Monitoring of Video Streaming Quality from Encrypted Network Traffic

The Case of YouTube Streaming

Abiy Biru
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with Emphasis in Telecommunication Systems. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Abiy Biru Chebudie
E-mail: abch14@student.bth.se

External advisor:
Dr. Junaid Shaikh
Experienced Researcher
Ericsson AB
Lulea, Sweden

University advisor:
Prof. Dr. Engr. Markus Fiedler
Department of Communication Systems

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
ABSTRACT

The video streaming applications contribute to a major share of the Internet traffic. Consequently, monitoring and management of video streaming quality has gained a significant importance in the recent years. The disturbances in the video, such as, amount of buffering and bitrate adaptations affect user Quality of Experience (QoE). Network operators usually monitor such events from network traffic with the help of Deep Packet Inspection (DPI). However, it is becoming difficult to monitor such events due to the traffic encryption. To address this challenge, this thesis work makes two key contributions. First, it presents a test-bed, which performs automated video streaming tests under controlled time-varying network conditions and measures performance at network and application level. Second, it develops and evaluates machine learning models for the detection of video buffering and bitrate adaptation events, which rely on the information extracted from packets headers. The findings of this work suggest that buffering and bitrate adaptation events within 60 second intervals can be detected using Random Forest model with an accuracy of about 70%. Moreover, the results show that the features based on time-varying patterns of downlink throughput and packet inter-arrival times play a distinctive role in the detection of such events.

Keywords: Quality of Experience, Machine Learning, Encrypted video traffic classification, Video Streaming Quality.
ACKNOWLEDGMENT

First and foremost, I praise my Lord, the almighty God for his mercy, love and blessings throughout my life. He was faithful in all the ups and downs that I passed through. Trusting in him was and will be the greatest treasure of my life.

My sincere gratitude goes to my university supervisor Prof. Dr. Engr. Markus Fiedler for his guidance and all the interesting discussions we have during the course of the thesis work. His research experience and inspiring ideas were real support for me in those couple of months. Working under his supervision was the most rewarding experience of my life.

I want to extend my deepest appreciation to my external supervisor Dr. Junaid Shaikh for his support and advice throughout the thesis work. He gave me a valuable guidance while building the lab setup, analyzing the data and writing the thesis document.

I am very much grateful to Jörgen Gustafsson, Gunnar Heikkilä and David Lindegren at Ericsson Lulea research office for their critical review of my work and expertise comments during the various internal presentations.

Words cannot express how grateful I am to my lovely wife Tigist Teshome. Your continuous support, encouragement, prayer, understanding and unwavering love was pivotal for all the success in my life. The first four months of the thesis work which were accompanied by your presence at Lulea were the most memorable times of my life.

I owe special appreciation to my family members. Without their support, this journey would have been impossible. First of all, I would like to thank my father and mother for their unlimited support in my education since childhood. Your continuous prayer for me is the reason for all my success. I am very much grateful to my brother Tadesse. If it were not for your support, I would not have reached at this point. You were my great role model in all the journeys of my life. I also want to extend my special thanks to my kind sister Meseret. I Cannot thank you enough for your kindness, support and prayer. You always have special place in my life.

My deepest appreciation goes to Yonas Tesfaye and his wife Lelena Zerihun for receiving me with open arms and supporting me in my stay at Lulea. Had it not been for your kindness, my stay at Lulea would have been harder.

Finally, I would like to extend my sincere acknowledgement to the Swedish Institute (SI) for the generous scholarship which financed my university tuition fees and my accommodation expenses.
CONTENTS

ABSTRACT ....................................................................................................................... I

ACKNOWLEDGMENT ....................................................................................................... II

CONTENTS .......................................................................................................................... III

LIST OF FIGURES .............................................................................................................. V

LIST OF TABLES ................................................................................................................ VI

ABBREVIATIONS .............................................................................................................. VII

1 INTRODUCTION ............................................................................................................. 1
   1.1 MOTIVATION ........................................................................................................... 2
   1.2 AIM AND RESEARCH QUESTIONS ...................................................................... 2
   1.3 CONTRIBUTION .................................................................................................... 2
   1.4 STRUCTURE OF THE DOCUMENT ..................................................................... 3

2 BACKGROUND .............................................................................................................. 4
   2.1 QUALITY OF EXPERIENCE ............................................................................. 4
   2.2 BUFFERING AND BITRATE ADAPTATION ..................................................... 4
   2.3 TYPES OF VIDEO STREAMING ......................................................................... 4
   2.4 YOUTUBE VIDEO STREAMING TRANSPORT PROTOCOLS ....................... 5
   2.5 MACHINE LEARNING ...................................................................................... 5
      2.5.1 Data Set ...................................................................................................... 6
      2.5.2 Training Set and Test Set ......................................................................... 6
      2.5.3 Categories of Machine Learning ............................................................ 7
      2.5.4 Logistic Regression ................................................................................ 7
      2.5.5 Nearest Neighbor ................................................................................... 8
      2.5.6 Support Vector Machine ....................................................................... 8
      2.5.7 Ensemble Methods ................................................................................. 8
      2.5.8 Decision Trees ........................................................................................ 8
      2.5.9 Random Forest ....................................................................................... 9
      2.5.10 Model Evaluation Metrics .................................................................. 9
      2.5.11 ROC Curve ............................................................................................ 10
      2.5.12 K-Fold cross validation ...................................................................... 11
      2.5.13 Scikit-learn .......................................................................................... 11

3 RELATED WORK .......................................................................................................... 12

4 EXPERIMENTAL METHODOLOGY .......................................................................... 14
   4.1 EXPERIMENTAL SETUP .............................................................................. 14
   4.2 BROWSER AND CLIENT SCRIPT ............................................................... 14
      4.2.1 onStateChange event listener .............................................................. 14
      4.2.2 OnPlaybackQualityChange event listener .................................... 15
   4.3 CONTROLLER AND SHAPER SCRIPT ..................................................... 15
   4.4 TRAFFIC SHAPING ....................................................................................... 17
      4.4.1 Token Bucket Filter ............................................................................. 17
      4.4.2 Hierarchical Token Bucket Filter ..................................................... 17

5 DATA ANALYSIS .......................................................................................................... 19
   5.1 DATA STATISTICS ........................................................................................ 19
   5.2 FEATURE CALCULATION .......................................................................... 21
      5.2.1 Download Throughput Over One Second (\( \Delta R \)) ....................... 21
      5.2.2 Download Throughput Over 10 Seconds (\( \Delta R \)) ......................... 21
      5.2.3 Change in download throughput over one second (\( \Delta R \)) .............. 22
LIST OF FIGURES

Figure 2-1 Datasets in machine learning ................................................................. 6
Figure 2-2 Tree structure ......................................................................................... 9
Figure 2-3 ROC Curve .............................................................................................10
Figure 4-1 Experimental setup ..............................................................................14
Figure 4-2 Interaction sequence diagram between controller, Shaper, client and  
  browser script .................................................................................................16
Figure 4-3 CDF of bitrates applied by the traffic shaper ..................................18
Figure 4-4 CDF of durations for which the traffic shaper applies the bitrate .......18
Figure 5-1 Video quality statistics .......................................................................19
Figure 5-2 Buffering/Bitrate adaptation statistics ..............................................20
Figure 5-3 Flow of events in a video ....................................................................20
Figure 5-4 Statistics of buffering/bitrate adaptation events in the one-minute window 21
Figure 5-5 Representations of download throughput over 10 second values .......22
Figure 6-1 Model development steps ...................................................................24
Figure 6-2 Statistics of detected/missed re-buffering events .........................28
Figure 6-3 Performance ratings in detecting re-buffering events .....................29
Figure 6-4 Feature importance for random forest ..............................................29
Figure 6-5 ROC curve of the model developed using random forest .............30
Figure 6-6 Statistics of detected/missed re-adaptation events .........................32
Figure 6-7 Performance ratings in detecting re-adaptation events .................32
Figure 6-8 Feature importance for random forest ..............................................33
Figure 6-9 ROC curve of the model developed using random forest .............33
Figure 7-1 Rs Cumulative distribution function ....................................................35
Figure 7-2 10Rp6 Cumulative distribution function .............................................35
Figure 7-3 10Rc6 Cumulative distribution function .............................................36
Figure 7-4 Inter-arrival time 90 percentile cumulative distribution function ......36
Figure 7-5 Inter-arrival time 75 percentile cumulative distribution function ......36
Figure 7-6 10Rp4 Cumulative distribution function .............................................37
Figure 7-7 10Rp5 Cumulative distribution function .............................................37
Figure 7-8 10Rp6 Cumulative distribution function .............................................38
Figure 7-9 10Rc1 Cumulative distribution function .............................................38
Figure 7-10 10Rc2 Cumulative distribution function ............................................38
Figure 7-11 10Rc3 Cumulative distribution function ............................................38
Figure 7-12 10Rc4 Cumulative distribution function ............................................39
Figure 7-13 10Rc5 Cumulative distribution function ............................................39
Figure 7-14 10Rc6 Cumulative distribution function ............................................39
Figure 7-15 Rs Cumulative distribution function ...............................................40
Figure 7-16 Δ1R 75 Percentile cumulative distribution function .......................40
LIST OF TABLES

Table 2-1 YouTube video quality levels.................................................................5
Table 2-2 Confusion matrix.....................................................................................10
Table 2-3 Model evaluation metrics.......................................................................10
Table 5-1 Resolutions for the observed video qualities.........................................19
Table 6-1 Features for detection of re-buffering......................................................25
Table 6-2 Features for detection of re-adaptation....................................................25
Table 6-3 Parameters tuned for the learning algorithms.........................................26
Table 6-4 Confusion matrix of Logistic Regression.................................................27
Table 6-5 Classification report of Logistic Regression.............................................27
Table 6-6 Confusion matrix of KNN......................................................................27
Table 6-7 Classification report of KNN..................................................................27
Table 6-8 Confusion matrix of support vector machine.........................................28
Table 6-9 Classification report of support vector machine.....................................28
Table 6-10 Confusion matrix of random forest......................................................28
Table 6-11 Classification report of random forest...................................................28
Table 6-12 Confusion matrix of random forest for equalized target values .........30
Table 6-13 Classification report of random forest for equalized target values .......30
Table 6-14 Confusion matrix of logistic regression...............................................31
Table 6-15 Classification report of logistic regression.............................................31
Table 6-16 Confusion matrix of KNN....................................................................31
Table 6-17 Classification report of KNN.................................................................31
Table 6-18 Confusion matrix of support vector machine.......................................31
Table 6-19 Classification report of support vector machine...................................31
Table 6-20 Confusion matrix of random forest......................................................31
Table 6-21 Classification report of random forest..................................................32
Table 6-22 Confusion matrix of random forest for equalized target values ..........33
Table 6-23 Classification report of random forest for equalized target values .......34
# Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>CSP</td>
<td>Communication Service Provider</td>
</tr>
<tr>
<td>DPI</td>
<td>Deep Packet Inspection</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>GB</td>
<td>Giga Byte</td>
</tr>
<tr>
<td>HAS</td>
<td>HTTP Adaptive Streaming</td>
</tr>
<tr>
<td>HTB</td>
<td>Hierarchical Token Bucket</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union</td>
</tr>
<tr>
<td>Kbps</td>
<td>Kilobit Per Second</td>
</tr>
<tr>
<td>KNN</td>
<td>K Nearest Neighbor</td>
</tr>
<tr>
<td>Mbps</td>
<td>Megabit Per Second</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>OTT</td>
<td>Over the Top</td>
</tr>
<tr>
<td>Qdisc</td>
<td>Queuing Discipline</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QUIC</td>
<td>Quick UDP Internet Connection</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for Comment</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>TBF</td>
<td>Token Bucket Filter</td>
</tr>
<tr>
<td>TCP</td>
<td>Transport Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

The size of the IP traffic that is traversing the Internet is showing a tremendous increment every year. By the end of 2016, the amount of IP traffic on the Internet will be above the zettabyte (1000 Exabyte) threshold. Globally, by 2018, the monthly and yearly size of IP traffic will reach to 1.6 zettabytes and 131.9 Exabyte, respectively [1]. IP video traffic contributes the highest share for such high increment of IP traffic. An individual would need to spend 5 million years to watch the whole video that is traversed through the IP network globally each month by 2018. IP video traffic will have a 79 percent share of the IP traffic over the internet by 2018, which were only 66 percent by 2013 [1]. Most of the bandwidth of the Internet is consumed by third-party video applications that run over-the-top (OTT) of a communications service provider’s (CSP) transport layer [2].

Two changes are observed on the behaviour of the consumer due to the fact that the Internet is dominated by traffic from these kinds of videos: the first one is that since such videos have high bandwidth utilization, high levels of peak bandwidth is observed, and the second is that consumers have become more sensitive to video quality changes. When there is a congestion on the network resources at the time of high network utilization, the users interpret this congestion as a reduction in the quality of the video that they observe. Accordingly, for an Internet Service Provider (ISP), measuring the Quality of Experience (QoE) of the consumer is highly related to measuring the QoE of video [2].

In the past, a lot of work has been done on the estimation of QoE of video traffic. But most of the works more or less follow Deep Packet Inspection (DPI) of the network traffic. According to a study on North American internet network traffic [3], more applications are getting encrypted in the interest of protecting their content from the exposure to third parties. By the end of 2016, 70% of the internet traffic is expected to be encrypted. Two of the big video content providers worldwide, Netflix and YouTube, have already started encrypting their network traffic. YouTube has officially announced that currently 97 percent of its traffic is encrypted [4]. An encrypted internet will be a challenge for ISPs from the perspective of measuring QoE using DPI. With encrypted internet, since the payload will be hidden from ISPs, DPI will be unviable for the QoE analysis of video traffic.

The measure of the video QoE differs based on the delivery mechanism that is followed. OTT video is delivered in two primary streaming mechanisms: progressive video, where a single video file with a specific display quality is delivered at once in bursts; and adaptive video, where a video is divided into chunks of smaller length videos with different display quality, and a specific video quality is delivered to the user based on the capability of the network and the user’s device. In the case of adaptive video streaming, the changes between the different video qualities is a parameter that needs to be considered in the study of QoE. This switch between video qualities does not exist for a single video in progressive video download. But in both cases there are two parameters that must be measured separately and then considered together: display quality and transport quality [2].

The goal of this thesis is to exploit performance information that is found in the header of a video packet and use that to estimate the QoE of encrypted video. Accordingly, the QoE estimation solely bases on information from the packet header. Both display quality (i.e. through the study of bitrate adaptation) and transport quality (i.e. through the study of buffering events) are studied. Encrypted network traffic from
YouTube is used and an estimation is done for the occurrence or non-occurrence of bitrate adaptation and buffering events within a one-minute window duration. An attempt will be made to develop models that will detect the presence or absence of bitrate adaptation and buffering events within a one-minute window duration.

1.1 Motivation

As stated above, the Internet traffic is dominated by video traffic. Accordingly, ISPs need to maintain a good QoE of video. Video QoE is highly influenced by the buffering and bitrate adaptation events that occur in the video playback [5] [6]. It is vital for an ISP to find the relation between the condition at the network and the quality of the service the user experiences. This can be achieved by identifying the points where buffering and the bitrate adaptation events occur. With most of the internet traffic being encrypted, an ISP can have access only to the packet header information.

The motivation behind initiating this research is to detect buffering and bitrate adaptation events on the video playback solely using the packet header information of the encrypted video traffic. Machine learning algorithms are used for developing models that detect buffering and bitrate adaptation events.

1.2 Aim and Research Questions

The aim of this research is to develop machine learning models that detect the occurrence or non-occurrence of buffering and bitrate adaptation events within a one-minute duration window. While detecting the events, number and duration of buffering events are not taken in to account. Additionally, the bitrate adaptations are detected without considering adaptation down or adaptation up cases. The number of bitrate adaptation events is not also taken in to account.

The research questions that this thesis addresses are:

1. How are buffering and bitrate adaptation events correlated with network traffic variations for encrypted video traffic?
2. Which traffic features contribute most to the detection of buffering and bitrate adaptation events of encrypted video traffic?
3. Which machine learning model is better in estimating buffering and bitrate adaptation events using features extracted from encrypted video traffic?

1.3 Contribution

The first contribution of this thesis is building a lab set-up for collecting network level and application level data for the study of video QoE. Collecting the data in a lab set-up has an advantage in that the videos are played in a controlled environment. This enables to collect as much amount of application and network level data as needed. Additionally, the lab set-up is generic in that different types of network traffic shaping scenarios can be implemented in order to emulate the real world environment.

The other contribution is finding sets of features from the header of an encrypted video packet that can help in detecting the occurrence or non-occurrence of buffering and bitrate adaptation events. The importance of the features in detecting the buffering and bitrate adaptation events will be calculated, and correlation between these features and the events will be presented.
Finally, this work contributes in comparing the detection accuracy of the commonly used machine learning algorithms and suggesting the algorithm that gives better detection accuracy.

### 1.4 Structure of the Document

This document is structured in the following way:

Chapter One provides an introduction to the thesis work and presents the motivations for the thesis and the research questions that are to be addressed by this thesis work.

Chapter Two gives a background information about the technical concepts used in this thesis.

Chapter Three makes a survey of previous works that are done in relation to estimation of video QoE and analysis of encrypted video traffic using machine learning algorithms.

Chapter Four presents the experimental methodology that is used to conduct this thesis.

Chapter Five presents the statistics of the collected data and how features are extracted from the data.

Chapter Six discusses the results obtained out of the work done in this thesis.

Chapter Seven discusses the distribution of the most important features within the data. This chapter also presents an analysis of the relation between the most important features and the buffering and bitrate adaptation events.

Chapter Eight gives conclusion about the work done and the results obtained in this thesis. Additionally, the set of possible future works that can be done on the area will be presented in this chapter.

Chapter Nine gives an answer to each of the research questions that this work intended to answer.
2 BACKGROUND

The intention of this chapter is to give an understanding on the technical concepts used in this thesis work. Concepts related to Quality of Experience, Video streaming and machine learning will be discussed.

2.1 Quality of Experience

ITU defines QoE in the way given below[7]:

“Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service.

QoE Influencing Factors include the type and characteristics of the application or service, context of use, the user’s expectations with respect to the application or service and their fulfilment, the user’s cultural background, socio-economic issues, psychological profiles, emotional state of the user, and other factors whose number will likely expand with further research.

QoE Assessment is the process of measuring or estimating the QoE for a set of users of an application or a service with a dedicated procedure, and considering the influencing factors (possibly controlled, measured, or simply collected and reported). The output of the process may be a scalar value, multi-dimensional representation of the results, and/or verbal descriptors. All assessments of QoE should be accompanied by the description of the influencing factors that are included. The assessment of QoE can be described as comprehensive when it includes many of the specific factors, for example a majority of the known factors. Therefore, a limited QoE assessment would include only one or a small number of factors.”

2.2 Buffering and Bitrate Adaptation

In this document the word buffering is used to describe the stalling or freeze events that occur in a video playback. Initial Buffering and re-buffering will be used explicitly to refer to buffering at the beginning of video and buffering once the video has started playing, respectively. Bitrate adaptation is used to refer to the change in video playback quality. Bitrate adaptation up and bitrate adaptation down is used to explicitly refer to the increase and decrease of video playback quality, respectively. The names initial adaptation and re-adaptation refers to video quality changes at the beginning of the video and once the video has started playing, respectively.

2.3 Types of Video Streaming

Video streaming is classified into two classes as progressive and adaptive streaming. In progressive video streaming, the entire video data is downloaded at once with a fixed display quality. While the video is being downloaded to the buffer of the video player, the user starts watching the content. Usually, based on the network bandwidth of the user, the rate at which the video downloads exceeds the rate at which the user consumes the downloaded video data [8]. As a result, some of the downloaded video might not be consumed by the user, if he moves away without observing the whole video content, which results in unnecessary waste of network bandwidth. In HTTP Adaptive Streaming (HAS) the content provider divides the video into chunks of fixed length (e.g. 2 seconds) of different quality, and the player, on behalf of the
user, requests the video quality that suits to the device capabilities (e.g. screen size) and available bandwidth [9]. Accordingly, based on the network condition or the user’s device capability, the user experiences different display quality levels for a single video.

In this thesis, a network traffic data of YouTube video is analyzed. YouTube is a video content provider that is owned by Google and has a high share of the video traffic over the internet. YouTube uses adaptive streaming for delivering video traffic. The bitrate encoding of each chunk of videos vary based on the network condition and device capability of the user. The bitrate variations result in the variation of the video quality observed by the user. The available YouTube video qualities are presented in Table 2.1 [10].

<table>
<thead>
<tr>
<th>Video Quality</th>
<th>1440</th>
<th>1080</th>
<th>720</th>
<th>480</th>
<th>360</th>
<th>240</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>2560x</td>
<td>1920x</td>
<td>1280x</td>
<td>854x</td>
<td>640x</td>
<td>426x</td>
</tr>
<tr>
<td></td>
<td>1440</td>
<td>1080</td>
<td>720</td>
<td>480</td>
<td>360</td>
<td>240</td>
</tr>
<tr>
<td>Video Bitrate range (kbps)</td>
<td>6,000—18,000</td>
<td>3,000—9,000</td>
<td>1,500—6,000</td>
<td>500—2,000</td>
<td>400—1,000</td>
<td>300—700</td>
</tr>
</tbody>
</table>

Table 2-1 YouTube video quality levels [10]

2.4 YouTube Video Streaming Transport Protocols

The web browser that is used in this thesis work is Google Chrome. In the Google Chrome browser, YouTube videos are downloaded using HTTP2/SPDY, HTTPS and QUIC [20]. HTTP2/SPDY and HTTPS use TCP as transport protocol, whereas QUIC uses UDP as transport protocol. TCP is a connection oriented protocol that provides an end-to-end reliability. TCP fits in to a layered hierarchy of protocols where multi-network applications are supported and it enables reliable communication between different processes that are run in host computers which are connected to different interconnected computer networks [11].

QUIC (Quick UDP Internet Connections) is a new transport protocol for the internet which is developed by Google. It brings a solution to many transport-layer and application-layer problems faced by web applications. When QUIC is implemented, little or no change is required from the application perspective. Practically, QUIC is seemed to be a combination of TCP+TLS+HTTP2, but it is implemented on top of UDP [12].

2.5 Machine Learning

Machine learning is a field that extracts information and patterns out of a data and works to optimize performance based on the information extracted from the data. Accordingly, the quantity and quality of the data from which the information is extracted has a vital importance in the learning process of a machine learning algorithm [13]. Generally, a learning problem learns or extracts information (i.e. knowledge or pattern) from n samples of data and uses the information that is extracted from the data to predict behaviour of unknown data [14].
2.5.1 Data Set

The notion of data set in machine learning is defined in the way described below.

Labelled dataset D:

\[ X = \{ x^n \in R^n \}_{n=1}^N, \quad Y = \{ y^n \in R \}_{n=1}^N \]  \hspace{1cm} 2.1

Unlabelled dataset D:

\[ X = \{ x^n \in R^n \}_{n=1}^N \]  \hspace{1cm} 2.2

where X denotes the set of features that contain N samples. Each sample contains a d-dimensional vector \( x^n = [x^n_1, x^n_2, ..., x^n_d]^T \) which is named as a feature vector or feature sample. Each element of the d-dimensional vector is called an attribute, feature, variable, or element. Y represents the label set, recording what label a feature vector corresponds to [13].

2.5.2 Training Set and Test Set

In machine learning, three types of data are assumed to exist, a universal dataset, a training data set and a test data set. The universal data set is unknown and contains all the possible data pairs that can exist and the probability distribution of the data pairs in the real world. The second type of data set is a subset of the universal data set which is observed in real applications. The observed data set is used to gain information about the universal data set which is referred as training the machine learning algorithm, thus this data set is called training set (or training data). Generally, it is assumed that vectors in the training set are independently and identically sampled from the universal dataset. The test data set is the other type of data. This data set is also a subset of the universal data set and is used to evaluate the performance of the machine learning algorithm [13].

Machine learning aims to extract information or knowledge from the training set where the extracted information doesn’t only describe the training set but also the universal data set. Using the properties extracted from the training set, the machine learning algorithms will be able to predict unseen samples from the universal data set. The training set cannot be used to evaluate the performance of the algorithm, since all the information in the training set is known by the algorithm. As a result, another data set which is called test set will be reserved to measure the performance of the learning [13].

![Figure 2-1 Datasets in machine learning [13]](image)
As it can be seen in Figure 2.1, the unknown data set contains all the existing data types. The training data set and the test data set are subsets of the universal data set. The training data set serves for training learning algorithms about the properties of the universal data set. The test data set serves to evaluate the performance of the learning process. On the figure, two separating lines are shown to represent the learning process by two algorithms. Both lines have separated the training set with a 100 percent accuracy, this is expected result since they already have the information about the training set. On the contrary, the two lines made some error while classifying the data set in the test set. This error has appeared because the test set is a data that hasn’t been seen by the learning algorithms before. For classifying the test set, the algorithms are using the property that they have extracted from the training set [13].

2.5.3 Categories of Machine Learning

Generally, there are three types of machine learning based on the problem at hand and the type of data set, (1) supervised learning, (2) unsupervised learning, and (3) reinforcement learning [13]:

Supervised learning: While training the learning algorithm, the training data that is used are labelled data. The learning algorithm works to extract information about the relationship between the feature set and the label set of the data. Then the algorithm will be fed only with the feature set of unknown data and it predicts the label set of the unknown data. If for each feature set the corresponding label set is a discrete value, the learning task is referred as classification. On the contrary, if for each feature set the corresponding label set is continuous value, the learning task is referred as regression [13].

Unsupervised learning: In the unsupervised learning case, the training set that is used to train the algorithm is unlabelled data set. Unsupervised learning is used in Clustering, probability density estimation, finding association among features, and dimensionality reduction. The results obtained from unsupervised learning might further be used as an input for supervised learning [13].

Reinforcement learning: Reinforcement learning is used in decision making problems which usually involve sequences of decisions like robot perception and movement, automatic chess player, and automatic vehicle driving [13].

Added to the above three sets of machine learning types, semi-supervised learning is another type of learning that is getting attention recently. Semi-supervised learning lays between supervised and unsupervised learning in that it extracts knowledge from the data by using both labelled and unlabelled data [13].

2.5.4 Logistic Regression

Logistic regression is a learning algorithm that is applied for classification problems. For a given data sample with a set of features and a target value, logistic regression uses the features sets to calculates the probability of the target value being positive or negative [15]. Mathematically, given a sample $x_i$ and a target value $y_i$, the probability of $y_i$ being positive is estimated using equation 2.3.

$$p(y_i = 1) = \frac{1}{1 + \exp(-B_0 - B_1x)} \quad 2.3$$

On the process of training the algorithm on the learning data set, optimum values for the parameters $B_0$ and $B_1$ is determined.
2.5.5 Nearest Neighbor

The nearest neighbor learning method is applied for both supervised and unsupervised learning problems. In the supervised learning task, the nearest neighbor gives functionality in both classification and regression tasks. In the process of training, the algorithm learns about the relation between the feature sets and the corresponding label values. While predicting the label of a new data point, the algorithm works to find predefined number of training samples that are closest in distance to the new point and predicts the label based on these data points [16].

2.5.6 Support Vector Machine

Support vector machine is a machine learning algorithm that is used for both classification and regression tasks. In the learning process, the algorithm constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space. A hyper-plane is said to obtain a good separation if it has the largest distance to the nearest training data points of any class which is called functional margin. As the margin size increases, the generalization error gets reduced [17].

In order to construct the hyper-planes, SVM uses labelled data vectors which are known as training data sets. The classification is based on a unique set of features of each data vector. When the data set is more complex and non-linear, finding the best separator is turned to a linear task by transferring the input data into a higher-dimensional space known as the feature space. There are different kernel functions that are used for this transfer. After the best separator is found, the trained SVM will classify a new data that is not labelled, which is referred as test data [18].

2.5.7 Ensemble Methods

Ensemble methods are machine learning algorithms that aggregate the predictions of many estimators that are built with a specific learning algorithm. The goal of aggregating the predictions of many estimators is to improve generalizing capacity and robustness of the estimators [19].

2.5.8 Decision Trees

Decision Trees is a supervised learning algorithm for classification. The algorithm has root, node, branches and leaves, which is a concept that is derived from an ordinary tree structure. In decision trees, the circles designate the nodes and the nodes are connected with each other by segments which are referred as branches. The root node is the point where the decision tree starts and the point where the decision tree ends is referred as the leaf node. All the other nodes that are not root or leaf are referred to as internal nodes. The nodes represent a certain characteristic, which is referred to as feature in machine learning. The branches that extend from the nodes designate the range of values that a feature can have. Accordingly, the branches serve as classification points for the set of values of a specific feature [20]. Figure 2.2 represents the structure of a tree.
Data that is not classified is used to build a decision tree. The features that are best in dividing the data determine the division into classes. So, the data items are split and the data is grouped in the decision tree based on the values of the attributes of the given data. The process of grouping the data is done recursively for each split until the data sets in the remained subset belong to the same class [20].

2.5.9 Random Forest

Random forest is an ensemble method that works by aggregating many decision trees. Each tree in the forest is built using a sample data set that is taken from the training set with replacement. In addition to this, while splitting a node, the algorithm doesn’t consider the whole features set to determine the best split. The split bases only on the best split among of random subset of the features. Due to the fact that a random forest uses random subset of the data while building the trees in the forest, a random forest is slightly biased as compared to a single tree that is built on a whole data set. But the averaging that is done in the development of the forest decreases the variance compensating to the increase in the bias. In the end, the random forest results a better model than the individual trees [19].

When the forest classifies new data, each tree in the forest does its own classification task and reports the classification result. The data will be put in the class which is reported by the majority of the trees in the forest [21].

2.5.10 Model Evaluation Metrics

When a classifier performs classification task about a target value, it can give four different type of results. The target value being positive, the classifier can classify it as positive which is referred as true positive classification or negative which is referred as false negative classification. On the other hand, the target being negative, the classifier can classify it as positive which is referred as false positive classification or negative which is referred as true negative classification. Given these possible classification outcomes, it is possible to build a two-by-two confusion matrix which contains all the possible outcomes from a classifier. Using the confusion matrix, it is possible to drive other metrics that are used in the evaluation of machine learning algorithms [22]. A confusion matrix and the possible model evaluation metrics are given in Table 2.2 and Table 2.3 respectively.
### Table 2-2 Confusion matrix

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Actual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Negative</td>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td></td>
<td>True Negative (TN)</td>
<td>False Negative (FN)</td>
<td>False Positive (FP)</td>
</tr>
</tbody>
</table>

Table 2-3 Model evaluation metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive Rate (FPR)</td>
<td>FP / (FP + TN)</td>
</tr>
<tr>
<td>True Positive Rate (TPR)</td>
<td>TP / (TP + FN)</td>
</tr>
<tr>
<td>Precision</td>
<td>TP / (TP + FP)</td>
</tr>
<tr>
<td>Recall</td>
<td>TP / (TP + FN)</td>
</tr>
<tr>
<td>F-measure</td>
<td>2(precision⁻¹ + recall⁻¹)⁻¹</td>
</tr>
<tr>
<td>Accuracy</td>
<td>(TP + TN) / (TP + FN + FP + TN)</td>
</tr>
</tbody>
</table>

### 2.5.11 ROC Curve

ROC curve is a graph that is used to show the tradeoffs between true positives and false positives. It gives a visual representation of how much false positive will be introduced if the decision threshold is varied in the interest of increasing true positives [22].

![ROC Curve](image)

**Figure 2-3 ROC Curve**

The ROC curve have several special points to which special attention need to be given. The first point to consider is (0, 0), which represents a classifier that doesn’t give a positive classification. Such classifier doesn’t commit errors by outputting false positives but also it doesn’t give true positive outputs. On the contrary, a classifier operating at the (1, 1) point gives a positive classification all the time. The desired classification is the point that is represented as (0, 1) on the ROC curve. This is the point where the classifier classifies all the true positives correctly without committing any false positive classification [22].

In the space of the ROC curve, a point is said to be better if it is closer to the (0, 1) point which means that the more a point is to the northwest of the ROC space, the better it is. The x=y diagonal line is the space of operation of a randomly guessing classifier. If a classifier appears on the upper side of the diagonal line, it is extracting information out of the data to do the classification. On the contrary, a classifier that
appears on the lower side of the diagonal line is performing worse than a random classifier. This type of classifier has actually extracted information from the data, but while making the classification, it is using the information in the wrong way. So, it need to negate the way it is using the information so that the it will appear on the upper side of the diagonal line [22].

2.5.12 K-Fold cross validation

Cross-validation is a mechanism in machine learning that is used to classify data sets into train and test set. The algorithm extracts information from the train set and uses that information to predict the labels of the test set. This way the performance of the algorithm is validated. In practice, the data is split into many folds so that each fold has a chance of being in the test set at least once.

In k-fold cross-validation, the data is split into k equally (or near equally) sized folds. Training and testing works are done k times, where in each phase the k-1 folds are used for training the model and the remaining single fold is used for testing the model. In doing so, each fold has a chance of being a training fold k-1 times and a test fold once. The average accuracy of the k runs is reported as the accuracy of the model [23].

2.5.13 Scikit-learn

Scikit-learn is a module in python that implements classic machine learning algorithms and gives a simple and efficient solutions to machine learning problems. It uses the python scientific tools such as numpy, scipy, matplotlib [24]. These scientific tools are designed for doing different scientific computations, such as matrix computation which is implemented by numpy, integration of differential equations, optimization, and interpolations can be done using Scipy, and matplotlib is used for plotting of graphs[25].
3 RELATED WORK

The intention of this chapter is to discuss the works done in the past in relation to estimation of buffering and video quality, and machine learning applications for the classification of encrypted video traffic. Most works done previously used DPI for the estimation of buffering events. These studies [26] [27] analyzed the video meta data from the network traffic and extracted information relevant to the video playback status. The buffering events are estimated using the information that is extracted from the network traffic. Some attempts have also been done to estimate buffering events through an application that is installed on the user device [28] [29] [30]. On the other hand, on some studies [31], machine learning is used from the perspective of estimating the quality representation of videos.

The authors in [26] analyzed a YouTube video network traffic to estimate the stalling (i.e. buffering) events by extracting information from the network traffic. Three different approaches are followed to estimate the stalling pattern in the video. In their approaches, download time of the YouTube video, end-to-end throughput of the connection and the actual video buffer status are used for the estimation of the stalling events. They estimated the total stalling time, the number of stalling events and the duration of stalling time. The authors claimed that the stalling pattern estimated from network traffic trace happened to be almost identical to the actual stalling pattern. In all of their methods, DPI is used to extract video meta data information from the network traffic trace. This is a viable mechanism only in the case that the network traffic is not encrypted.

The authors in [31] worked on encrypted video traffic to classify the quality that the video is playing with. An algorithm is trained on network traffic from 120 videos running on three fixed video qualities. The test data set is based on videos that run both in fixed quality and auto quality mode. Bit per peak (traffic burst) is used as a feature and they claimed that 97.18 % classification accuracy is achieved. The robustness of the model is checked for different values of delays and packet drop on the network. It is observed that the classification accuracy of the model becomes as low as 70 percent for a condition that has higher delay and packet loss values. This gives an indication that more work need to be done with regard to videos that are being played in a dynamic network condition like wireless networks.

In [27] DPI is used for the estimation of length and number of buffering events in the video playback. The authors extracted information on the network traffic and attempted to estimate the buffering events. Their method mainly focuses on extracting video timestamps which are encoded within the payload and comparing this with the time stamp of the respective TCP segment. The authors claimed that they get a 100 percent accuracy for conditions where there is no buffering and a high accuracy when there is buffering. When a buffering happens, some difference is observed between the estimated duration of buffering and the actual duration of the buffering. Since YouTube is encrypting its traffic, their DPI method cannot be implemented currently. On top of this, the fact that the playback buffering events are collected manually raises questions regarding the accuracy of the number and duration of buffering events.

The authors in [28] approached the problem from the user end perspective. They developed an application called YoMo that is installed on the user device. The application interacts with the YouTube player through a YouTube API and collects information about the video data that is downloaded from the server to the YouTube player. The application predicts the occurrence of buffering events based on the
remaining downloaded video data in the buffer of the YouTube player. But this approach is difficult to be applied as ISPs do not own user devices.

In [29] an application called YOUQMON is developed that measures the QoE of YouTube video on operational 3G networks. The application estimates the number and duration of buffering events through the analysis of the video network traffic. The traffic analysis consists of two steps where initially the beginning of every new YouTube video flow is identified by HTTP header inspection and in the second step the playtime offsets of the corresponding video frames is extracted to estimate the buffered video playtime at the YouTube player. The remaining buffered video playtime is equal to the difference between video playtime so far downloaded and the current time. The occurrence of the buffering event is estimated based on the remaining buffered video playtime. The estimation of buffering events is also extended to a corresponding MOS rating. They claimed that the estimation of the number and duration of buffering is highly consistent with the buffering measured on the YouTube player. The MOS ratings are also seen to have high accuracy. Like the previous cases, from the prospective of current YouTube network traffic, their method fails to work on encrypted YouTube traffic.

An application named as YouSlow is developed in [30] to collect buffering statistics of YouTube video playbacks worldwide. The application is customized for Google Chrome browser and it collects statistics like initial buffering time, requested bitrates, buffer stalling duration, and approximate location of buffer stalling events and local ISP information. The buffering statistics information is logged to a central server. The authors claimed that they collected more than 20000 YouTube buffering events from more than 40 countries and the analysis of the collected information is presented in their work. Though the application is not aimed at estimating buffering events from a specific ISP perspective, it is a good effort to understand and compare YouTube video buffering scenarios for different ISPs in different geographical region.
4 EXPERIMENTAL METHODOLOGY

This chapter discusses the experimental setup, hardware devices and software tools that are used in this thesis work.

4.1 Experimental Setup

The experiment is done using the experimental setup shown in Figure 4.1. The experimental setup consists of two devices, a shaper machine and a client machine. The shaper machine operates on Ubuntu 10.01, with Intel core i7 processor, 2.9 GB RAM and disk size of 908.4 GB. It is connected through a USB modem to a 100Mbps intranet. The intranet is connected to the public internet.

Packet capture before applying traffic shaping, i.e. Pre-shaping packet capture, is done at the shaper machine’s interface connected to the internet. A traffic shaping is applied on the egress traffic at the shaper machine’s interface that connects the shaper machine with the client machine. The client machine has an Intel core i7 processor, 4 GB of RAM, and 732 GB disk space. It operates on Ubuntu 15.10. It is connected to the shaper machine via Ethernet full duplex link, bandwidth of 100Mbps. Packet capture after applying traffic shaping, i.e. Post-shaping packet capture, is done at the client machine’s interface connected to the shaper machine. The client machine gets access to the internet through the shaper machine. The shaper and the client machine have their time synchronized using the Network Time Protocol (NTP).

4.2 Browser and Client Script

The client machine runs the browser and client script. The browser script is a JavaScript code that implements a YouTube player API called IFrame player API. The IFrame API enables to embed a YouTube video player on a website. The API has JavaScript functions that allow to control the YouTube video playback such as, queue videos for playback; play, pause, or stop those videos; adjust the player volume; or retrieve information about the video being played. Event listeners can also be added. These event listeners execute in response to certain player events, such as a player state change or a video playback quality change [32]. In this experiment, two event listeners of the IFrame API are used, onStateChange and onPlaybackQualityChange.

4.2.1 onStateChange event listener

This event listener executes whenever there is a change in the state of the player. On the occurrence of state change, the API passes an integer value to the event listener. The integer values correspond to the new player state. The possible values are: -1 (unstarted), 0 (ended), 1 (playing), 2 (paused), 3 (buffering), 5 (video cued) [32].

![Figure 4-1 Experimental setup](image-url)
For each of the video played in the experiment, all the video state changes that have occurred during the video playback have been captured and saved in a text file for further analysis.

4.2.2 **OnPlaybackQualityChange event listener**

This event listener executes whenever the video playback quality changes. On the occurrence of quality change, the API passes to the event listener a string that identifies the new playback quality. Possible values are: small, medium, large, hd720, hd1080, highres [32]. The video quality increases as we go from small to highres. In this lab implementation the maximum video quality that is observed is hd1080. This is because the size of the client machine’s screen do not support video resolutions above hd1080. In addition to the mentioned video qualities, a tiny video quality is reported by the API when the traffic bit rate value lowers below the one required for small video quality. For each of the video played in the experiment, all the video quality changes that have occurred during the video playback have been captured and saved in a text file for further analysis.

The client script is a bash script code that launches the browser script at the start of a new video and closes the browser script when a video ends playing. The client script is fed with list of 70 YouTube video IDs with a playback length ranging from 00:01:08 up to 01:19:11 hours. All the videos have a 4k video resolution. When the client script launches the browser script, it passes a YouTube video ID to the browser script. The browser script plays the specific video using Google Chrome browser.

4.3 **Controller and Shaper Script**

The shaper machine runs the controller and the shaper script. The controller script is a bash script code that launches the client script at the start of the lab. While the lab is running, the controller script launches and closes the shaper script on the start and end of a video respectively. The shaper script is a bash script code that implements a traffic shaping mechanism that is inbuilt in the Ubuntu operating system. The interaction between browser, client, controller and shaper script is given in Figure 4.2.
Figure 4-2 Interaction sequence diagram between controller, Shaper, client and browser script
4.4 Traffic Shaping

Traffic shaping is the mechanism that is used to keep the output rate of a transmission to a desired rate by delaying packets [33]. The Ubuntu operating system has built-in traffic control mechanism. Scheduling and classifying are the two traffic shaping functionalities that are used in this experiment.

Scheduling is used to arrange or rearrange packets between input and output of a particular queue [33]. Queuing disciplines (qdisc) offer a scheduling capability in the Ubuntu operating system. There are two types of queuing disciplines, classless and classful. The classless qdiscs do not have classes or subdivisions. It is not possible to give special treatment to a certain type of traffic using classless qdisc. The traffic shaping that is implemented on classless qdisc is applied on the entire interface [33] [33]. On the contrary, classful qdiscs can contain classes, and a filter can be attached to them to classify traffic to different subdivisions. Accordingly, in a classful qdisc, it is possible to apply different traffic shaping on two different types of network traffic. In this experiment, the Hierarchical Token Bucket (HTB) classful queuing discipline and the Token Bucket Filter (TBF) classless queuing discipline are implemented.

Classifying is used to separate packets into different queues, so that they will be treated differently. In the Linux operating system, filters perform the role of classifier. When packet arrives to an interface, it enters to the root qdisc. If a filter is attached to the root qdisc, the packet will be directed to a subclass for a special treatment. The subclasses themselves might have their own filter to sort the packet for further classification [33].

4.4.1 Token Bucket Filter

The Token Bucket Filter (TBF) is a simple qdisc that shapes network traffic by passing only packets that arrive at a certain administratively set rate. A certain amount of traffic burst can be set above the administratively set rate. In the implementation of TBF, there is a buffer which is named as bucket. This bucket is filled with a virtual pieces of information called tokens at the rate that is set administratively called token rate. When a data packet arrives to the interface, it collects one token from the bucket and be transmitted. If there is no token in the bucket, packets arriving on the interface will wait in a queue until a token becomes available in the bucket. The rate limiting is directly applied on the tokens, but since packets are transmitted based on the availability of tokens, the packets will be transmitted with a maximum rate upper-bounded by token arrival rate [34].

4.4.2 Hierarchical Token Bucket Filter

Hierarchical Token Bucket filter qdisc is the same as Token Bucket (TB) filter qdisc except that HTB is classful. It follows the same principle of limiting the packet rate by limiting arrival rate of tokens. But since it is classful, packets can be classified in to different classes so that they will pass through different treatment [35].

In this experiment, one root HTB queuing discipline and two leaf HTB queuing disciplines are implemented. TBF qdisc is attached to each of the leaf HTB qdisc. One of the leaf HTB qdisc applies traffic shaping. The other leaf HTB qdisc allows a traffic without applying any shaping. All network traffic that originates from YouTube server is by default directed to the leaf qdisc that applies traffic shaping. A filter is attached to
the root queuing discipline to direct all SSH and NTP control traffic to the qdisc that applies no traffic shaping.

A bit rate values ranging from 1 kbps up to 20 Mbps are applied for shaping the traffic. The bit rate values are generated following a uniform random distribution. Each bit rate value is applied for a duration that ranges from 0 second up to 180 seconds. The duration values are also generated following a uniform random distribution. As can be seen from the CDF plot of the applied bitrates and durations in Figures 4.3 and 4.4, respectively, the generated values follow a uniform distribution. The variation range of the applied bitrates is chosen to be in the range of 1 kbps up to 20 Mbps so that all the video resolutions that are observed in this lab setup, i.e. from tiny up to hd1080, can be played effectively. The upper bound of the duration is set to 3 minutes in the interest of randomizing the applied bitrate values frequently so that a single applied bitrate value will not have a prolonged effect.

The experiment is run for 24 rounds where each round contains 70 videos which results to 1680 runs of individual videos. This took 36683 minutes, which is equivalent to 25.474 days.

![Figure 4-3 CDF of bitrates applied by the traffic shaper](image)

![Figure 4-4 CDF of durations for which the traffic shaper applies the bitrate](image)
5 **DATA ANALYSIS**

In this chapter, the statistics of the collected data and the methodologies used in the analysis of the data will be discussed.

5.1 **Data Statistics**

The data that is collected at the pre-shaping traffic capture point is used in the analysis of the network traffic data. At the pre-shaping capture point a network traffic data of size 596.6GB is collected. The video data transfer occurred using both TCP and UDP (i.e. QUIC) protocol. The payload of all the captured packet is encrypted. The packet headers are transferred in clear text.

An application data is also collected using IFrame YouTube API [32]. At the application, video state change events (i.e. Unstarted, Playing, Buffering and Ended) and video quality change events (i.e tiny, small, medium, large, hd720, and hd1080) are collected. The relation between the collected video quality change events and their corresponding video resolution (in pixel) is given in Table 5.1. The tiny video quality is observed when the bitrate becomes less than a value that can play the small video quality, which is 300 kbps. The statistics of video quality change and buffering events is given in Figures 5.1 and 5.2 respectively.

<table>
<thead>
<tr>
<th>Observed video quality</th>
<th>hd1080</th>
<th>hd720</th>
<th>large</th>
<th>medium</th>
<th>small</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video resolution (in pixel)</td>
<td>hd1080</td>
<td>hd720</td>
<td>480</td>
<td>360</td>
<td>240</td>
</tr>
</tbody>
</table>

Table 5-1 Resolutions for the observed video qualities

![Figure 5-1 Video quality statistics](image-url)
The main goal of the model that is to be developed is to estimate the occurrence or non-occurrence of buffering and bitrate adaptation events within a duration of one-minute window.

To estimate the events within the one-minute window, the data is divided into sets of one-minute durations. The relevant feature sets that will be explained in section 5.2 are calculated for each of the one-minute duration window. In this thesis, the naming “current one-minute window” is used to refer to the one-minute window duration where the estimation of buffering and bitrate adaptation is currently being done. The naming “previous one-minute window” is used to refer to the one-minute window that is found right before the current one-minute window. The set of all possible events that can happen in the video playback, and the concept of current one-minute and previous one-minute is depicted in Figure 5.3.

![Figure 5-2 Buffering/Bitrate adaptation statistics](image)

**Figure 5-2 Buffering/Bitrate adaptation statistics**

![Figure 5-3 Flow of events in a video](image)

**Figure 5-3 Flow of events in a video**
The statistics of the one-minute duration window with respect to buffering and bitrate adaptation events are given in Figure 5.4.

![Figure 5-4 Statistics of buffering/bitrate adaptation events in the one-minute window](image)

From the collected network traffic data, two basic features are extracted, server bitrate and server packet inter-arrival time. Further features are derived from these basic features. In the next section all features that are used in the model development are explained.

## 5.2 Feature Calculation

The features that are explained in this section are calculated for each of the one-minute duration windows.

### 5.2.1 Download Throughput Over One Second ($1R$)

This feature is the sum of bits arriving in the server to client direction within an interval of one second. The download throughput over one second calculation is summarized by equation 5.1.

Assume:

- $P_t =$ Server packet size in bits that has arrived at the $t^{th}$ second.
- $T =$ time in seconds

\[
1_R = \sum_{t=T}^{T} P_t
\]

### 5.2.2 Download Throughput Over 10 Seconds ($10R$)

This feature is the sum of bits arriving in the server to client direction within an interval of ten seconds. The download throughput over 10 seconds calculation is summarized by equation 5.2.
Assume:

\[ 1^R_t = \text{Download throughput over one second value at the } t^{th} \text{ second} \]
\[ T = \text{time in seconds} \]

\[ 10_R = \sum_{t=T-9}^{t=T} 1_R_t \]

5.2

This feature is motivated in the interest of keeping the sequence information of the data. As it is stated in RFC2330 [36], the information of the sequence in which the data arrive will be lost when percentiles of data are calculated. Since the data that is considered in this thesis is a time series data, the sequence in which the data arrives is an important information. As a result, the bitrate per 10 second values are calculated for each one-minute window and feed the values to the machine learning algorithm keeping their sequence. A value of 10 second is selected based on the effect on the estimation accuracy. Bitrate per 5, 10, 15 and 25 seconds are tried during the model development and bitrate per 10 second values are the ones that gave the highest accuracy. This might be due to the size of the player buffer or the length of chunks of videos that are sent from YouTube server at a time.

Each of the one-minute duration window will have 6 download throughput over 10 second values. For the current and the previous one-minute windows, the six download throughput per 10 second values are represented as in Figure 5.5. The subscripts “P” and “C” are used to refer to previous and current minute respectively.

Figure 5.5 Representations of download throughput over 10 second values

5.2.3 Change in download throughput over one second (\( \Delta 1^R \))

This is the absolute valued difference between the two consecutive download throughput over one second values. The calculation of this feature is summarized by equation 5.3.

Assume:

\[ 1^R_t = \text{Download throughput per second value at the } i^{th} \text{ second} \]
\[ 1^R_{i+1} = \text{Download throughput per second value at the } (i+1)^{th} \text{ second} \]
\[ \Delta 1^R = \text{Absolute valued difference between the download throughput per second at the } i^{th} \text{ and } (i+1)^{th} \text{ second} \]

\[ \Delta 1^R = |1^R_{i+1} - 1^R_i| \]
5.2.4 Low Relative Download Throughput ($R_s$)

This feature is calculated, for each one-minute window, by summing the download throughput over one second values that are less than or equal to 1000 kbps and dividing this by the sum of all download throughput over one second values. Mathematically, this feature is represented by equation 5.4.

$$R_s = \frac{\sum t \min (R, 1000 \text{ kbps})}{\sum t R}$$  \hspace{1cm} 5.4

While calculating this feature, the initial motivation behind choosing a threshold value of 1000 kbps arises from the observation of the conditions where buffering happens in the video playback. It is observed that usually buffering happens when the download throughput is lower than 1000 kbps. Other threshold values, i.e. 800 kbps, 1500 kbps, and 2000 kbps, are also tested to calculate this feature, but the best estimation accuracy is obtained when the threshold value is set to 1000 kbps.

5.2.5 Server Packet Inter-arrival time

This feature is the time difference between the arrival of two consecutive packets from the server to the client. The server packet inter-arrival time calculation is summarized by equation 5.5.

Assume:

- $T_i = \text{The time at which a packet arrives at the probe point from server}$
- $\Delta t_i = \text{Server packet Inter-arrival time}$

$$\Delta t_i = T_{i+1} - T_i$$  \hspace{1cm} 5.5
6 Result

This chapter presents the model development steps and the result on the detection of buffering and bitrate adaptation events. All the detection is done for a one-minute duration window.

6.1 Model Development

Two machine learning models are developed for detecting buffering and bitrate adaptation events within a duration of one-minute window. The buffering and bitrate adaptation events that occur at the beginning of the videos are removed from the data. This is because of the fact that it is the re-buffering and re-adaptation events that occur due to a change in the network condition. The initial buffering and initial bitrate adaptation events always occur when the video starts, irrespective of the network condition. The model development process is represented by Figure 6.1.

![Figure 6-1 Model development steps](image)

The raw data contains columns of values of server packet arrival time, server and client packet size, and events occurred at the video playback. The set of features explained in section 5.2 are extracted from the raw data. Machine learning algorithms process data that are structured as set of features with a corresponding label. Accordingly, the data is divided into samples of one-minute windows and each sample contains the calculated features with the corresponding labels. The labels are represented as zero and one. For the model that detects buffering, zero corresponds to the event where there is no buffering and one corresponds to the event where there is buffering. Similarly, for the model that detects bitrate adaptation, zero corresponds to the event where there is no bitrate adaptation and one corresponds to the event where there is bitrate adaptation. The structured data is divided in to 75 and 25 percent splits where the 75 percent contains the training set and the 25 percent contains the test set. In the development of the model, these splits are used for training and testing the model. 10-fold cross validation is used while developing the models using the training set.

The model development task is a process where various features are calculated and the model is developed based on the calculated features. The features are retained or rejected based on their effect on the accuracy of the model. Some of the features that are tested in the development of the model are: the size of packets from YouTube server and the client machine, inter-arrival time between packets from YouTube server and inter-arrival time between packets from the client, and statistical and time-series
analysis of the extracted features. The final set of features that are used for the detection of re-buffering and bitrate re-adaptation event are given in Table 6.1 and Table 6.2.

<table>
<thead>
<tr>
<th>Basic Features</th>
<th>Derived Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-arrival time</td>
<td>Mean, Variance, Range, 75 percentiles, 90 percentiles, Maximum</td>
</tr>
<tr>
<td>$^1R$</td>
<td>Mean, Variance, Range, 50 percentiles, 75 percentiles, 90 percentiles, Maximum, $R_5$</td>
</tr>
<tr>
<td>$\Delta^1R$</td>
<td>Mean, Variance, Range, 25 percentiles, 50 percentiles, 75 percentiles, 90 percentiles, Maximum</td>
</tr>
<tr>
<td>$^{10}R$</td>
<td>$^{10}R_{p4}$, $^{10}R_{p5}$, $^{10}R_{p6}$</td>
</tr>
</tbody>
</table>

Table 6-1 Features for detection of re-buffering

<table>
<thead>
<tr>
<th>Basic Features</th>
<th>Derived Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^1R$</td>
<td>Mean, Variance, 75 percentiles, 90 percentiles, $R_5$</td>
</tr>
<tr>
<td>$\Delta^1R$</td>
<td>75 percentiles, 90 percentiles, Sum</td>
</tr>
<tr>
<td>$^{10}R$</td>
<td>$^{10}R_{p4}$, $^{10}R_{p5}$, $^{10}R_{p6}$, $^{10}R_{c1}$, $^{10}R_{c2}$, $^{10}R_{c3}$, $^{10}R_{c4}$, $^{10}R_{c5}$, $^{10}R_{c6}$</td>
</tr>
</tbody>
</table>

Table 6-2 Features for detection of re-adaptation

The python scikit-learn module is used to train four well known machine learning algorithms named as K-Nearest Neighbor (KNN), Logistic Regression, Support Vector Machine, and Random Forest. Support Vector Machines and Random Forest are chosen as previous studies [37] [38] [39] suggest that those algorithms give a good accuracy in many circumstances. Additionally, Logistic Regression and KNN are trained because these algorithms have an advantage in that they are less complex and faster to train [40] [41].

The machine learning algorithms have different parameters that need to be tuned for optimal estimation accuracy. In the scikit-learn implementation of the trained algorithms, the parameters that usually need tuning are: “C” for Logistic Regression, “Number of Neighbours” for KNN, “C” and “gamma” for support vector machine, and “Number of estimators” and “Maximum depth” for Random Forest.

The “C” parameter for logistic regression determines the complexity of the model. As the value of “C” increases, the model becomes more complex that it memorizes the details of the training data and faces difficulty in predicting the labels of a test data. On the contrary, a model with low value of “C” is less complex and it studies properties of the training data, rather than memorizing the details of the data, and can better predict the labels of a test data [42].

The “Number of neighbours” parameter in KNN determines the number of neighbours that the algorithm needs to consider while predicting the label of a given data point. The weight parameter can be “uniform” or “distance”, where in the case of “uniform” weight, all the considered neighbours have equal weight, whereas while using “distance” weight, closer points of a query will have greater influence than neighbours that are far away [43].

In Support Vector Machine, the values of “gamma” and “C” are tuned. The “gamma” parameter determines the influence distance of a training data that is selected as support vector. Low values of “gamma” signifies “far” influence by the support vectors. On the other hand, the “C” parameter determines the complexity of the model. Like logistic regression, model with high values of “C” is more complex and memorizes the details of the data than a model with low values of “C” [44].
The two parameters that are usually tuned in random forest are “Number of estimators” and “Maximum depth”. The “Number of estimators” parameter determines the number of trees in the forest. The “Maximum depth” parameter controls the maximum depth of a tree [45].

In scikit-learn, the Grid search module is used to search for optimum model parameters that are not directly learned by the estimators [46]. Accordingly, this module is used to search the optimum values for the parameters that need to be tuned for each of the learning algorithms. The tuned parameters, the range of tested values and the selected parameter values are presented in Table 6.3.

<table>
<thead>
<tr>
<th>Trained Algorithm</th>
<th>Logistic Regression</th>
<th>K-Nearest Neighbor</th>
<th>Support Vector Machine</th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tuned Parameters</strong></td>
<td>C</td>
<td>n_neighbors and weights</td>
<td>C and gamma</td>
<td>n_estimators and Max_depth</td>
</tr>
<tr>
<td><strong>Tuning Range</strong></td>
<td>C: 0.001, 0.01, 0.1, 1, 10, 100, 150, 200</td>
<td>n_neighbors: 5, 7, 10, 15, 20, 100, weights: uniform, distance</td>
<td>C: 0.001, 0.01, 0.1, 1, 10, 100, gamma: 0.001, 0.01, 0.1, 1, 10, 100</td>
<td>n_estimators: 10, 20, 30, 40, 50, 100, 150, 200, Max_depth: 5, 10, 15, 20, 25, 30</td>
</tr>
<tr>
<td><strong>Selected Parameters</strong></td>
<td>C: 10</td>
<td>n_neighbors: 10, weights: distance</td>
<td>C: 10, gamma: 0.01</td>
<td>n_estimators: 100, Max_depth: 20</td>
</tr>
</tbody>
</table>

Table 6-3 Parameters tuned for the learning algorithms

6.2 Model Evaluation

Most of the model evaluation metrics that are described in section 2.5.10 of this document are used for evaluating the detection accuracy of the developed models.

The data that is used in this thesis has a very high class imbalance. The number of samples that contain no re-buffering/bitrate re-adaptation are much higher than the number of samples with re-buffering/bitrate re-adaptation. Because of this imbalance, models can easily predict the labels of the class having the highest appearance in the data. As a result of this, a model that is not actually extracting information from the data can end up with a very high overall accuracy, which make the overall accuracy measure to be unreliable. Accordingly, rather than overall accuracy, other model evaluation metrics like precision, recall, f1 score and ROC curve are used as a main measure of model performance in this thesis.

6.2.1 Detection of Re-Buffering Events

The model for detecting re-buffering events is developed using the data where each sample contains set of features over one-minute window and the corresponding labels. The labels are either zero which represents no re-buffering event, or one which corresponds to re-buffering event. All the four learning algorithms are trained on the training split of the data and the performance of the models is evaluated on the data split that is reserved for testing. The overall accuracy, confusion matrix, and classification report of all the trained learning algorithms are given in the next
sections. These evaluation metrics are generated by running all the four learning algorithms 40 times and taking the average of the 40 runs.

For each class, the confusion matrix presents the number of samples that are identified correctly and that are missed by the learning algorithm. For the case of logistic regression, as presented in Table 6.4, out of a total of 8358 samples that does not contain re-buffering events; the algorithm has correctly detected 8294 of them. On the contrary, the remaining 64 samples, which actually does not contain re-buffering events, are predicted by the algorithm as classes that does contain re-buffering events. While detecting the samples that contain re-buffering events, out of a total of 372 samples, the algorithm detected 167 of them correctly. The algorithm confused the remaining 205, with the samples that does not contain re-buffering events.

From the statistics of the confusion matrix table, it is possible to derive various evaluation metrics, like precision, recall and f1-score. In the prediction of the re-buffering events, precision tells the chance that a re-buffering event prediction is correct. On the other hand, recall tells the chance that a sample that is actually re-buffering event will get predicted as such. Accordingly, for logistic regression, 73 percent of the re-buffering event predictions are correct and the algorithm has correctly detected 45 percent of the samples that contain re-buffering events. The same argument holds true for the samples with no re-buffering events.

The overall accuracy of Logistic Regression is 0.969

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-buffering</th>
<th>Re-buffering</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>8294</td>
<td>64</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>205</td>
<td>167</td>
</tr>
</tbody>
</table>

Table 6-4 Confusion matrix of Logistic Regression

| No Re-buffering | 0.97 | 0.99 | 0.98 | 8358 |
| Re-buffering | 0.73 | 0.45 | 0.56 | 372 |
| Avg/Total | 0.964 | 0.969 | 0.965 | 8730 |

Table 6-5 Classification report of Logistic Regression

The overall accuracy of KNN is 0.971

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-buffering</th>
<th>Re-buffering</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>8278</td>
<td>80</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>169</td>
<td>203</td>
</tr>
</tbody>
</table>

Table 6-6 Confusion matrix of KNN

| No Re-buffering | 0.98 | 0.99 | 0.98 | 8358 |
| Re-buffering | 0.72 | 0.55 | 0.62 | 372 |
| Avg/Total | 0.969 | 0.969 | 0.969 | 8730 |

Table 6-7 Classification report of KNN
The overall accuracy of Support Vector Machine is 0.972

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-buffering</th>
<th>Re-buffering</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>8292</td>
<td>66</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>172</td>
<td>200</td>
</tr>
</tbody>
</table>

Table 6-8 Confusion matrix of support vector machine

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>0.98</td>
<td>0.99</td>
<td>8358</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>0.75</td>
<td>0.54</td>
<td>372</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.969</td>
<td>0.971</td>
<td>8730</td>
</tr>
</tbody>
</table>

Table 6-9 Classification report of support vector machine

The overall accuracy of Random Forest is 0.975

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-buffering</th>
<th>Re-buffering</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>8290</td>
<td>68</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>150</td>
<td>222</td>
</tr>
</tbody>
</table>

Table 6-10 Confusion matrix of random forest

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>0.98</td>
<td>0.99</td>
<td>8358</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>0.77</td>
<td>0.60</td>
<td>372</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.971</td>
<td>0.975</td>
<td>8730</td>
</tr>
</tbody>
</table>

Table 6-11 Classification report of random forest

As can be seen on the confusion matrix and classification report tables, Random forest has shown the best performance in all regards in detecting the re-buffering events. Out of 372 re-buffering events, random forest has detected 222 of them and missed the remaining 150. It has shown an f1-score value of 67 percent. Logistic regression is the algorithm that has performed the least. It has f1-score accuracy of 56 percent in detecting the re-buffering events.

95 percent confidence interval are calculated for the performance of the algorithms in detecting re-buffering events and the resulting bar graphs are given in Figure 6.2 and Figure 6.3.
Since random forest is showing better performance in all aspects, the model developed using random forest will be studied more closely in the sections below. The importance of the features in detecting re-buffering events for the model developed using random forest is given in Figure 6.4.

The Feature importance figure shows that low relative download throughput is the most important feature in detecting the re-buffering events. The download throughput values in the last 20 second of the previous minute have also a higher contribution in the feature importance list. The other important features are the inter-arrival times between packets from YouTube server. Some of the most important features will be studied and discussed in more detail in the Discussion and Analysis chapter of this document.

The ROC curve of the model developed using random forest is given in Figure 6.5. As can be seen on the ROC graph, the model has a curve that is closer to the northwest of the ROC space. This signifies that true positive rates can be increased by varying the decision threshold of the model without incurring higher increase on the false positive rates.
As discussed previously, an imbalance between the target classes in a data creates a bias in the accuracy calculation of a model. The accuracy favours the target value that has the highest number of frequency in the data. If the frequencies of the target values is equalized, the reported accuracy for the less frequent target value is expected to increase. For the data that is used in this thesis, the samples that do not have re-buffering events are much larger than the samples that have re-buffering events. Accordingly, the detection accuracy of the “no re-buffering” events is very high. In the interest of validating the model performance, some of the samples that doesn’t have re-buffering event are randomly dropped and the data is made to contain an equal number of samples with re-buffering and no re-buffering events. The model accuracy is tested on the equalized data, and the confusion matrix and classification report values are reported in Table 6.12 and Table 6.13, respectively.

The overall accuracy is 0.95

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Re-buffering</td>
</tr>
<tr>
<td>No Re-buffering</td>
<td>344</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 6-12 Confusion matrix of random forest for equalized target values

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-buffering</td>
<td>0.94</td>
<td>0.95</td>
<td>0.95</td>
<td>362</td>
</tr>
<tr>
<td>Re-buffering</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>382</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>744</td>
</tr>
</tbody>
</table>

Table 6-13 Classification report of random forest for equalized target values

As it can be seen from the confusion matrix and classification report table, the detection accuracy of the model has shown an increment for the samples that contain re-buffering events. For equalized target values, a model that is guessing without extracting information from the data is expected to be correct 50 percent of the time. This suggests that the developed model is extracting information from the data while doing the estimation, since it achieved much higher accuracy than 50 percent.

6.2.2 Detection of Re-Adaptation Events

Like the case of re-buffering, the model for detecting re-adaptation events is developed using the data where each sample contains set of features over one-minute window and the corresponding labels. The labels are either zero which represents no re-adaptation event, or one which corresponds to a re-adaptation event. All the four learning algorithms are trained on the training split of the data and the performance of the models is evaluated on the data split that is reserved for testing. The overall
accuracy, confusion matrix, and classification report of all the trained learning algorithms is given in sections below. The evaluation metrics are generated by running all the algorithms 40 times and calculating the average of the 40 runs.

The overall accuracy of Logistic Regression is 0.892

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-adaptation</th>
<th>Re-adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>7456</td>
<td>177</td>
</tr>
<tr>
<td></td>
<td>766</td>
<td>353</td>
</tr>
</tbody>
</table>

Table 6-14 Confusion matrix of logistic regression

<table>
<thead>
<tr>
<th>No Re-adaptation</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.91</td>
<td>0.98</td>
<td>0.94</td>
<td>7633</td>
</tr>
<tr>
<td>0.67</td>
<td>0.32</td>
<td>0.43</td>
<td>1119</td>
</tr>
</tbody>
</table>

Table 6-15 Classification report of logistic regression

<table>
<thead>
<tr>
<th>No Re-adaptation</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.876</td>
<td>0.89</td>
<td>0.874</td>
<td>8752</td>
</tr>
</tbody>
</table>

The overall accuracy of KNN is 0.919

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-adaptation</th>
<th>Re-adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>7440</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>516</td>
<td>603</td>
</tr>
</tbody>
</table>

Table 6-16 Confusion matrix of KNN

<table>
<thead>
<tr>
<th>No Re-adaptation</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.94</td>
<td>0.97</td>
<td>0.95</td>
<td>7633</td>
</tr>
<tr>
<td>0.76</td>
<td>0.54</td>
<td>0.63</td>
<td>1119</td>
</tr>
</tbody>
</table>

Table 6-17 Classification report of KNN

<table>
<thead>
<tr>
<th>No Re-adaptation</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.911</td>
<td>0.92</td>
<td>0.912</td>
<td>8752</td>
</tr>
</tbody>
</table>

The overall accuracy of Support Vector Machine is 0.917

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-adaptation</th>
<th>Re-adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>7455</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>548</td>
<td>571</td>
</tr>
</tbody>
</table>

Table 6-18 Confusion matrix of support vector machine

<table>
<thead>
<tr>
<th>No Re-adaptation</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.93</td>
<td>0.98</td>
<td>0.95</td>
<td>7633</td>
</tr>
<tr>
<td>0.76</td>
<td>0.51</td>
<td>0.61</td>
<td>1119</td>
</tr>
</tbody>
</table>

Table 6-19 Classification report of support vector machine

The overall accuracy of Random Forest is 0.929

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No Re-adaptation</th>
<th>Re-adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td>7427</td>
<td>206</td>
</tr>
<tr>
<td></td>
<td>413</td>
<td>706</td>
</tr>
</tbody>
</table>

Table 6-20 Confusion matrix of random forest
<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Re-adaptation</strong></td>
<td>0.95</td>
<td>0.97</td>
<td>0.96</td>
<td>7633</td>
</tr>
<tr>
<td><strong>Re-adaptation</strong></td>
<td>0.77</td>
<td>0.63</td>
<td>0.70</td>
<td>1119</td>
</tr>
<tr>
<td><strong>Avg/Total</strong></td>
<td>0.926</td>
<td>0.929</td>
<td>0.926</td>
<td>8752</td>
</tr>
</tbody>
</table>

Table 6-21 Classification report of random forest

From the accuracy reports, it can be seen that random forest has performed better than the other algorithms in detecting the re-adaptation events, securing a 70 percent f1-score. Out of 1120 re-adaptation events, the model based on random forest has correctly detected 706 of them and missed the remaining 413 re-adaptation events. Logistic regression is the algorithm that performed the least in detecting the re-adaptation events, with an f1-score accuracy of 43 percent. In the same way as re-buffering, 95 percent confidence interval are calculated for the performance of each of the algorithms in detecting re-adaptation events and the results are given in Figure 6.6 and Figure 6.7.

Because random forest is performing best, the model developed based on random forest is studied more closely. The importance of the features in detecting the re-buffering events is given in Figure 6.8.
The feature importance plot shows that the download throughput values in the last 20 second of the previous minute has higher importance in detecting re-adaptation events. The low relative download throughput is the other feature that has a higher importance. The distribution of the most important features and their trend of variation will be discussed in detail in the Discussion and Analysis chapter of this document.

The ROC curve of the model developed using random forest is given in Figure 6.9. The model has a curve that is closer to the north west of the ROC space. This indicates that the decision threshold of the model can be varied to increase true positive rate without introducing much false positive rates.

In the detection of re-adaptation events also, the model developed using random forest is tested using data samples that have equal number of target class values. Accordingly, the number of samples that has no re-adaptation labels is made to be the same as the number of samples with re-adaptation labels. The model is expected to perform better in identifying the labels the has re-adaptation. On the other hand, the overall accuracy of the model is expected to show a reduction. The confusion matrix and the classification report ratings are given in Table 6.22 and Table 6.23 respectively.

The overall accuracy is 0.90

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-Adaptation</td>
<td>861</td>
</tr>
<tr>
<td>Re-Adaptation</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 6-22 Confusion matrix of random forest for equalized target values
<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>f1-score</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Re-Adaptation</td>
<td>0.92</td>
<td>0.85</td>
<td>0.89</td>
<td>1013</td>
</tr>
<tr>
<td>Re-Adaptation</td>
<td>0.88</td>
<td>0.94</td>
<td>0.91</td>
<td>1235</td>
</tr>
<tr>
<td>Avg/Total</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>2248</td>
</tr>
</tbody>
</table>

Table 6-23 Classification report of random forest for equalized target values

As it can be seen from the confusion matrix and classification report ratings, on the data with an equalized target classes, the model detected the re-adaptation events with a better accuracy. The overall accuracy of the model shows a reduction.
7 ANALYSIS AND DISCUSSION

This chapter aims to analyze the distribution of the most important features in the data set. A comparison is done between the one-minute windows with re-buffering/bitrate re-adaptation and the one-minute windows without re-buffering/bitrate re-adaptation. Cumulative Distribution Function (CDF) of the data sets is plotted for comparison. To make a fair comparison between the data sets, the number of one-minute windows with and without re-buffering/bitrate re-adaptation is made to be equal.

7.1 Data Analysis for Detection of Re-Buffering Events

In the detection of re-buffering, Rs has the highest feature importance. Additionally, 10RP6, 10RP5, and 90 and 75 percentile of inter-arrival time have also shown higher feature importances. The CDF of these features are given below.

![Figure 7-1 Rs Cumulative distribution function](image)

![Figure 7-2 10RP6 Cumulative distribution function](image)
The CDF plot of $R_S$ shows that for the one-minute windows that doesn’t have re-buffering, around 99 percent of the values lays between 0 and 0.1. On the contrary, the one-minute windows with re-buffering have 60 percent of the $R_S$ value greater than 0.1. This indicates that re-buffering occurs when most of the download throughput per second values within the one-minute window are less than or equal to 1000 kbps. This agrees with the way a YouTube video behaves practically. It is observed that, usually re-buffering happens when the download throughput per second value is less than 1000 kbps. If this value is above 1000 kbps, the variation will not usually cause re-buffering because of the fact that the YouTube player adapts the resolution of the video to suit the existing download throughput.

The CDF plot of $^{10}R_{P5}$ and $^{10}R_{P5}$ generally show a higher value for one-minute windows that do not exhibit a re-buffering. This indicate that low value of download throughput in the last 20 second of the previous minute leads to occurrence of re-buffering event in the current minute.
The 75 and 90 percentile of the server packet inter-arrival times are generally higher in the condition where there is re-buffering. This suggests that there is slower arrival of packets from YouTube server for the one-minute windows that suffer from re-buffering. In the case of inter-arrival time 75 percentile, the one minute windows with re-buffering has larger number of inter-arrival times that are less than $10^{-7}$ seconds. This can be explained by the scenario that happened after the recovery of the re-buffering events. Re-buffering events happen when there is an outage on the network which causes the buffer of the YouTube player to be empty. After the recovery of the re-buffering event, in order to re-fill the emptied YouTube player buffer, packets from YouTube server have to arrive faster than the normal condition.

7.2 Data Analysis for Detection of Bitrate Re-Adaptation

In the estimation of bitrate re-adaptation events, download throughput per 10 second values hold higher feature importance. Additionally, low relative download throughput, and ΔR 75 percentile appear to be the other important features. The CDF of these features will be discussed in this sub-section. In the interest of comparison, the CDF of all the download throughput over 10 second values are given below.

![Figure 7-6](image1.png)

**Figure 7-6** $10^{R_{P4}}$ Cumulative distribution function

![Figure 7-7](image2.png)

**Figure 7-7** $10^{R_{P5}}$ Cumulative distribution function
Figure 7-8 $^{10}R_{P6}$ Cumulative distribution function

Figure 7-9 $^{10}R_{C1}$ Cumulative distribution function

Figure 7-10 $^{10}R_{C2}$ Cumulative distribution function

Figure 7-11 $^{10}R_{C3}$ Cumulative distribution function
As it can be seen on the CDF plots, the one minute windows that don’t suffer from bitrate re-adaptation seem to have a download throughput per 10 second values that mostly lay between 0 up to 40 Megabits per 10 second. This trend is seen to be consistent for all the values in the previous and current one-minute window. On the other hand, the download throughput per 10 second values of the one-minute windows with bitrate re-adaptation goes on increasing as it moves from $^{10}R_{P4}$ to $^{10}R_{C6}$.

The three download throughput per 10 second values from the previous minute, i.e. $^{10}R_{P4}$, $^{10}R_{P5}$, and $^{10}R_{P6}$, are higher when there is no re-adaptation and lower when there is re-adaptation. For these three values, the separation between the values when there is re-adaptation and no re-adaptation is clear. Accordingly, it is easier for the learning algorithm to identify between the two classes which make these three features to have higher feature importance. On the other hand, the download throughputs per 10 second values of the current minute are easily separable at the beginning and end of
the minute. The values at the middle of the minute, i.e. $^{10}R_{C3}$ and $^{10}R_{C4}$, have higher overlap between the one-minute windows with and without re-adaptation. This makes the middle values to have lower feature importance.

In the Appendix of this thesis, the CDF plots of download throughput per 10 second values are given by separating the cases of the one-minute windows that have only re-adaptation down, re-adaptation up, and re-adaptation up and down together. The variation trend of the values can easily be seen with the separated plot. Additionally, the reason explained above for one feature being more important than the other in the detection of re-adaptation is clearer on the CDF plots that are drawn separating the different re-adaptation cases.

The CDF plots of $R_S$ and $\Delta^1R$ 75 percentile are given in Figure 7.15 and Figure 7.16.

For the one-minute windows without re-adaptation, the CDF plot of $R_S$ shows that around 98% of the values lay between 0 and 0.1. On the contrary, the one-minute windows with re-adaptation showed a higher value of $R_S$.

From the CDF plot of $\Delta^1R$ 75 percentile, it can be seen that the one minute-windows with re-adaptation has a lower $\Delta^1R$ 75 percentile than the one-minute windows without re-adaptation. This is due to the adaptive streaming nature of YouTube video. Under normal condition, YouTube traffic flows in such a way that there is a high data flow for a certain duration of seconds which is followed by a period where there is no data from YouTube. This makes the difference between the download throughputs per second values to be high. Bitrate re-adaptations usually occur due to a disturbance on the network, especially for the one-minute windows that contain adaptation down. Under such disturbances, traffic from YouTube server does
not follow the normal traffic flow trend, which is peak data flow session followed by no data flow session, but rather traffic flows continuously from YouTube to fill the player buffer that is emptied by the disturbances. Accordingly, the difference between download throughput values of two consecutive seconds will be reduced.
8 CONCLUSION AND FUTURE WORK

8.1 Conclusion

This thesis attempted to develop machine learning models for monitoring quality of YouTube video from the analysis of encrypted video traffic. Network layer and application layer data are collected under a controlled lab setup. Hierarchical token bucket filter traffic shaping is applied to simulate disturbances on a network. The aim of the developed models is to detect re-buffering and bitrate re-adaptation events within a one-minute duration.

Four well known machine learning algorithms named as logistic regression, K-nearest neighbor, support vector machine and random forest are trained to develop the models. The detection performance of the developed models is evaluated using different evaluation metrics from the python scikit-learn module.

The occurrence of re-buffering events within a given one-minute window is seen to be correlated with network performance metrics like inter-arrival time between packets from YouTube server, download throughput values over one second and over ten second, low relative download throughput, and the difference between consecutive download throughput over one second values. Occurrence of bitrate re-adaptation events is also seen to be correlated with the above performance metrics with the exception of inter-arrival times between packets from YouTube server.

In the detection of re-buffering events, for a given one-minute window, the low relative download throughput is observed to be the most important feature. Additionally, inter-arrival times between packets from YouTube server and download throughput over 10 second values from the previous minute are also seen to be important features for detecting re-buffering events within a given one-minute window. On the contrary, download throughput over 10 values of the current and previous minute windows are seen to be important in the detection of bitrate re-adaptation events within a given one-minute window.

Random forest is seen to perform better in detecting both the re-buffering and re-adaptation events. It detected the re-buffering and bitrate re-adaptation events with an f1-score accuracy of 67 and 70 percent respectively. Logistic regression is the algorithm that has detected both the re-buffering and the bitrate re-adaptation events with the least accuracy.

From the findings of this thesis, it is possible to conclude that re-buffering events are mainly correlated with the performance metrics that are derived from download throughput and inter-arrival times between packets from the YouTube server. On the other hand, re-adaptation events are seen to be correlated with different variants of download throughput values. With the scope of the data analyzed in this thesis, random forest has shown the best performance in detecting both the re-buffering and the bitrate re-adaptation events.

8.2 Future Work

The scope of future work that can be done with regard to data collection, estimation of re-buffering events, and estimation of re-adaptation events includes:
• Collecting the network level and application level data from a real world
network has an advantage in that the buffering and bitrate adaptation
events are realistic. But this advantage is gained at the cost of controlled
environment that has to be created in a lab setup. Accordingly, a lab setup
with an advanced traffic shaping pattern that can better emulate the real
world network will be a good compromise that can be done in the future
while collecting data.
• The detection accuracy of the re-buffering and bitrate re-adaptation events
can be improved in the future.
• Estimating the duration and number of buffering events is the other future
work that can be done.
• The number of bitrate re-adaptation events can also be estimated in the
future.
ANSWERING RESEARCH QUESTIONS

1. **How are buffering and bitrate adaptation events correlated with network traffic variations for encrypted video traffic?**

   For a given one-minute window, when most of the download throughput over one second values are less than 1000 kbps, there is a high chance for the occurrence of re-buffering events. Low values of download throughput over 10 second in the last 10 second of the previous minute leads to occurrence of re-buffering events in the current minute. Generally, when the inter-arrival time between packets from YouTube server increases, there is a high probability for the occurrence of re-buffering events within a given one-minute window.

   The bitrate re-adaptation events are highly influenced by the different variants of the download throughput values. Low values of download throughput over 10 second in the last 20 second of the previous minute lead to the occurrence of bitrate re-adaptation events in the current minute. Additionally, when most of the download throughput over one second values in a given one-minute window are less than 1000 kbps, there is a high chance for the occurrence of the bitrate re-adaptation events.

2. **Which traffic features contribute most to the detection of buffering and bitrate adaptation events of encrypted video traffic?**

   In the detection of the re-buffering events, low relative download throughput is the most important feature and it is highly correlated with the occurrence or non-occurrence of re-buffering events. The download throughput over 10 second value in the last 10 second of the previous minute, i.e. $10R_{P6}$, is the second most important feature in detecting the re-buffering events. The 90 and 75 percentile of packet inter-arrival time from the YouTube server took the next ranks in the features importance list while detecting re-buffering events.

   In detecting the bitrate re-adaptation events, download throughputs in the last 20 seconds of the previous minute take the highest feature importance. The low relative download throughput also has significant contribution in detecting the bitrate re-adaptation events.

3. **Which machine learning model is better in estimating buffering and bitrate adaptation events using features extracted from encrypted video traffic?**

   With the scope of the data that is analyzed in this work and in comparison to the learning algorithms that are trained in this thesis, random forest has shown the best performance in detecting both the re-buffering and the bitrate re-adaptation events within a duration of one-minute window.
APPENDIX A

A.1 CDF plot of One-minute Windows with Adaptation Down and No Adaptation

![CDF plot of One-minute Windows with Adaptation Down and No Adaptation](image-url)

**Fig A.1** $\text{R}_{P4}$ Cumulative Distribution Function

**Fig A.2** $\text{R}_{P5}$ Cumulative Distribution Function

**Fig A.3** $\text{R}_{P6}$ Cumulative Distribution Function
Fig A.4 $^{10}R_{C1}$ Cumulative Distribution Function

Fig A.5 $^{10}R_{C2}$ Cumulative Distribution Function

Fig A.6 $^{10}R_{C3}$ Cumulative Distribution Function

Fig A.7 $^{10}R_{C4}$ Cumulative Distribution Function

Fig A.8 $^{10}R_{C5}$ Cumulative Distribution Function
A.2 CDF plot of One-minute windows with Adaptation UP and One-minute Windows with No Adaptation

---

**Fig A.9** $10^6$RCS Cumulative Distribution Function

**Fig A.10** $10^6$RP4 Cumulative Distribution Function

**Fig A.11** $10^6$RP5 Cumulative Distribution Function

**Fig A.12** $10^6$RP6 Cumulative Distribution Function
Fig A.13 $^{10}R_{C1}$ Cumulative Distribution Function

Fig A.14 $^{10}R_{C2}$ Cumulative Distribution Function

Fig A.15 $^{10}R_{C3}$ Cumulative Distribution Function

Fig A.16 $^{10}R_{C4}$ Cumulative Distribution Function
Fig A.17 $^{10}R_{C5}$ Cumulative Distribution Function

Fig A.18 $^{10}R_{C6}$ Cumulative Distribution Function

A.3 CDF Plot of One-minute Windows with Both Adaptation Up and Down, and One-minute windows with No Adaptation

Fig A.19 $^{10}R_{P4}$ Cumulative Distribution Function

Fig A.20 $^{10}R_{P5}$ Cumulative Distribution Function
**Fig A.21** $^{10}R_{P6}$ Cumulative Distribution Function

**Fig A.22** $^{10}R_{C1}$ Cumulative Distribution Function

**Fig A.23** $^{10}R_{C2}$ Cumulative Distribution Function

**Fig A.24** $^{10}R_{C3}$ Cumulative Distribution Function
Fig A.25 $^{10}R_{C4}$ Cumulative Distribution Function

Fig A.26 $^{10}R_{C5}$ Cumulative Distribution Function

Fig A.27 $^{10}R_{C6}$ Cumulative Distribution Function
REFERENCES

[34] H. Bert, ‘Linux Advanced Routing & Traffic Control HOWTO’.