IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

# Threshold-oblivious on-line web QoE assessment using neural network-based regression model

ISSN 1751-8628
Received on 19th November 2019
Revised 21st March 2020
Accepted on 14th April 2020
doi: 10.1049/iet-com.2019.1229
www.ietdl.org

*Enge Song[1], Tian Pan[1,2] ✉, Qiang Fu[3], Rui Zhang[1], Chenhao Jia[1], Wendi Cao[4], Tao Huang[1]*

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, People's Republic of China*

[2]*Purple Mountain Laboratories, Nanjing 211111, People's Republic of China*

[3]*School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand*

[4]*Peking University, Beijing 100871, People's Republic of China*

✉ *E-mail: pan@bupt.edu.cn*

**Abstract:** The evaluation of the web-browsing quality of experience (QoE) is difficult to complete through traditional methods (e.g. deducing formulas or setting thresholds) due to the diversity of websites and their contents. To evaluate web-browsing QoE through a general way, the authors propose a web QoE evaluation architecture based on machine learning, consisting of two parts: traffic classification sub-system and QoE prediction sub-system. When evaluating user experience, traffic classification sub-system first classifies the packets generated by visiting a website into a flowthrough some fields in the packet header, to model each website separately. The traffic classification accuracy of $> 2000$ packets over six websites reaches 96.63%. Then, in the network layer, the traffic metric cumulative traffic volume is generated from the size and arrival time of packets. When a user visits a web page, their regression model predicts the above-the-fold time (ATF) and thus QoE. The output of the regression model is an exact ATF value that is mapped to user experience. In addition, reversing input variables further improves the model, which is evaluated on two popular websites. The QoE prediction results of the improved method for 5400 visits are obtained within 0.0975 s, reaching 0.9 $R^2$ score.

## 1 Introduction

A study has found that interactive hypertext transfer protocol (HTTP) traffic once again dominated residential broadband internet traffic, accounting for $> 50\%$ of the traffic [1] and it gradually becomes the de facto narrow waist of the internet. People often visit varieties of websites at work or in their precious leisure time, including search engines, video sites, and social networking sites. Whether the websites can be loaded successfully within a short time is of crucial importance. Also, predictably, even tiny network delay of page loading can ruin the user experience. The longer users have to wait for the web page to load, the more likely they are to be unsatisfied with the service [2].

Akamai Technologies, Inc. found the correlation of e-commerce website performance and an online shopper's behaviour [3]. The experiment of 1048 online shoppers demonstrates that 2 s is a new threshold, after which the consumers would become impatient, and if the pages still fail to load 40% of consumers will wait no more than 3 s. According to a 2015 article [4], Amazon has calculated that only 1 s of page loading would cost $1.6 billion in sales a year. Google has calculated that the reduction of search results by mere four-tenths of a second could lead to the loss of 8 million searches per day, which means the loss of millions of online ads. Therefore, the slow page loading causes not only inconvenience to the users but also losses to internet content providers.

Quality of service (QoS), which is aimed at helping the information and communication technology (ICT) engineers develop their products and provide better services, is widely used in measuring the network's capability [5, 6]. To the end, diverse traffic monitoring and load balancing approaches [7, 8] are proposed to reduce network congestion, especially in data centre networks. However, the relation between user experience and QoS is not a simple positive correlation, which means high QoS does not necessarily lead to a perfect user experience. To obtain a more accurate user experience, the concept of quality of experience (QoE) is concerned as an alternative to QoS by the ICT industry and internet service providers (ISPs) in recent years. ISP and device vendors can have an estimation of the current user experience anywhere in the network by using device traffic to create a hypothetical QoE prediction model [9–15].

However, it is very difficult to evaluate the web-browsing QoE in real-time on a running network. Monitoring network-level performance criteria are easier and more usual. So the problem is how to correlate the network-level QoS to the QoE perceived by the users. Casas *et al.* [16] predicted the YouTube stalling events with network-layer data alone, by calculating the estimated buffer size with several formulas and comparing it with the empirical thresholds. Their system could only evaluate the user experience of YouTube because excessive specific thresholds and formulas were used for estimating QoE accurately [16].

Web-browsing QoE mainly depends on the above-the-fold time (ATF), i.e. the loading time of content that the screen directly displays [17, 18]. Generally, the longer the ATF is, the worse the QoE will be. Numerous studies that predict web browsing QoE are based on ATF or its variants [19–22]. Although predicting QoE through ATF works well, getting ATF is not an easy task since ATF is a subjective metric related to the web page loading. Most existing methods get ATF through analysis of the video of page loading. This mode, however, carries many disadvantages: (a) running on the client-side requires user's cooperation to capture video of page loading; (b) video analysis takes a great deal of time; (c) it takes up network bandwidth and storage space.

In such a situation, we introduce a data-driven web-browsing QoE prediction system, which is based on machine learning (ML) and can predict the web QoE on the gateway side. This enables close collaboration between ISPs and content providers to serve their clients. Our system consists of the traffic classification sub-system and the QoE prediction sub-system. The traffic classification sub-system can classify packets into flows belonging to different websites so that we can independently model each website. It was declared in [23] that about 90% of the network traffic was a small number of large-sized elephant flows, while the remaining 10% was a large number of small-sized mice flows. We can filter out the mice flows and then only predict QoE of elephant

flows. After that, we get information from transmission control protocol (TCP) flows occurring when the user visits a website, and then characterise the traffic by the proposed traffic metric: cumulative traffic volume (CTV). CTV has different forms in different network conditions. Also, there will be different ATFs in different network conditions. Generally, the smaller the ATF, the better the network condition and the user experience. Therefore, we can predict ATF by identifying the forms of CTV. The prediction can be done with little application-layer data, which proves that our system can be deployed on the gateway. After training, the regression model in the QoE prediction sub-system can get the exact ATF prediction value, such as ATF = 1.38 s. Finally, ATF can be mapped to the user experience QoE by the mapping function or expert model [21]. In the QoE prediction sub-system, we propose three supervised learning models for ATF prediction instead of video analysis. The trained model takes up extremely little storage space, with a minimum of 18 kB and a maximum of 1 MB. Furthermore, the system works efficiently and can predict ATF in a very short time with high accuracy. Specifically, when our model predicts 5400 unknown samples within 0.0975 s, the $R^2$ score is more than 0.9 ($R^2$ score = 1 means no error). In other words, using a computer like ours, QoE prediction of a flow can be completed within 0.00018 s, demonstrating that our system can predict QoE online.

Our major contributions are summarised as follows:

• We have proposed an ISP-level general method that requires little application-layer data and takes up extremely little storage space, allowing equipment vendors and ISPs to deploy our model on network intermediate devices, to evaluate the QoE when users visit a web page. In particular, our approach is data-driven, which does not require a manual parameter setting, and we can update the model in real-time by adding new data to adapt the model to the changing website contents (Section 3).

• We have designed the traffic classification sub-system that can classify different website packets based on various fields in the header. The system can classify > 2000 packets belonging to six websites with an accuracy of 96.63% (Section 4).

• We have developed three models based on our architecture to predict the exact ATF value, e.g. ATF = 0.38 s. Since the ML method does not need formula deduction or threshold setting according to the website content as traditional methods do, our method is content-independent and all parameters for websites or contents can be trained with one unified model (Section 5).

## 2 Related work

### 2.1 Traffic classification

The process of categorising packets into ‘flows’ in an internet router is called packet classification. Gupta and McKeown [24] described the algorithms representing each category and discussed which type of algorithm might be suitable for different applications. Pan *et al*. [25] implemented a high-speed traffic classification system based on application-layer patterns with field-programmable gate array. It was declared in [23] that about 90% of network traffic is a small number of large-sized elephant flows and the remaining 10% is a large number of small-sized mice flows. After classifying the packets into flows, we can filter out the mice flows. Therefore, only the user experience corresponding to the elephant flow will be analysed. Identifying elephant flows is of great importance in developing effective and efficient traffic engineering schemes. On the basis of Bayes' theorem, Mori *et al.* [26] identified elephant flows in periodically sampled packets. First, they introduced a prior distribution of the packet number of each flow. Next, they found the number of sampled packets of flow was greater than expected. Eventually, the flow was identified as an elephant flow.

The methods above can filter out mice flows on the Internet. Inspired by these algorithms, we have designed our packet classification sub-system.

### 2.2 Traditional methods

QoE, which can intuitively reflect the user perception of loading a web page, is difficult to quantify, so the research community is committed to finding an effective QoE evaluation method, hoping to improve the user experience through precise QoE evaluation. Three passive probing methods were put forward by Schatz *et al.* [27] that used network-layer data to predict stalling patterns through formula deductions when users are watching YouTube videos. They calculated the total stalling time $T$, the number of stalling events $N$, and the duration or length of a stalling event $L$ based on numerous formulas. Using data from the network layer, they calculated the stalling events according to substantive formulas and empirical thresholds derived from large measurement activities. Staehle *et al.* [28] proposed a client tool to monitor YouTube traffic at the application layer, which could make predictions of the pause in the video by estimating the size of the playback buffer. Casas *et al.* [16] introduced an approach to measure YouTube pause at the network layer. With various formulas, they estimated the size of the YouTube playback buffer. They then predicted the video's pause by comparison of the empirical threshold and estimated buffer size. Bagga *et al.* [29] studied how service consumers could benefit by selecting the appropriate Web Service based on QoS through multi-criteria decision-making (MCDM). They tended to assist the researchers in two conducts: firstly, observe the performance of different MCDM methods for plenty of alternatives and attributes. Secondly, perceive the possible deviation in the ranking from these methods. HoBfeld *et al.* [30] discussed the impact of memory effect on the web QoE modelling. They presented a QoE model for web traffic, which took into account the memory effect and temporal dynamics. Then, three different QoE models were proposed based on the memory effect to improve basic models.

All the above studies predict the QoE by formula deductions and threshold settings, which does not apply to our issue. Their mathematical formulas and thresholds increase complexity and inflexibility, weakening universality to some extent. So, for solving generalised issues, fewer formulas or thresholds should be introduced. Inspired by recent advancements in classification problems and network problems of ML, we consider building systems to evaluate user experience with multitudes of labelled data.

### 2.3 ML methods

Casas *et al.* [31] proposed an unsupervised network intrusion detection system (UNIDS), which detected unknown network attacks without labelled traffic or training. UNIDS employs a novel unsupervised outlier detection method based on sub-space clustering and multiple evidence accumulation techniques to identify different network intrusions and attacks. However, it is hard to apply clustering to the classification of time series due to the large dimensions of the latter. Ben-Letaifa [14] analysed the impact of network behaviour on web service quality and proposed the QoE tool based on prediction algorithms and deep learning. The data set they used encompassed personal attributes (age, genre, and look), web page attributes (site type, content, and Index of site content), and network conditions (downstream speed, bandwidth, and delay). They obtained the merged dataset using the framework Pandas of Python. Then, six common ML algorithms were trained with the dataset through the framework Keras of Python. The model with the highest accuracy was gradient boosting, with an accuracy of 0.66087. However, the model cannot run on the gateway side because it requires user attributes that are difficult to obtain on the gateway side. Mushtaq *et al.* [32] analysed the impact of parameters such as QoS on video streaming service QoE, and assessed how ML methods could help establish a precise and objective QoE model that connected low-level parameters with high-level user experience. Luo *et al.* [33] presented a data-driven scheme for industrial Internet of things QoS loss prediction based on kernel least mean square (KLMS). They found the relevant QoS values for every known QoS entry from similar service users and web service items. Then, using the KLMS, they delved the relationships between all the known QoS data and their

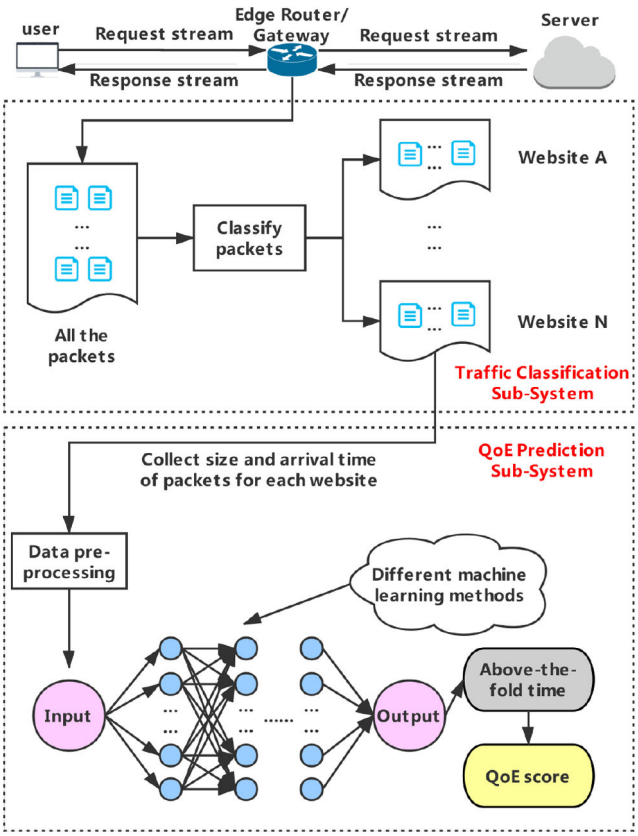


**Fig. 1** *System architecture*

corresponding highest-similarity QoS data. Wang *et al.* [34] proposed a data-driven architecture that enhanced the personalised QoE of 5G networks. Zhang *et al.* [35] proposed a long-term QoS forecasting approach based on neural networks, but their work focused on long-term QoS prediction. Liu *et al.* [36] proposed a collaborative QoS prediction framework for privacy preservation, which could guarantee the accurate QoS prediction while protecting users' privacy.

These studies did not use time series analysis to extract meaningful features from the data. However, their work demonstrates the possibility of using ML algorithms for QoE analysis.

### *2.4 ATF and its variants*

Sackl *et al.* [37] defined new key performance indicators, capturing the impact of bandwidth fluctuations on the user experience through subjective laboratory research. Their research suggests that web browsing QoE is less sensitive to interruptions as users do not always detect interruptions. Page loading time (PLT) used to be a popular metric for web-browsing QoE prediction, but it is not the perfect method for user-perceived PLT (UserPerceivedPLT). First, the user may only care about the loading time of the visible portion, which means the PLT may be overestimated. Second, the script may continue to load objects after the OnLoad fires, leading to the underestimation of PLT [20]. A new metric proposed by Google is ATF time [19], defined as the rendering time of the visible portion of the web page. After that, ATF and its variants are used by many researchers for web-browsing QoE prediction. Varvello *et al.* [20] argued that the existing PLT indicator was merely a partial reflection of the participant's experience, and they proposed a new indicator to reduce the gap between OnLoad and UserPerceivedPLT. Da-Hora *et al.* [21] made a comprehensive assessment of the performance of the user experience prediction, using all indicators (PLT, ATF, and its variants such as IIATF) through expert models (ITU-T, IQX etc.) and different ML algorithms. Their results indicated that the best correlation with MOS was IIATF (0.85), while PLT ranked seventh (0.81). These results have confirmed the possibility of using ATF time to represent UserPerceivedPLT.

All the above methods estimate ATF by analysing the video loaded on the page, which is more accurate but requires more time and resources. In such a context, we propose our system that can predict ATF with a large amount of network-layer data and a small amount of application-layer data. Although it cannot achieve fine-grained results as the video analysis method does, our approach performs outstandingly well on coarse-grained prediction.

In this work, for web-browsing QoE prediction, we predict the continuous ATF instead of the discrete ATF as we did in the previous poster version of WebQMon.ai [38]. To predict QoE by the ML model, we have implemented a traffic classification sub-system to aggregate packets of the same web page. The most significant difference between the two articles is that one adopts a regression model, while the other adopts a classification model to solve the problem. Regression model means the more accurate prediction of ATF, which could predict the exact value of ATF, e.g. ATF = 1.38 s, while WebQMon.ai could only predict whether ATF falls in a certain range, e.g. < 3 s, so the regression model is more conducive to further prediction of user experience.

## 3 Architecture

### *3.1 System architecture*

Web browsing QoE is largely determined by the ATF that the user experiences. To this end, we put forward our system for ATF prediction of different web pages. As shown in Fig. 1, network-level data is collected from the Edge Router or Gateway. The traffic classification subsystem distinguishes packets from different websites as depicted in Fig. 1. Then, raw data is transformed into a useful format as shown in the QoE prediction sub-system in Fig. 1. After that, the processed data is used for the training of different ML models, such as fully connected neural networks and long-short-term memory (LSTM) neural networks. Different prediction models are essential since different webpages have different contents and attributions. Finally, the predicted ATF is obtained, which can be mapped to the web-browsing QoE through existing functions [21]. In Section 5, to predict web-browsing QoE, we propose three methods based on our architecture, using different input variables and ML algorithms.

For traditional methods, different mathematical formulations or thresholds have to be set for different sites due to the diversity of web content. However, machine learning (ML) models can be trained and/or updated with data reflecting the diversities of contents so that different/new contexts can be learned and adapted.

### *3.2 Dataset and data pre-processing*

*Traffic classification.* Data, the basic elements of ML, determines the quality of the model to some extent. For maximisation of the prediction accuracy, prediction models are built for each site. Specifically, video sites have a great number of images with few texts, while new sites have substantial texts. Different contents inevitably lead to different numbers and sizes of requests and responses, resulting in the failure to use the same model to predict ATF accurately. The purpose of the traffic classification sub-system is to obtain data for each website separately, which lays a solid foundation for subsequent data analysis.

Our system is trained and tested with the dataset captured from TCP flows appearing when the user visits a page. All the TCP traffic can be easily obtained at the Edge Router or Gateway as depicted in Fig. 1. The traffic classification sub-system can aggregate related packets into a cluster by multiple iterative classifications through the packet headers. Eventually, each cluster represents the packets of the same websites that the user visits. The specific classification algorithm will be detailed in Section 4.

*Dataset.* Through the traffic classification sub-system, the traffic generated by visiting a page can be aggregated into a cluster. Our system works with the size and arrival time of packets, which have a direct bearing on the network conditions. Specifically, TCP segments reassembling and application-layer data delving are unnecessary since the contents of TCP traffic are not in consideration. Obviously, the packets arrive quickly when the network is in good condition (high download speeds, low latency,

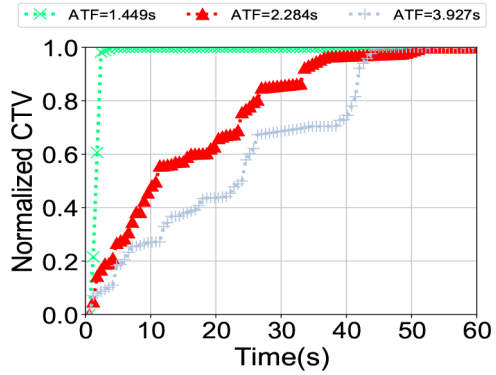


**Fig. 2** *Cumulative traffic volume (CTV)*

**Require:** $packet$
**Ensure:** $target\ URL$
1: **if** $Method = CONNECT$ **then**
2: $U = request\ address$
3: **else**
4: **if** $Referer$ in $packet\ header$ **then**
5: $U = Referer$
6: **else**
7: $U = request\ address$
8: **end if**
9: **end if**
10: **return** $U$

**Fig. 3** *Algorithm 1: GET URL*

and no packet loss). Otherwise (low download speeds, high latency, or severe packet loss), they arrive slowly. We get ATF by a Chrome plugin (named Approximate ATF, downloaded from Chrome web store in June), which can estimate the ATF time of page loading by website content analysis. The ATF time we used, such as 4.325 s, has four digits after the decimal point so that the predicted ATF problem can be modelled as a regression problem. The plugin needs to work with the browser, i.e. it has to be deployed on the client-side. In contrast, our system runs on the server-side, which helps ISPs and webmaster provide better service. Next, the collected data will be converted into the input and output of the ML model through the data pre-processing module.

*Data pre-processing.* Traffic characteristics at packet level are remarkably diverse, which makes it difficult to extract statistically meaningful features. On the other hand, the number and total size of packets generated by one request may vary significantly from page to page. A fixed dimension, however, is required for the input of the ML model, which prevents us from directly using the unprocessed data for training and prediction. To this end, a traffic metric is proposed to represent the features of TCP flows, which can reflect the network conditions and help to predict the web-browsing QoE. The raw data will be processed into the proposed traffic metric by statistical data processing methods [39]. Through testing, we find that every form of traffic metric in time series may correspond to different network conditions and ATFs. Therefore, ATF can be predicted by distinguishing these different forms of traffic metric. In this study, the traffic metric that we will be investigating is CTV as detailed below.

CTV Accumulation is a commonly used method in statistical analysis. It is impractical to measure traffic with packet-level granularity, which is statistically meaningless. Statistical features can be extracted by cumulative packet traffic. Fig. 2 shows the normalised CTVs under excellent, not good, and terrible network conditions, representing low, medium, and high ATF, respectively. Different network conditions are controlled by Dummynet. We adjust the delay so that it corresponds to excellent, not good, and terrible network conditions from small to large. As shown in the figure, the curve exhibits a steep slope when the network is in excellent condition. With the deterioration of network conditions, the curve gradually flattens out. Different forms of CTV, representing different ATFs, validate the practicality of ATF prediction by distinguishing traffic metric forms.

### *3.3 Training and prediction*

The input variables of ML models are obtained from CTV through feature engineering methods [40], which will be elaborated in Section 5. The output variable is ATF captured by the chrome plugin mentioned above. A pair of an input value and its corresponding output value is called an ordered pair, written as (a, b). The pairs are used for training and testing. In the training process, the predicted value is obtained by matrix operation, and then the difference between the label and the predicted value is reduced by iteration. The training modifies the weights of neural networks, enabling the model to predict ATF well. In the prediction process, only simple matrix operations are needed to obtain the result, ATF. After that the ATF can be mapped to the user QoE by mapping function or expert model [21].

## 4 Traffic classification sub-system

When surfing the web, users often visit plenty of pages, inevitably generating a great number of requests and responses. Our model can predict the web-browsing QoE for each website separately, which requires classification of the mixed packets into different clusters based on their source websites. This is the objective of our traffic classification sub-system. The system can classify packets belonging to the same website into the same cluster, which lays a foundation for predicting the ATF of each website by ML algorithms at a later stage. The traffic classification sub-system has three parts: first, second, and third classification. The system can aggregate 96% of the packets into the target flow.

*Indirect access.* When users visit a web page, a phenomenon called indirect access usually occurs. That is to say, many websites have pictures and other multimedia resources that need to be retrieved from other servers. So, the user sends a request to the server *w*, after which the server *w* sends a request to another server *s*, and finally the server *s* sends a response. The domain name in the first line of the HTTP header is the server address requested by the packet, which is called the request address. However, we cannot classify packets by request address alone due to the indirect access phenomenon, which will lead to coarse-grained classification results and failure to classify the packets belonging to flow into a cluster. Our system has used multiple fields for iterative classification to reduce errors caused by the indirect access. The mainly used fields which contain the string associated with the request uniform resource locator (URL) are as follows: request address (the domain name in the first line of the HTTP header), Referer, and Host.

*GET URL.* HTTP defines a set of request methods to indicate the required operations for a given resource, such as 'GET', 'HEAD', and 'CONNECT'. The packets with different request methods usually have different fields, so multiple fields are needed to ensure that all requested methods are covered. By analysis of the relationship between the fields and the request method, we have reached the following conclusions: (i) there must be a request address and a Host in all packet headers. (ii) The Referer field exists only in the header of a packet whose request method is not 'CONNECT', not in all the headers. Furthermore, by parsing the contents of the packet header, we find that the source URL is the most common in the Referer field, followed by the request address. Specifically, the Referer field matches the source URL in the case of indirect access. Based on the above assumptions, we propose an algorithm to extract the target field from each packet header. It is based on the Referer field, and the request address field is used as a supplement.

Algorithm 1 (see Fig. 3) shows how to obtain the target URL from the packet header. For each packet, we make $U$ equal to the request address using the request method of 'CONNECT' (line 1-3). Then, we look for the Referer field in the header of the packet whose request method is not 'CONNECT'. If the request method is not 'CONNECT', we will set $U$ equal to the Referer field; if not,

**Require:** $packets$
**Ensure:** $1^{st}$ $classification$ $results$
1: $cluster\_sets = []$
2: **for** $packet$ in $packets$ **do**
3: $U =$ GET URL($packet$)
4: **if** $U$ in $cluster\_sets$ **then**
5: classify $packet$ as the matched cluster.
6: **else**
7: create a new cluster and label it $U$.
8: classify $packet$ as this cluster.
9: **end if**
10: **end for**
11: **return** $cluster\_sets$

**Fig. 4** *Algorithm 2: first classification*

**Require:** $packets$, $1^{st}$ $classification$ $results$
**Ensure:** $2^{nd}$ $classification$ $results$
1: sort $cluster\_sets$ by number of packets from small to large.
2: $N =$ number of $cluster\_sets$
3: **for** $i$ from 1 to $N-1$ **do**
4: **for** $j$ from $N$ to $i+1$ **do**
5: $target =$ label of $cluster[j]$
6: **for** $packet$ in $cluster[i]$ **do**
7: **if** find $target$ in $request\ address$, $Referer$, $Host$ **then**
8: $cluster[j] = cluster[i] + cluster[j]$
9: $cluster\_sets = cluster\_sets - cluster[i]$
10: go to $line$ 3
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: **return** $cluster\_sets$

**Fig. 5** *Algorithm 3: second classification*

we will set $U$ equal to the request address (line 4-8). After that, $U$ will be returned as the target URL.

*First classification.* Algorithm 2 (see Fig. 4) demonstrates how to use the obtained target URL for the first classification. For each packet, we will determine whether its target URL already exists in the cluster sets. If it does, classify it as the cluster, and if it does not, create a cluster and label it the target URL of this packet (line 2-10). The recursion stops when all the packets are classified, and then the current function call will return the cluster_sets as its return value (line 11).

*Second classification.* The second classification uses the results of the first classification and the relevant fields of packet header. Algorithm 3 (see Fig. 5) demonstrates how to obtain the second classification results based on the first classification. First, we need to sort the cluster sets according to the packet number in the cluster from small to large (line 1). After that we get the number $N$ of clusters in the cluster sets (line 2). For each cluster[$i$], except the largest cluster in the sets, we get the label (target) of each cluster[$j$] whose packets are more than that of cluster[$i$] (line 3-5). We loop the cluster[$j$] from a larger one to a smaller one to reduce the number of merges (line 4). For each packet in cluster[$i$], we look for target in the request address, Referer and Host fields (line 6-7) in the packet header. If the target matches one of these fields, merge cluster[$i$] with cluster[$j$], and remove cluster[$i$] from the cluster_sets (line 8-9). Then, go to line 3 to handle the next cluster[$i$] (line 10). The recursion stops when $i$ is equal to $N-1$, and then the current function call will return the cluster_sets as its return value (line 15).

*Third classification.* Similar to the second classification, the third classification uses the results of the second classification and the relevant fields in the packet header. However, unlike the second classification, the third classification needs to sort the cluster sets according to the packet number in the cluster from large to small. Note that third classification and second classification differ only in the classifier and the input data, but the algorithm flow is the same (line 2-15).

*Preliminary results.* We have captured > 2000 packets from six popular websites to test the performance of the traffic classification sub-system. The classification accuracy of first, second, third is 68.32, 81.59, and 96.63%, respectively. Through multiple iterations of classification, the accuracy has been significantly improved. 96.63% classification accuracy also enables us to independently predict the user experience of each website. Although there are only 2000 packets of six websites, we have repeated the experiment several times and have found that the classification results are stable with no significant fluctuation in accuracy. Detailed experimental results will be presented in Section 6.

## 5 QoE prediction sub-system

In this section, we will detail how ML methods can help identify different forms of CTV, as introduced in Section 3.

*Algorithm introduction.* The key challenge is the accurate distinction of traffic metrics. For this reason, we propose three ML methods: 'Slope', 'Cumulation', and 'R-Cumulation'. All the models are based on our architecture, except that each of them adopts different ML algorithms and feature engineering methods. 'Slope', based on the max slope and some features of CTV, predicts the ATF with a lightweight neural network. 'Cumulation' relies on the linear interpolation data of the CTV and uses the LSTM neural networks for ATF prediction. Having improved 'Cumulation' by reversing the input variables, we name it 'R-Cumulation', which has fast training speed and good performance in predicting ATF. All of them predict ATF based on the size and arrival time of packets.

*Training and prediction.* First, we collect TCP packets arriving within 60 s after the user visits a web page, and calculate the normalised CTV, which is used as the input of the three models later. After that, the input variables of ML models from CTV can be obtained through some feature engineering methods [40].

*Slope.* The normalised CTV is obtained through the traffic classification sub-system. The CTV curves have some features that can be mapped to the ATF, one of which is the maximum slope. We calculate the CTV slope at 1 s intervals and find the maximum slope. It can be seen from Fig. 2 that the maximum slope of the CTV curve is a good indication of ATF—the deeper the slope is, the smaller the ATF is. In addition, we define the time when normalised CTV reaches x% as $t_{x\%}$. We use $t_{10\%}$, $t_{20\%}$, $t_{30\%}$, $t_{40\%}$, $t_{50\%}$, $t_{60\%}$, $t_{70\%}$, $t_{80\%}$, and $t_{90\%}$ as classification features, as they form a nice CTV envelope in time domain. The above features are combined into a 10-dimensional feature vector as the input variable. The form of the input variable is ($t_{10\%}$, $t_{20\%}$, $t_{30\%}$, $t_{40\%}$, $t_{50\%}$, $t_{60\%}$, $t_{70\%}$, $t_{80\%}$, $t_{90\%}$, and maximum slope) as depicted in Fig. 6. We name the model Slope because the maximum slope is included in its input variables. The Slope model employs fully connected neural networks as the predictor. The predictor derives the prediction result—the predicted value. In the training process, the difference between the predicted value and actual ATF is in continuous decrease through backpropagation [41]. In the prediction process, the Slope model can obtain the prediction results in real-time through forwarding propagation.

*Cumulation.* A time series is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. We use linear interpolation to approximate 100 points of the CTV curve, which are used as input variables of the cumulation model. Hence, the input variable of cumulation is a typical time series. Therefore, we use LSTM neural networks, a variant of recurrent neural network (RNN), to deal with the problem of explosion and vanishing gradients during traditional RNN training. LSTM neural networks are well-suited to data classification, processing, and predictions based on time series due to the possible lags of unknown duration between important events in a time series.

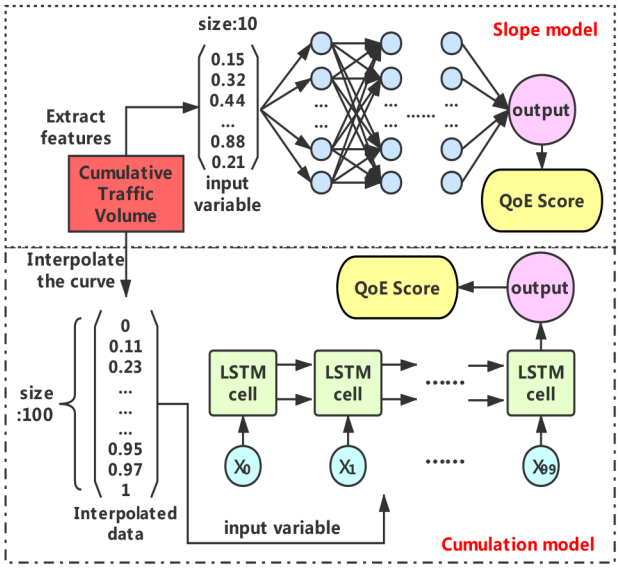


**Fig. 6** *Architecture of the slope model and the cumulation model*

**Table 1** Other model parameters

| | Number of input units | Number of hidden units | Number of hidden layers |
|---|---|---|---|
| *slope* | 10 | (9,8) | 2 |
| *cumulation* | 100 | 256 | 1 |

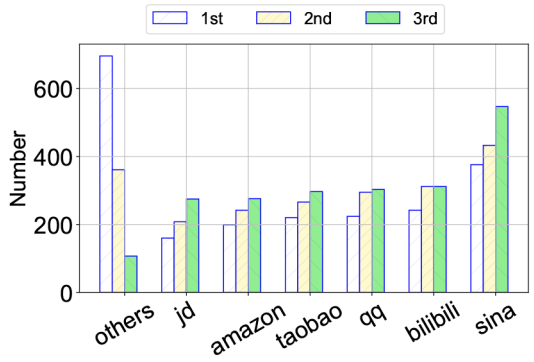


**Fig. 7** *Packets number versus iterations*

Through loop iteration, LSTM neural networks can retain all input information of the sequence and the hidden information of non-linear transformation from the starting moment to the current moment. Therefore, LSTM is particularly suitable for addressing the long delays of the linear interpolation vector. As shown in Fig. 6, the 100 approximate points are used as input variables of the LSTM neural network, and the output of the LSTM is the predicted value. We name the model Cumulation because it uses cumulative data.

*R-Cumulation.* As shown in Fig. 2, the front of the CTV curve is significantly different under different ATFs. Obviously, the packets arriving earlier within 60 s represent the initial response of the loading process, and thus may have a greater impact on ATF. The interpolated data is sequentially inputted to LSTM neural networks, and the output is generated when the last data enters, ensuring the less impact of early data on the output, which is a feature of LSTM neural networks. For this purpose, we reverse the interpolated data so that the post-end data enters first, i.e. the early packets of the CTVs are processed in the end. Therefore, the early part of interpolated data will have a profound impact on the output of the Cumulation model for better prediction. This model is named R-Cumulation because it takes the reversed interpolated data as the input variable. Note that Cumulation and R-Cumulation differ only in the input variable—the architectures are the same. We will demonstrate in Section 6 that R-Cumulation does have better performance than Cumulation.

*Summary.* All models based on our architecture are data-driven, guaranteeing that they can be updated on a regular basis with new data. QoE prediction of the different websites is quite simple because only the data of corresponding websites is needed. All the methods can predict user experience in real-time through ATF, allowing ISPs, and equipment vendors to detect poor user experience and take actions if necessary.

## 6 Implementation and evaluation

### 6.1 Experiment setup

Our measurements took place from June 2019 to August 2019 in a laboratory in China. The hardware was equipped with i5-8600K CPU and GTX 1070Ti graphics card. All models were trained with GPU-version of TensorFlow. We tested the performance of the traffic classification sub-system and QoE prediction sub-system separately. The respective experimental settings are as follows.

*Settings for traffic classification.* When a user visits multiple web pages, the packets may arrive together. The traffic classification sub-system can distinguish the packets of different websites, providing the basis for establishing a QoE prediction model for each website. We simultaneously simulated visits to six popular websites for testing: (taobao.com), (sina.com.cn), (bilibili.com), (amazon.com), (qq.com) and (jd.com). In what follows, we use 'taobao', 'sina', 'bilibili', 'amazon', 'qq' and 'jd' to refer to these websites for simplicity.

*Settings for QoE prediction.* To train the neural network model, extensive labelled data were needed. We simulated lots of visits to 'tmall.com' and 'weibo.com.cn', which ranked the fifth and 16th, respectively, in Alexa Traffic Rank [42]. In what follows, for simplicity, we use 'tmall' and 'weibo' to refer to the sites. We collected the TCP packets arriving within 60 s after visits, and got the ATF time by a chrome plugin. We used Dummynet, which could control the network condition (such as packet loss, delay, and downstream throughput), to construct samples with different ATF time. Then, we got the normalised CTV from these packets and labelled the sample according to the captured ATF time.

*Dataset.* We simulated 13,000, 18,000 visits to tmall and weibo, respectively. 70% of the data was used as the training dataset and the rest as the test dataset. For a certain website, different models shared the same training dataset and test dataset, ensuring that the results were not affected by the way of dataset segmentation. Using the training dataset, each model was trained separately.

*Model parameters.* A 'structured trial and error' method is commonly used for creating a neural network's layer approximation [43]. The architecture of models was determined by this method, so the best performance on the test dataset could be achieved by tuning these parameters. The three models have some common parameters: number of iterations = 5000, learning rate = 0.003, and batchsize = 128. Other parameters are shown in Table 1. The number of input units is the dimension of the input variable, and the number of hidden units is the number of neurons in the hidden layer. The number of hidden layers is the number of network layers between input layers and output layers. They exemplify the scale of the model. It indicates that all models are lightweight networks.

### 6.2 Evaluation

*6.2.1 Traffic classification: Number of packets per cluster.* The traffic classification sub-system needed to distinguish packets from different websites. We captured 2,080 packets from six websites and classified them by the first, second, third classification. As shown in Fig. 7, the number of unclassified packets (others) dropped from > 600 to 70. Also, the packet number of each website steadily increased by the first, second, and third classification. After the third classification, the unclassifiable packets accounted for only 3.3% of the total, which was negligible. Furthermore, all packets of different websites were correctly classified into their own clusters without classification errors.

*Classification result.* As depicted in Fig. 8, the classification accuracy is gradually increasing. The number of correctly classified packets after the first, second, and third classification is

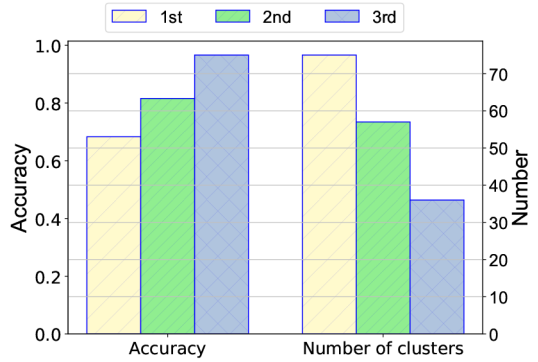


**Fig. 8** *Traffic classification results*

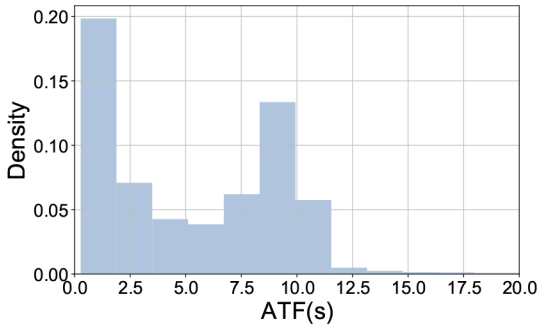


**Fig. 9** *ATF distribution of weibo*

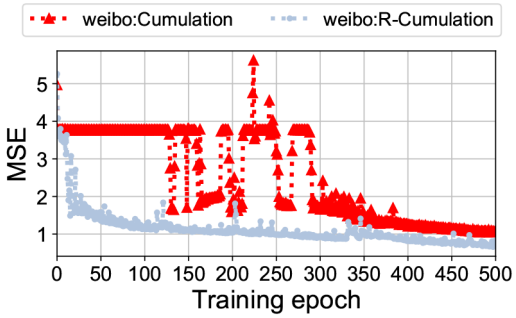


**Fig. 10** *Cumulation versus R-Cumulation*

1,412, 1,697, and 2,010, respectively. Also, the classification accuracy is 67.88, 81.59, and 96.63%, respectively. It can be seen that the results of the first classification and the second classification are not satisfactory. The third classification is essential. The second classification and the third classification increase the accuracy by 13.27% and 15.05%, respectively. The results of our classification are classified packets, and they are divided into different clusters. Through iterative classification, the number of clusters is reduced and the number of packets contained by several large clusters is increased. As shown in Fig. 8, the number of clusters after the first, second, and third classification decreases as iterations. It shows that the second classification and the third classification succeed in merging clusters of the same website. Although 3.3% of the data packets are not classified into six clusters, they will not affect the subsequent prediction of web-browsing QoE due to their limited impact on CTV.

*6.2.2 QoE prediction: Evaluation metrics.* In statistical analysis of regression problem, mean square error (MSE), $R^2$ score, adjusted $R^2$ score and explained variance are the commonly used metrics for performance evaluation. In the formula below, $y$ is the actual value, $f$ is the predicted value, and $n$ is the number of samples.

In statistics, the MSE of the estimator measures the mean square of the error, i.e. the mean square deviation between the estimated value and the actual value. The MSE is a measure of estimator quality. It is always non-negative and the closer it gets to zero, the better.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_i)^2$$

R-squared ($R^2$ score) is a statistical measure, representing the proportion of the variance for a dependent variable explained by an independent variable or variables in a regression model. Correlation explains the strength of the relationship between an independent and dependent variable, while R-squared explains to what extent the variance of one variable explains the variance of the second variable. The closer $R^2$ is to 1, the more accurate the prediction of the model is.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

The adjusted $R^2$ score is a modified version of $R^2$ score that has been adjusted according to the number of predictors in the model. The adjusted $R^2$ score increases only if the new term improves the model more than expected. It decreases when a predictor improves the model less than expected. It should be specially stated that $p$ refers to the dimension of the input variable.

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n-1)}{n - p - 1}$$

Explained variance is used to measure the discrepancy between a model and actual data [44]. The best possible score is 1.0.

$$\text{Explained variance} = 1 - \frac{\text{Var}(y - f)}{\text{Var}(y)}$$

*Data distribution.* We got lots of labelled data with different ATFs by Dummynet. The performance of the supervised learning model is sightly affected by the distribution of data. For example, a similar amount of data for each category is generally required in classification problems. Our problem is regression, which usually requires a reasonable distribution of data [45]. Fig. 9 shows the ATF distribution of weibo. It can be seen that most ATFs are between 0 and 13 s. By controlling the network condition, we got a great amount of data in 0–2 s, an extremely common interval. When the network condition fluctuates, ATF is large, which usually indicates poor user experience. As shown in the figure, the ATFs basically cover the situations encountered by users in normal use, to meet the needs of the subsequent regression model.

*Cumulation versus R-Cumulation.* We will now demonstrate that the improved Cumulation model, R-Cumulation, has better performance. Fig. 10 shows the prediction MSE by Cumulation and R-Cumulation against the number of training epochs for weibo. It suggests that the model converges much faster when R-Cumulation is at about epoch 150. With Cumulation, the model converges at about epoch 350. In addition, the MSE of the R-Cumulation model is also lower than that of the Cumulation model. The minimum MSE of the Cumulation and R-Cumulation models is 1.0554 and 0.6587, respectively. Based on the results, we believe that the training accuracy and convergence speed of the Cumulation model can be significantly improved by reversing input variables.

*Performance.* Table 2 shows the performance of models on the $R^2$ score, explained variance, and adjusted $R^2$ score. The data volume of weibo and tmall test dataset is about 5400, 3900 samples (30% of the dataset), respectively. The results show that for weibo, the three models work remarkably well on ATF prediction. We can conclude that the performance of predicting ATF for weibo is, R-Cumulation > Cumulation > Slope, despite their close metrics. The metrics are close to 1, suggesting the performance of predicting ATF is close to optimal. For tmall, the performance is less impressive and there are great differences among the three models. Among the three models, R-Cumulation appears to be the best performer on tmall. The metrics are close to 0.84. The Slope model's metrics are all below 0.4, indicating its poor prediction performance. Cumulation is slightly better than Slope, but it is also

**Table 2** ATF prediction performance comparison

| | Weibo | | | tmall | | |
|---|---|---|---|---|---|---|
| | Slope | Cumulation | R-Cumulation | Slope | Cumulation | R-Cumulation |
| $R^2$ *score* | 0.8839 | 0.8991 | 0.9019 | 0.3951 | 0.5537 | 0.8319 |
| *explained variance* | 0.885 | 0.9 | 0.9 | 0.42 | 0.54 | 0.835 |
| *adjusted* $R^2$ *score* | 0.8837 | 0.8972 | 0.9002 | 0.3936 | 0.5376 | 0.8275 |

**Table 3** Training and test time comparison

| | Train | Test |
|---|---|---|
| slope | 15.2912 | 0.0975 |
| *R*-Cumulation | 1462.47 | 8.859 |

a poor predictor of ATF, the metrics of which are around 0.55. Since tmall is a shopping site, there are a lot of dynamically loaded contents on its home page. Therefore, the diversity of content (or the dataset) on its home page is a challenge to classification regardless of network conditions, which may decrease the ATF prediction accuracy. As shown in Table 2, the metrics of the R-Cumulation model are much higher than those of the Cumulation model. The performance of the Cumulation model is improved by reversing input variables. Therefore, R-Cumulation will be used instead of Cumulation in later experiments.

*Training and test time.* Table 3 shows the training and test time of the three models on weibo. The data volume of the training and test datasets is about 12,600 and 5400 samples, respectively. It shows that the training time of R-Cumulation is much longer than that of the Slope. R-Cumulation uses LSTM neural networks as the predictor. Also, the training time of LSTM neural networks depends on the number of iterations, i.e. 100. Consequently, per training batch of Cumulation requires 100 backpropagation, resulting in excessive training time. In contrast, Slope uses fully connected neural networks as the predictor, which requires only one backpropagation per training batch. As a result, the training time of R-Cumulation is much longer than that of Slope. So is the test time. The time it takes R-Cumulation to complete 5400 predictions is much longer than it takes Slope. To generate output, R-Cumulation requires 100 forward propagation, while Slope only requires once. Furthermore, the time of predicting 5400 flows by the two models is only about 0.0975 s and 8.859 s, respectively. It suggests that QoE of a flow can be predicted in only 0.00018 s, thus enabling the real-time web-browsing QoE prediction of the model. Additionally, there are lots of mice flows and fewer elephant flows in the network. Mice flows can be filtered out by traffic classification algorithms or other methods [26, 46], and then only elephant flows will be dealt with. Moreover, the better the machine, the faster the prediction.

*Summary.* We have proposed three models based on our architecture for ATF prediction. Based on the above results, we have come to the conclusion that the R-Cumulation model has the best performance, but the longest training and prediction time. Regardless of websites, the R-Cumulation model is a good predictor of the web-loading ATF time. Cumulation has a reasonable performance with the longest training and prediction time. This indicates that reversing the input variable is indeed very effective. The Slope model has short training and prediction time, but the worst performance. In addition, it took us three months to collect data, which included a variety of web contents. Nonetheless, R-Cumulation is still a good predictor of ATF, demonstrating that our model can be updated by re-training with new data. This effect cannot be achieved by setting thresholds or deducing mathematical formulas. Furthermore, the well-trained model takes up extremely little storage space. Specifically, Slope requires 18 kB of storage space, while R-Cumulation and Cumulation require 1065 kB due to more parameters.

## 7 Conclusion

In this study, to predict web-browsing QoE, we present our architecture, which relies on packet-level measurements without deeply parsing the packet payload and can be deployed on the network intermediate devices. After classifying packets belonging to different websites and modelling each website separately, our architecture can evaluate user perceived experience by predicting the exact ATF value of page loading through a regression model. Furthermore, we demonstrate the potential and feasibility of predicting user experience of surfing the website through ML methods.

The proposed web-browsing QoE architecture does not take into account that users may have different tolerance for web page loading speed. This is a limitation of the current solution as the level of tolerance has an impact on the web-browsing experience. A possible extension to the approach is to predict the user experience directly. The user's real experience can be obtained through crowdsourcing tests, and then the direct prediction of user experience can be made through the model.

## 8 Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful suggestions. This work was supported by the National Key Research and Development Program of China (no. 2019YFB1802600), the National Natural Science Foundation of China (NSFC) (no. 61702049), the Research and Development Program in Key Areas of Guangdong Province (no. 2018B010113001), and the Fundamental Research Funds for the Central Universities.

## 9 References

[1] Maier, G., Feldmann, A., Paxson, V., *et al.*: 'On dominant characteristics of residential broadband internet traffic'. Proc. 9th ACM SIGCOMM Conf. on Internet Measurement, Chicago, IL, USA., 2009, pp. 90–102

[2] Schatz, R., Egger, S.: 'An annotated dataset for web browsing QoE'. 2014 Sixth Int. Workshop on Quality of Multimedia Experience (QoMEX), Singapore, Singapore, 2014, pp. 61–62

[3] 'Akamai research'. Accessed 2019. https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp/

[4] 'How one second could cost amazon $1.6 billion in sales'. Accessed 2019. https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales/

[5] Cardoso, J., Sheth, A., Miller, J., *et al.*: 'Quality of service for workflows and web service processes', *Web Semantics Sci. Serv. Agents World Wide Web*, 2004, **1**, (3), pp. 281–308

[6] Pan, T., Huang, T., Mao, J., *et al.*: 'Opensched: programmable packet queuing and scheduling for centralized QoS control'. 2017 ACM/IEEE Symp. on Architectures for Networking and Communications Systems (ANCS), Beijing, China, 2017, pp. 95–96

[7] Pan, T., Song, E., Bian, Z., *et al.*: 'Int-path: towards optimal path planning for in-band network-wide telemetry'. IEEE INFOCOM 2019-IEEE Conf. on Computer Communications, Paris, France, 2019, pp. 487–495

[8] Wang, S., Zhang, J., Huang, T., *et al.*: 'Fdalb: flow distribution aware load balancing for datacenter networks'. 2016 IEEE/ACM 24th Int. Symp. on Quality of Service (IWQoS), Beijing, China, 2016, pp. 1–2

[9] Lycett, M., Radwan, O.: 'Developing a quality of experience (QoE) model for web applications', *Inf. Syst. J.*, 2019, **29**, (1), pp. 175–199

[10] Hoßfeld, T., Skorin-Kapov, L., Heegaard, P.E., *et al.*: 'Definition of QoE fairness in shared systems', *IEEE Commun. Lett.*, 2016, **21**, (1), pp. 184–187

[11] Jahromi, H.Z., Delaney, D.T., Rooney, B., *et al.*: 'Establishing waiting time thresholds in interactive web mapping applications for network QoE management'. 2019 30th Irish Signals and Systems Conf. (ISSC), Maynooth, Ireland, 2019, pp. 1–7

[12] Pedras, V., Sousa, M., Vieira, P., *et al.*: 'A no-reference user centric QoE model for voice and web browsing based on 3G/4G radio measurements'. 2018 IEEE Wireless Communications and Networking Conf. (WCNC), Marrakesh, Morocco, 2018, pp. 1–6

[13] Saverimoutou, A., Mathieu, B., Vaton, S.: 'A 6-month analysis of factors impacting web browsing quality for QoE prediction', *Comput. Netw.*, 2019, **164**, p. 106905

[14] Ben-Letaifa, A.: 'Wbqoems: web browsing QoE monitoring system based on prediction algorithms', *Int. J. Commun. Syst.*, 2019, **32**, (13), p. e4007

[15] Skorin-Kapov, L., Varela, M., Hoßfeld, T., *et al.*: 'A survey of emerging concepts and challenges for QoE management of multimedia services', *ACM Trans. Multimedia Comput. Commun. Appl.*, 2018, **14**, (2s), pp. 1–29

[16] Casas, P., Seufert, M., Schatz, R.: 'Youqmon:a system for on-line monitoring of youtube QoE in operational 3 g networks', *ACM Sigmetrics Perform. Eval. Rev.*, 2013, **41**, (2), pp. 44–46

[17] Saverimoutou, A., Mathieu, B., Vaton, S.: 'Web browsing measurements: an above-the-fold browser-based technique'. IEEE Int. Conf. on Distributed Computing Systems, Vienna, Austria, 2018

[18] Hoβfeld, T., Metzger, F., Rossi, D.: 'Speed index: relating the industrial standard for user perceived web performance to web QoE'. 2018 Tenth Int. Conf. on Quality of Multimedia Experience (QoMEX), Cagliari, Italy, 2018, pp. 1–6

[19] Brutlag, J., Abrams, Z., Meenan, P.: 'Above the fold time: measuring web page performance visually'. Velocity: Web Performance and Operations Conf., Santa Clara, CA, USA., 2011

[20] Varvello, M., Blackburn, J., Naylor, D., *et al.*: 'EYEORG: a platform for crowdsourcing web quality of experience measurements'. Proc. 12th Int. on Conf. on emerging Networking Experiments and Technologies, Irvine, CA, USA., 2016, pp. 399–412

[21] Da-Hora, D.N., Asrese, A.S., Christophides, V., *et al.*: 'Narrowing the gap between QoS metrics and web QoE using above-the-fold metrics'. Int. Conf. on Passive and Active Network Measurement, Berlin, Germany, 2018, pp. 31–43

[22] Gao, Q., Dey, P., Ahammad, P., *et al.*: 'Perceived performance of top retail webpages in the wild', *ACM SIGCOMM Comput. Commun. Rev.*, 2017, **47**, (5), pp. 42–47

[23] Chhabra, A., Kiran, M.: 'Classifying elephant and mice flows in high-speed scientific networks'. Proc. INDIS 2017, Denver, CO, USA., 2017

[24] Gupta, P., McKeown, N.: 'Algorithms for packet classification', *IEEE Netw.*, 2001, **15**, (2), pp. 24–32

[25] Pan, T., Guo, X., Zhang, C., *et al.*: 'Tracking millions of flows in high speed networks for application identification'. 2012 Proc. IEEE INFOCOM, Orlando, FL, USA., 2012, pp. 1647–1655

[26] Mori, T., Uchida, M., Kawahara, R., *et al.*: 'Identifying elephant flows through periodically sampled packets'. Proc. 4th ACM SIGCOMM Conf. on Internet measurement, Sicily, Italy, 2004, pp. 115–120

[27] Schatz, R., HoBfeld, T., Casas, P.: 'Passive YouTube QoE monitoring for ISPs'. Sixth Int. Conf. on Innovative Mobile & Internet Services in Ubiquitous Computing, Palermo, Italy, 2012

[28] Staehle, B., Hirth, M., Pries, R., *et al.*: 'Yomo: a YouTube application comfort monitoring tool'. New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications, Tampere, Finland, 2010, vol. 6

[29] Bagga, P., Joshi, A., Hans, R.: 'QoS based web service selection and multi-criteria decision making methods', *Int. J. Inter. Multimedia Artif. Intell.*, 2019, **5**, (4), pp. 113–121

[30] HoBfeld, T., Schatz, R., Biedermann, S., *et al.*: 'The memory effect and its implications on web QoE modeling'. Int. Teletraffic Congress, San Francisco, CA, USA., 2011

[31] Casas, P., Mazel, J., Owezarski, P.: 'Unsupervised network intrusion detection systems: detecting the unknown without knowledge', *Comput. Commun.*, 2012, **35**, (7), pp. 772–783

[32] Mushtaq, M.S., Augustin, B., Mellouk, A.: 'Empirical study based on machine learning approach to assess the QoS/QoE correlation'. 2012 17th European Conf. on Networks and Optical Communications, Vilanova i la Geltru, Spain, 2012, pp. 1–7

[33] Luo, X., Liu, J., Zhang, D., *et al.*: 'A large-scale web QoS prediction scheme for the industrial internet of things based on a kernel machine learning algorithm', *Comput. Netw.*, 2016, **101**, pp. 81–89

[34] Wang, Y., Li, P., Jiao, L., *et al.*: 'A data-driven architecture for personalized QoE management in 5G wireless networks', *IEEE Wirel. Commun.*, 2017, **24**, (1), pp. 102–110

[35] Zhang, P., Jin, H., Dong, H., *et al.*: 'LA-LMRBF: online and long-term web service QoS forecasting', *IEEE Trans. Serv. Comput.*, 2019 (Early Access)

[36] Liu, A., Shen, X., Li, Z., *et al.*: 'Differential private collaborative web services qos prediction', *World Wide Web*, 2019, **22**, (6), pp. 2697–2720

[37] Sackl, A., Casas, P., Schatz, R., *et al.*: 'Quantifying the impact of network bandwidth fluctuations and outages on web QoE'. 2015 Seventh Int. Workshop on Quality of Multimedia Experience (QoMEX), Messinia, Greece, 2015, pp. 1–6

[38] Song, E., Pan, T., Jia, C., *et al.*: 'WebQMon.ai: threshold-oblivious on-line web QoE assessment using LSTM neural networks'. IEEE INFOCOM 2019-IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 1033–1034

[39] Basseville, M.: 'Divergence measures for statistical data processing–an annotated bibliography', *Signal Process.*, 2013, **93**, (4), pp. 621–633

[40] Turner, C.R., Wolf, A.L., Fuggetta, A., *et al.*: 'Feature engineering'. Proc. 9th Int. Workshop on Software Specification and Design, Washington, DC, USA., 1998, p. 162

[41] Hecht-Nielsen, R.: 'Theory of the backpropagation neural network'. Neural networks for perception, 1992, pp. 65–93

[42] 'Alexa top 500 global sites'. Accessed 2019. https://www.alexa.com/topsites

[43] Bei, X., Chen, N., Zhang, S.: 'On the complexity of trial and error'. Proc. Forty-Fifth Annual ACM Symposium on Theory of Computing, Palo Alto, CA, USA., 2013, pp. 31–40

[44] Rosenthal, G., Rosenthal, J.A.: '*Statistics and data interpretation for social work*' (Springer Publishing Company, USA., 2011)

[45] Chen, Y., Caramanis, C.: 'Noisy and missing data regression: distribution-oblivious support recovery'. Int. Conf. on Machine Learning, Atlanta, GA, USA., 2013, pp. 383–391

[46] Curtis, A.R., Mogul, J.C., Tourrilhes, J., *et al.*: 'Devoflow: scaling flow management for high-performance networks'. ACM SIGCOMM Computer Communication Review, 2011, vol. 41, pp. 254–265