1203 too [11]).

1.3.2 Encrypted Streams

The standard describes a way of estimating the video Quality of Experience, if we can extract the necessary information from the stream somehow. But in the

encrypted case, even the media parameter extraction is more difficult. Partly because of the relatively new standard, and partly because of the difficulties that the input data preparation for using the model poses, alternative proposals have been made to estimate the video QoE (more) directly from the encrypted streams, including machine learning based approaches.

Many studies focus on Youtube, as the largest OTT video service provider, such as [14]. The study shows that different video streaming clients represent different traffic patterns. In the example, for Youtube videos in fixed mode (when the user selects the desired resolution) the Flash based player's traffic showed the same characteristics as in the automatic mode (a large burst at the start, and then on-off cycles), while the HTML5 based players downloaded the whole video at the start in one burst, with no additional traffic later during playback, as it can be seen on Figure 1.2.



(a) Firefox auto mode over HTTP2.

(b) Firefox fixed mode over HTTP2

(c) Safari auto mode over HTTPS

(d) Safari fixed mode over HTTPS

Figure 1.2: *Video bitstreams in different browsers, from [14]*

Otherwise, their analysis (with a focus on the Safari browser) shows, that the different quality representations have overlapping bandwith ranges, which makes classifying them more difficult. The first and last segments usually have more variance, and they may not be as relevant to the session, as the other parts, so they were not part of the classification, to reduce the error rate. The most important feature that the described classification method is based on is the bit rate (in the case of Safari, the pattern is the same when using auto or fixed mode, so the bit rate measurements and patterns from fixed mode sessions can be used to classify the auto mode sessions as well). From these features, a KNN classifier is used for classifying the segments to 3 quality representation classes even in real time, and shows promising results, though on rather small datasets, and only for Youtube.

Another approach concentrates on the mobile terminal applications([15], [16]). While focusing on estimates describing the whole session, there is also an important idea mentioned in these studies, that can be reused. If a few test devices can gather quality metrics of video sessions, while the network traffic is also monitored, then these

together can form the dataset from which a machine learning model can be trained. Taking it further, if the different client side applications or hardware specifications, or - on the server side -, if the different service providers' traffic patterns (e.g. because of different supported codecs or cypher suites during the encryption) are so distinct, that one model can't effectively classify the QoE influencing features, then the training data can be collected with these different configurations, and each of them may get a separate model during training and evaluation.

Youtube is also in the center of [17]. Non-overlapping windows of one minute length were classified, whether there was a buffering and/or bitrate adaptation event during that period, and the two most important features were found to be the download troughput, and the packet inter-arrival time from the server. Out of four algorithms (Logistic Regression, KNN, SVM, Random Forest), the Random Forest scores best on the buffering detection task (0.975 accuracy). It is worth noting that the data points containing no rebuffering are overrepresented (similarly to real world scenarios), so the results are good partly because of the model learning to predict the 'no buffering' class most of the time. Overall this is a similar approach to the one presented here later, but the one-minute resolution lacks what is necessary for the troubleshooting purposes.

The patterns of the network traffic seem to be so distinct, that it may even be possible to identify the video being played back from the network traffic, as [9] shows. Basically all of the major OTT video service providers use variable bitrate video playback, based on the contents of the video and the perceptually meaningful information, as opposed to constant bitrate methods - this saves storage space and bandwith, without degrading the quality of the video. Since the adaptive streaming has the above mentioned bursty characteristic, and the videos are stored and played with variable bit rate encoding, the size of a segment doesn't only depend on the resolution of the video, but also on the content. Based on this information, it's possible to train a machine learning model, to classify a video, whether it is one of the examples from the training set. (According to the authors, the traffic pattern is not distinctive enough for every video to be recognizable.)

Based on influencing factors, in this semester stalling was chosen because it's a main influencing factor of the QoE, and detecting whether the video playback was stalled (and when) during the session would prove useful during troubleshooting. The impact of stalling is mentioned in separate studies ([16] [12] [17]) and the ITU-T standard [11] also.

Chapter 2

Tools and Methodology

In this chapter, first one of the standard methodologies for Data Mining problems is presented, since the phases are so general, that parts of the method can also be useful for solving Machine Learning problems. A discussion about the used models and metrics follows.

2.1 CRISP-DM Methodology

The CRISP-DM (CRoss-Industry Standard Process for Data Mining) [18] model is a general approach to Data Mining problems - useful, when we want to find non-trivial patterns in the data. CRISP-DM consists of 6 phases:

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

The phases follow each other in the mentioned order, but entering a new phase during a process doesn't mean that one can not return to a previous one, as it can be seen in Figure 2.1.

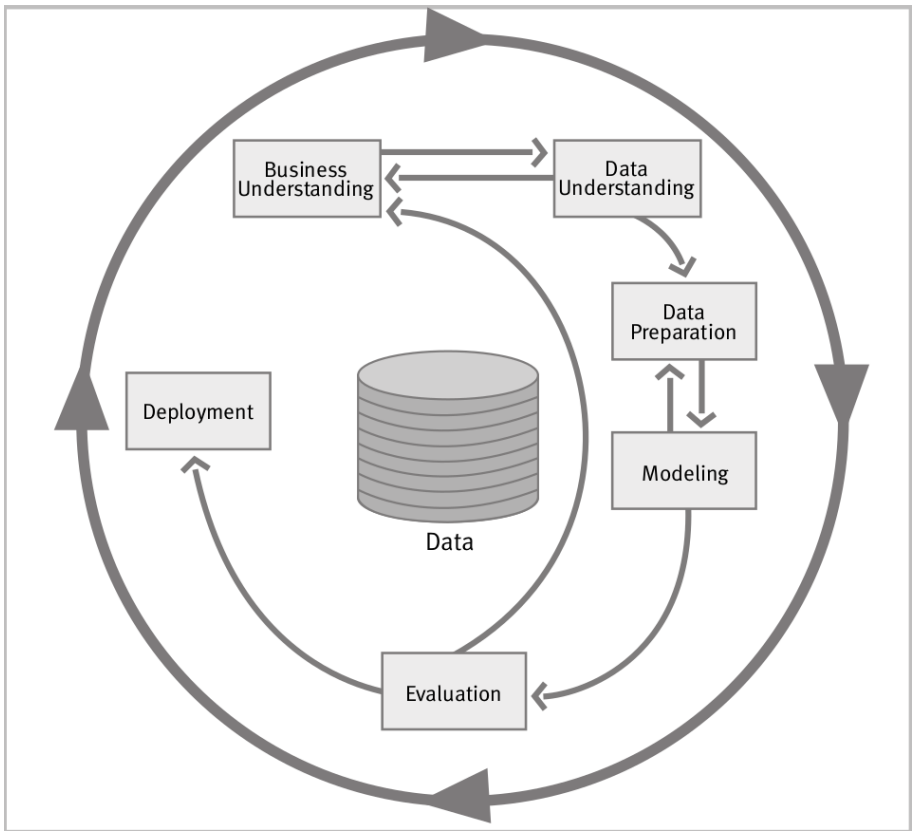


Figure 2.1: *Phases of the CRISP-DM Model[18]*

Business Understanding

The main goal in the business understanding phase is to gather (and understand) the objectives and requirements of the project from a business point of view, and to map the problems to the common data mining / data science areas and tasks. In this case, the objectives were mentioned before: A (mobile) network operator should be able to detect and troubleshoot problems occuring during video streaming sessions, based on the information that they can gather without active contributon from the end users, in order to maintain a sufficiently high service quality. The goal of troubleshooting implies that a video streaming session should be described with fine resolution information about the service quality, meaning that the main factors characterizing the session should be calculated not for the entire session, but for each piece of the session separately. So for each sufficiently small piece of a session (this parameter should be defined based on its usefulness to the operator on the one hand, and on the model's performance given the available data on the other hand. Inferring that there was a problem during the first half of the session, and everything went fine during the second half is not much better than one number for the whole session. Also, trying to give microsecond resolution is useless if the predicted numbers are wrong 99% of the time).

Predicting the stall times can be thought of as a binary classification problem: we'd like to know, whether the playback was stalled in the examined time period, thus dividing the results to two disjunct sets. [19] This is the main focus of the current semester.

It is worth mentioning, that out of the many factors that have an effect on this characteristic, only a portion are in the network operators' domain. The service providers aim to eliminate these kinds of defects from their services, but there is always a chance of degradation on the server side. On the other hand, the status of the mobile terminal can also play a role in the Quality of Experience. (In the latter case however, the network operator may be capable of mitigating the situation, by e.g. offering a subscription with a new device to the users who experience these problems.)

Data Understanding

This phase is about the collection and description of data that is available to solve the tasks defined in the first phase. This is of course not a one-way process, finding out more about the available data can result in a better understanding of the business use cases. In this case, the data can be collected and may hold information relevant to the goal are the TCP level metrics, and the mobile network events and metrics could also prove useful. Of these two, the first ones were available during this semester.

Data Preparation

In this step, the goal is to transform the raw data, so that a model can process it to solve the task. This may include filtering (e.g. NaN values or outliers), scaling (normalization or standardization), feature selection / creation of additional features, etc. The exact methods depend on the model that is used, so again, the data preparation and the modeling phase don't strictly follow each other. The methods used in this case will be detailed in section 3.

Modeling

This phase is about selecting or building the best models for the tasks. This of course leads to the specification of what we mean by "best", in other words, what are the metrics that we want to optimize for. The models also have a lot of configuration parameters, and it can vary which set of these parameters are best for achieving the goal. The relevant models and metrics are detailed in 2.4 and 2.3.

Evaluation and Deployment

In the previous step, the models were evaluated based on some metrics, while in this phase, the business use case is in focus: was it achieved and how, what are the next steps.

According to the general method, during the deployment phase, the solution goes live, and the users can benefit from it. The maintenance and monitoring of the system should be planned, the final report prepared, and the whole project reviewed and evaluated.

2.2 Data mining and Machine learning

Data mining and the CRISP-DM process as described tells the story from the perspective of the person executing its steps with a certain goal in mind, while the machine learning approach focuses more on the algorithms and the methods the problem are solved with, even though the same models may be used in both fields, it's not entirely the same.

According to the definiton of machine learning by [20]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The task and the performance metrics are described in 2.3, while the experience (i.e. the input data) is described in 3.1.

2.3 The task and the metrics

Most of the use cases can be translated to more or less traditional machine learning tasks. In this case the task is to classify the sufficiently small parts of the sequences representing the network traffic of the video sessions to two classes: the video was either playing, or the playback was stalled. This means that the models should estimate a function, which takes multiple variables as input, and outputs one variable indicating the class the example belongs to. Because of the two choices, it's a binary classification problem. There are multiple metrics available to evaluate the performance of a classifier.

A binary classifier's outputs can be categorized to four groups: If the classifier predicted the positive class for the example, and the true class of it is indeed positive,

then it's a true positive. On the contrary, if the classifier predicted negative class for a positive example, it's a false negative. If the example is negative, and the judgment of the classifier is also negative, the result is a true negative, while a positive output for a negative example is a false positive. These categories form the confusion matrix (which is also defined for multi-class classification tasks).

		True class	
		Positive	Negative
Predicted class	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

The most common metrics, *Accuracy, Precision, Recall, F-Score* are defined in terms of the mentioned categories as follows [21] :

$$Accuracy = \frac{TP + TN}{P + N}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F\text{-}Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

As it was mentioned earlier, the accuracy score may not be very verbose in the case of imbalanced or skewed classes: if the dataset consists of 99% of examples that belong to one class, and only 1% that belongs to the other class, then a "model" that doesn't actually learn anything, but always predicts the first class, would have an accuracy of 0.99. This example is an exaggeration, but it can be easily seen, that there is a need for other metrics.

The most important metric can vary on the task or the application of the models. If both precision and recall are important, the F-Score (or F1-Score) provides a way to keep both of them in mind, but there are also cases when one of them is more important, than the other (or in other words, false positives or false negatives cause the bigger problem - e.g. in a healthcare diagnostic system, a false positive could be supervised by a human expert, but a false negative could prove to be harmful to the patients). In the stalling classification case, the best would be that all of the metrics showed high scores, but if that's not possible, recall would be more important, than

precision. It's important to note however, that during training, not directly these metrics are optimized for, but a loss or error function is used to measure the progress of the model. E.g. in the binary classification case, binary crossentropy is used as the loss function (which approximates the difference between the optimal and the learned probability distributions, based on the outputs of the model) [20].

2.4 Machine learning tools and models

One of the most popular languages used in machine learning is python, and one of the main reasons of that are the frameworks and libraries, that are available to use in this area, ranging from the support of numerical computation (Numpy [22]), to Deep Learning frameworks. Another reason is of course, that since it's a high level language, it allows fast development and experimentation.

The main frameworks that were used in this semester were Scikit-Learn [23] as a general machine learning framework, Tensorflow [24] and its high level API Keras [25] for neural networks and deep learning. Scikit Learn provides a plenty of useful functionality not only for modeling, but also for data preprocessing.

A baseline model was chosen, so that the performance of other models and their configurations can be measured and compared to the results of a simpler model. The models used were the following:

- Logistic Regression [26] as a baseline
- Random Forest [27] as a popular, traditional model for reference, also used in [17]
- Recurrent Neural Networks (LSTMs [28],[29] in particular) as the default choice of neural network architectures for sequence processing [19]

Logistic Regression

The baseline model was a Logistic Regression model [26], using the default parameters from Scikit-learn [23]. It is a fairly simple and widely used algorithm, especially as a baseline. It's based on the logistic (or sigmoid) function of the input variables weighted by the model parameters.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Random Forests

Random forests are based on decision trees, with the goal of mitigating overfitting. It's an ensemble learning method, and as such it constructs multiple decision trees, and uses an aggregation of the individual trees' predictions as the output of the ensemble (e.g. the mode of the class predictions at classification tasks). In a nutshell, the decision trees in a random forest use only a subset of the input features. This set is chosen using the random subspace method [27]. Additionally the training examples are sampled with the bagging method, so the output comes from the aggregation of the result of many shallow trees, trained on slightly different samples. If a feature is an important predictor of the target variable, it will be chosen in many of the trees (so constructing a random forest inherently measures feature importance). Random Forests can be trained relatively fast, show promising results and are among the most popular algorithms on the Data Science / Machine Learning competition site, Kaggle ([30], [31]).

Neural Networks

This isn't intented to be an in-depth look at neural networks, just a short overview. Neural networks consist many of neurons, each being a simple mathematical unit. [19] A neuron can take multiple inputs, computes a weighted sum of the inputs, and applies an activation function (such as the sigmoid mentioned before), producing an output value. The inputs to a neuron can be the outputs of other neurons, thus forming a network. These neurons are grouped to form layers - there are no connections between neurons inside a layer, the inputs can be connected to the outputs from the previous layer, and current layers outputs can be used as input to a successive layer. The number of layers gives the depth of the network (the term deep learning refers to this). The weights are tuned during training, using a method called backpropagation of errors and gradient descent (or a similar optimization function) [19].

In the simplest case, each neuron of a previous layer is connected to all of the neurons of the previous layer (also called a Fully Connected or Dense layer). Another type of neural networks, that is commonly used when processing sequential data is the recurrent neural network (RNN), and its variants.

RNNs are based on the idea of recurrent connections: that is, a neuron may receive it's own output (produced by feeding it an example) as an input (when processing the next example). This makes it an intuitive candidate for sequences: the output

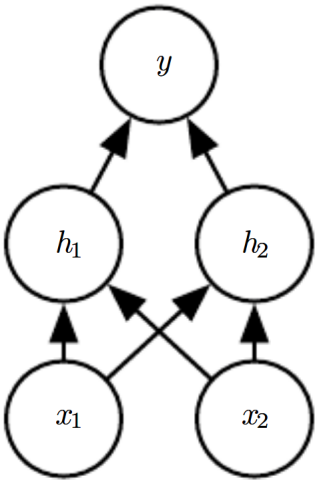


Figure 2.2: *Example representation of a neural network [19]*

depends on the previous timesteps (or rather, on a state derived from them), capturing the context of the current example this way. The networks have different applications based on the output they are expected to produce, e.g. yield an output value for each input step, or only one output for the whole sequence. Out of these examples we currently use the latter, as it can be seen on Fig. 2.3.

The LSTM (Long-Short Term Memory) networks are an improved recurrent network architecture, that try to solve (or mitigate) the vanishing gradient problem (which refers to the weight updates calculated from the gradients getting smaller -eventually vanishing - with the increasing depth of the networks). Additionally, LSTMs attempt to solve the problem of long-term dependencies in sequences (the distance between the relevant information and it's actual usage can become large), by managing what information should be kept and updated during training. "The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation." [32] The gates controlling the information flow are the forget gate (determining what parts, and in what percentage of the previous state do we want to keep, compared to the new inputs), the input gate (which values do we want to update) and the output gate (based on the state, what information should the LSTM output). These additions proved beneficial [19], but the gates add more parameters to the models, and depending on the task, this might lead to overfitting, described in section 3.2.1.

A simple LSTM based architecture was chosen: an LSTM layer receiving sequences of a maximal length, and with the default activation functions (sigmoid at the gates and tanh at the output), and a fully connected layer, consisting of one neuron, and

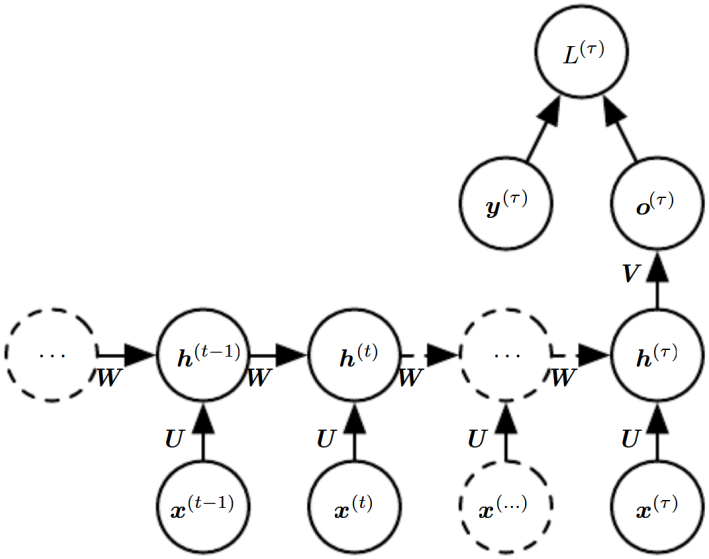


Figure 2.3: *Recurrent neural network producing a single output for a sequence. L is the loss, calculated from the ground truth value (y) and the output of the model (o) [19]*

a sigmoid activation, since the task is binary classification.

The LSTM model was trained with the RMSProp optimizer and the mentioned binary cross entropy error function. [25] To avoid overfitting, a validation set was separated to measure the approximate performance after each epoch, and early stopping was used - this means that regardless of the number of epochs specified at the start, the training stops, if a parameter (in this case the validation loss) isn't improved anymore (in this case, decreased during the last 20 epochs). This is complemented by model checkpointing, which saves the best model during training (because the overfitting has already happened, when we notice it when evaluating the early stopping step). Regularization methods also help mitigating overfitting: with the LSTM model dropout and recurrent dropout techniques were used. Using dropout means that during each epoch, a random subset of the neurons are masked, leaving them out temporarily. Recurrent dropout is the same idea extended to the recurrent connections. [25]

Chapter 3

Contributions

3.1 The Dataset

The provided dataset consists of transport layer (TCP) metrics from the network of a major mobile network operator collected by passive probing/monitoring, along with reference files describing the video sessions on the client side, e.g. what was the actual bitrate, and when the presentation was stalled.

The TCP metrics are reported in records describing a maximum of 30 seconds, and contain aggregated values on one second granularity - this means that for example, the bytes downloaded in a second are summed and reported as one number, and a record contains a maximum of 30 'downloaded bytes' values. The record also contains information about the OTT service provider, type of the protocol (in practice, QUIC based records were not collected), and the timestamp when it was created. The one second granularity in the input also sets a lower bound to the finest granularity the model could predict the status of stalling. On one hand this already would be a lot more detailed information, than the previous attempts providing one number per session or per minute, and on the other hand, gathering even more detailed information would be difficult (e.g. because of timing issues), and may not yield as much value.

Based on the assumption (to be proved or rejected later), that the traffic of video playback differs for each service provider, they were processed separately - and in this semester, only one of them (Facebook) was chosen. There were 34 sessions available for this provider, out of which 25 contained stalling.

The collected data includes the number of downloaded and uploaded bytes, number of active connections, various events such the establishment, close or timeout of a connection, and statistical information about the distribution of active and waiting

periods in that second (i.e. how much time during that second was spent waiting and how much with active transmissions).

Preprocessing

The first type of input files is the one holding the records (in JSON format). From these, the metrics are extracted forming ordered time series in a tabular form: the rows represent the seconds of the session in order, and for each row, the column values are the corresponding metrics (or zeros if the value is missing). This also means that these input variables are numerical, which is the preferred case with most of the models. The number of event types available is limited, so a column is added for each of them, and the value of the column is the number of such events in that second.

Additionally, since the time series based on the input records may not be continous - some periods may be missing due to the logic of the data collector module (e.g. when there is no traffic, no record will be created) - the missing gaps are filled by full-zero rows. This helps when the ground truth values are calculated. After these preprocessing steps, the matrices can be persisted to disk. The metrics extracted from the record files form the input variables to the machine learning models. With supervised learning, the models need a target variable (or label) too - in this case, a column containing zeroes, when there was no stalling, and ones when there was -, which are extracted from the reference files. These files contain the necessary information, in a shorter (but strict) form: a row describing these events starts with a keyword (e.g. 'buffering') and the interval in seconds when it happened (in presentation time). This means that the ground truth values can be calculated: with the same number of rows as in the input feature files, a row gets a 'one' label, if it's in an interval included in the reference file, and 'zero' otherwise.

After the input and target variables are available, additional data preparation steps are necessary, most importantly the scaling or normalization of the data. The different input variables are on different scales: e.g. the typical values of downloaded bytes and active connections differ by orders of magnitude. A lot of models require that their inputs are on the same scale, and work best when every input is in one interval (typically between 0 and 1, or -1 and 1). Because of this, all inputs are normalized. [23]

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This is a relatively simple task, but it's important to keep in mind, that a part of the dataset should be separated and not used during training, only for evaluation,

simulating the behaviour of the model for data it hasn't seen before (like in a real deployment). Since the test set is separate, it also shouldn't be included in the calculation of minimal and maximal values for normalization. However, the variables of the test set should also be scaled according to the values extracted from the training set. The training and testing set were split on a session basis, so all datapoints of a session are either in the training or the testing set. (Another choice would have been to split on a datapoint, or subsequence basis, so regardless of which session a subsequence belongs to, some parts were put to the training and some to the testing set). This approach was chosen to better simulate the real world scenario, with fully unknown examples that the model would have to deal with after deployment. The preprocessing and modeling pipeline can be seen on Fig. 3.1.

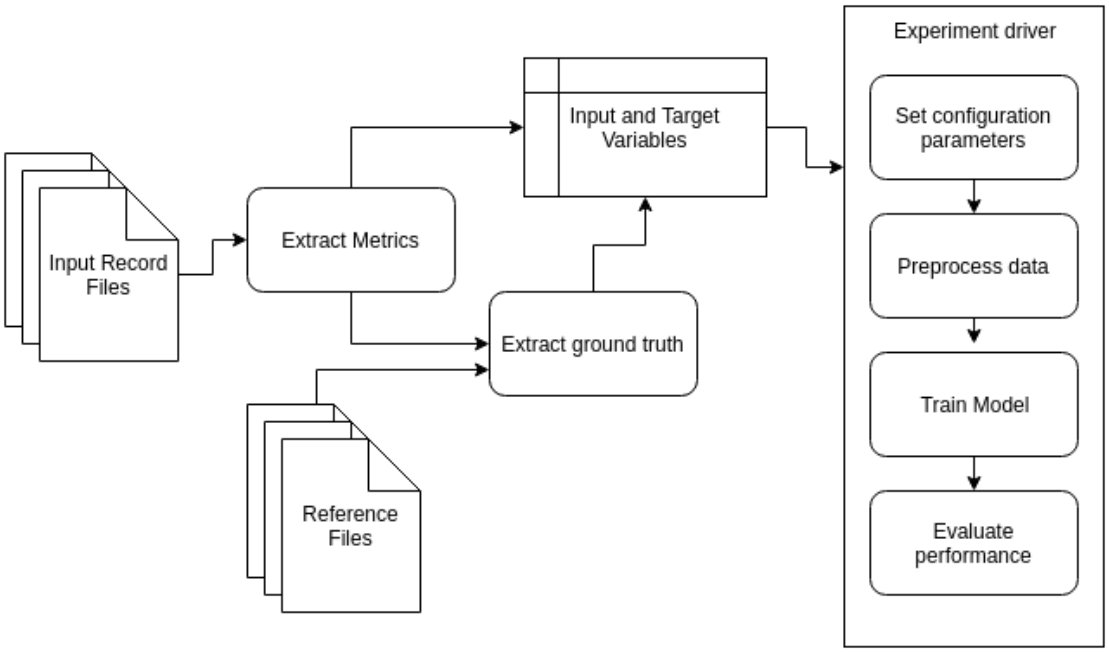


Figure 3.1: *The preprocessing and modeling pipeline described in chapter 3.1. The input and target variables can be stored and reused for experimenting with different configurations and models.*

3.2 Input formulation

The finest granularity, that the models can hope to achieve with the stalling classification based on the available input data, is one second. But a one-to-one classification in this case wouldn't lead to good results - the metrics from one second are not clearly indicative of the target variable. E.g. the number of downloaded bytes can be zero because the playout buffer is full, and there is no need for additional data

at the moment, or it can be zero because the connection timed out in the previous second. In other words, the context (the sequence or a part of it) is necessary to infer the target. There are different solutions to handling the context, some of them were experimented with.

The 'traditional' models, logistic regression and random forest can't handle this by default - however, the input can be reshaped to match the tabular form these models expect. This means that the variables of the previous (or successive) examples can be added as additional columns to the current row, resulting in wider matrices.

Another way that the models (such as RNNs) might be able to process the input data, is if it's in a 3D tensor form (the time being the third dimension). This means, that one example is a sequence, where each point in the sequence has multiple input variables, and the whole dataset consists of many such sequences.

3.2.1 Influencing factors

There are a number of different configurations and factors that may impact the results and finding the best will require experimentation. The results published are preliminary, and should be considered as proof-of-concept, they are included to show that the approach is feasible, the actual performance can't be measured until more training data is available (if we wish to avoid introducing closed world assumptions [9]). Until that it's possible that the models overfit the current training data (the model learns the characteristics special to the training examples, demonstrating low error rates (only) on the training set, and doesn't generalize well to other data [26]). An illustration of overfitting can be seen on Fig. 3.2.

Context direction and size

In this case, the direction is meant in time: what should the context of an example include? Only previous examples, or successive examples also? How many neighbouring examples are necessary to infer the target variable? So far only the previous examples were considered, but the cardinality of the context was experimented with. It is also worth noting, that if the previous data points are used, then the first examples have to be dealt with, since the history is not available there. Two ways of solving this problem are: padding the first subsequences with zeros (or other values); or simply exlcuding these examples from the processing.

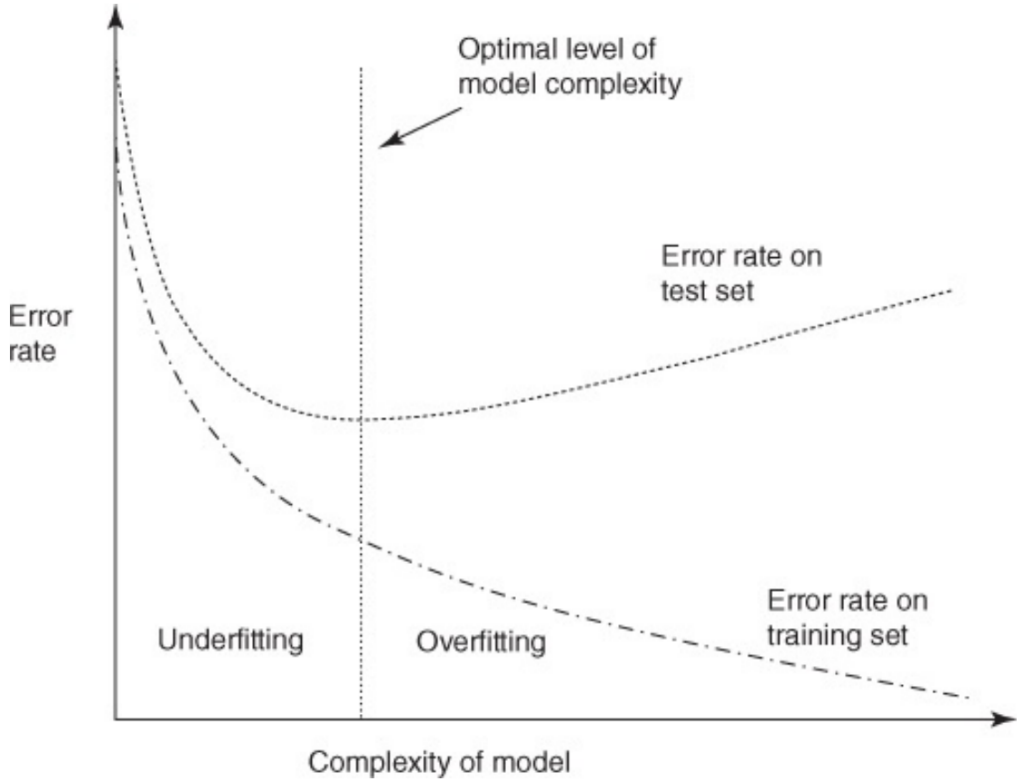


Figure 3.2: *Overfitting [26]*

Overlapping or Non-Overlapping subsequences

The context windows or subsequences can overlap (in a sliding window fashion), or it's also possible to use disjunct, non-overlapping windows. The latter approach implies that the prediction resolution is equal to (or less than) the size of the window, i.e. if we use 10 second disjunct windows, we can predict whether there was stalling somewhere during that 10 second period. With the sliding window method we add the datapoints of e.g. the last 15 seconds to an example during the preparation, then move the sliding window by a unit. The unit by which the window is shifted (the stride) also specifies the granularity of the predictions (a 2 step stride means one prediction for every 2 seconds), and if the stride reaches the window size, the results are disjunct windows again.

Imbalanced classes

Since most of the time the users don't experience stalling during the video sessions, the number of positive targets is much lower, than the negative ones. There are various ways to try to solve this issue (e.g. undersampling of the more populated class or undersampling of the one with less examples, etc.) in this case the most

straightforward to use was the weighting of the classes: during training, the examples with positive labels got a larger weight on parameter updates, while the negative examples got smaller weights.

During preprocessing, the sequences were filled up to be continous. However, this is not necessary, and leaving out long parts of full zero rows might help in balancing classes. (It's also possible to calculate the ground truth values in another way, and not to fill the missing values in the sequences at all).

3.3 Early Results

Three tasks were defined based on the resolution of the predictions:

- The most detailed information would come from a one second resolution classifier, which makes it a natural choice
- 5 second resolution as the middle ground between 1 and 10
- 10 second resolution, which would still be a lot more detailed than earlier methods, but in theory would be an easier task to achieve

The tables show the best results from three attempts.

Model	Accuracy	Recall	Precision	F1-Score
Logistic regression	0.84	0.37	0.40	0.39
Random Forest	0.92	0.42	0.88	0.57
LSTM	0.94	0.65	0.85	0.73

Table 3.1: *One second resolution, with 30 second time windows*

An interesting phenomenon is that while one might expect, that inferring the presence of stalling in a 10 second window is easier, than detecting the same for every second, currently it's not the case. Since the number of examples is limited, as we increase the window for which to do the predictions, the examples the model can be trained with decreases. (I.e. the summed length of all video session in the dataset is about 12000 seconds. That means 12000 examples for the second granularity classifiers, but only 1200 examples for the 10 second resolution classifiers.

When evaluating the results, it's important to keep in mind, that the classes are imbalanced, with the 'no stalling' class vastly overrepresented. The accuracy scores reflect this: even the Logistic Regression model has achieved over 70% accuracy on all three tasks, due to the model effectively learning the zero class representation.

Model	Accuracy	Recall	Precision	F1-Score
Logistic regression	0.85	0.08	1.0	0.15
Random Forest	0.90	0.42	0.92	0.58
LSTM	0.94	0.68	0.91	0.78

Table 3.2: *5 second resolution, with 30 second time windows*

This model wouldn't be of much use in practice though, and this is most apparent on Table 3.2. The logistic regressor achieved a Recall score of 0.08, meaning that it only labeled a small percentage of the 'True' examples correctly - but the rows marked as positive were all True Positives, as it is indicated by the precision score.

Model	Accuracy	Recall	Precision	F1-Score
Logistic regression	0.72	0.44	0.41	0.42
Random Forest	0.75	0.37	0.47	0.41
LSTM	0.88	0.5	0.86	0.63

Table 3.3: *10 second resolution, with 30 second time windows*

In all three tasks, the LSTM model performed best by the accuracy, recall and F1-score metrics. These results support the popular understanding that the recurrent neural networks work well when dealing with sequential data, but since the available dataset is small it's too early to conclude which method works best. It's worth mentioning, that the Random Forest model also performed well, while being much faster to train.

However, there are a plenty of other neural network model architectures and parameters, for example the vastly different Convolutional Neural Networks (CNNs) seem to be worth experimenting with, as [33] suggests that convolutional network architectures achieve comparable results in sequence processing tasks to RNNs.

Chapter 4

Conclusion

The Quality of Experience approach is user centric, and measures the condition of a service by the satisfaction of the end user. In the context of video streaming services, some of the main influencing factors were described such as the stalling ratio during presentation. The quality of an OTT service is not only important for the service provider, but also for the network operators, and the feedback that a QoE estimation provides is particularly useful, when it can be used for troubleshooting. This requires fine resolution information about when the impairments happen. However, with the spread of transport layer encryption among the video service providers, this is no simple task, and can't be solved effectively in an analytical way. During this semester, after summarizing the related work, a machine learning based attempt was presented to estimate stalling of the videos, based on transport layer metrics collected from a network of a major network operator, along with the video captures of the sessions, that were used as reference. A data preparation and modeling pipeline was written that enables straightforward experimentation with different models and configurations. 3 models were trained for 3 different tasks, predicting different resolution information. Evaluating the results shows that the approach is promising, but the early results don't allow far reaching conclusions.

4.1 Future plans

For the next semester and the full master's thesis, there are still ideas to cover and tasks to be done; this section attempts to give an overview of them.

Different models for different service providers

The service providers may use different encryption methods, video codecs or different streaming parameters, and correctly classifying them may require different models.

Improvement of the current solution

The current solution is promising, but there is still room for improvement: the effect of additional data (measurements in a magnitude of hundreds are expected), and additional features (either from transport layer metrics or events from the radio access network) may open up more possibilities to e.g. derive which features are the most important in the question video streaming of quality, find out further correlations, and to come to a practically applicable solution, which generalizes more to previously unseen examples. This may require tuning the first models and experimenting with different model architectures, like CNNs.

Other QoE influencing factors

In this semester, the prediction of stalling was in focus, but it's worth attempting to infer other influencing factors of the video Quality of Experience, like the image resolution or bitrate the video was presented with, the changes of the quality representation, or the initial buffering time at the start of the sessions.

QUIC

Currently, TCP metrics were used as input to the models. But QUIC operates over UDP and also handles the encryption differently from TCP [34]. This means that the same methods are probably not transferable without changing them. This point depends on the availability of input data for QUIC.

Deployment and Continous Learning

As the video codecs, encryption algorithms, etc. and their parameters can be changed, and as they evolve over time, the originally trained models' performance may degrade. Exploring the ideas of (partially) retraining the models to fit the new environment would be useful.

Bibliography

[1] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020 white paper," *White paper, CISCO*, 2016.

[2] A. E. Consumer and I. I. Report, "TV and media 2017." `https://www.ericsson.com/en/trends-and-insights/consumerlab/consumer-insights/reports/tv-and-media-2017`. Accessed: 2018.05.03.

[3] Google, "Youtube engineering and developers blog: Youtube's road to HTTPS." `https://youtube-eng.googleblog.com/2016/08/youtubes-road-to-https.html`, August 2016. Accessed: 2018.05.03.

[4] Google, "Google webmaster central blog: Https as a ranking signal." `https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html`, August 2014. Accessed: 2018.05.03.

[5] ITU-T, "ITU-T P.10/G.100, vocabulary for performance, quality of service and quality of experience," November 2017.

[6] U. Reiter, K. Brunnström, K. De Moor, M.-C. Larabi, M. Pereira, A. Pinheiro, J. You, and A. Zgank, "Factors influencing quality of experience," in *Quality of experience*, pp. 55–72, Springer, 2014.

[7] ITU-T, "ITU-T P.800.1, mean opinion score (MOS) terminology," July 2016.

[8] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, pp. 62–67, April 2011.

[9] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *USENIX Security*, 2017.

[10] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video QoE from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 513–526, ACM, 2016.

[11] ITU-T, "ITU-T P.1203, parametric bitstream-based quality assessment of progressive download and adaptive audiovisual streaming services over reliable transport," October 2017.

[12] J. G. Gunnar Heikkilä, "Video QoE: leveraging standards to meet rising user expectations," *Ericsson Technology Review*, 2017.

[13] S. Göring, A. Raake, and B. Feiten, "A framework for QoE analysis of encrypted video streams," in *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*, pp. 1–3, IEEE, 2017.

[14] R. Dubin, A. Dvir, O. Pele, O. Hadar, I. Richman, and O. Trabelsi, "Real time video quality representation classification of encrypted HTTP adaptive video streaming-the case of safari," *arXiv preprint arXiv:1602.00489*, 2016.

[15] I. Orsolic, L. Skorin-Kapov, and M. Suznjevic, "Towards a framework for classifying youtube QoE based on monitoring of encrypted traffic,"

[16] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "Youtube QoE estimation based on the analysis of encrypted network traffic using machine learning," in *Globecom Workshops (GC Wkshps), 2016 IEEE*, pp. 1–6, IEEE, 2016.

[17] A. B. Chebudie, "Monitoring of video streaming quality from encrypted network traffic: The case of youtube streaming." `https://www.researchgate.net/publication/317570242_Monitoring_of_Video_Streaming_Quality_from_Encrypted_Network_Traffic_The_Case_of_YouTube_Streaming`, 2016. Accessed: 2018.04.20.

[18] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "CRISP-DM 1.0 step-by-step data mining guide," 2000.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[20] T. M. Mitchell *et al.*, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[21] T. Fawcett, "An introduction to ROC analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[22] O. Travis E, *A guide to NumPy*. Trelgol Publishing, 2006.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine

learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[25] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015. Accessed: 2018.05.09.

[26] D. T. Larose and C. D. Larose, *Data mining and predictive analytics*. John Wiley & Sons, 2015.

[27] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[30] "Kaggle." `https://www.kaggle.com`. Accessed: 2018.05.05.

[31] A. Goldbloom, "What algorithms are most successful on kaggle?." `https://www.kaggle.com/antgoldbloom/what-algorithms-are-most-successful-on-kaggle/notebook`, 2016. Accessed: 2018.05.05.

[32] C. Olah, "Understanding LSTM networks." http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2018.05.08.

[33] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[34] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pp. 183–196, ACM, 2017.