DeepQoE: Real-time Measurement of Video QoE from Encrypted Traffic with Deep Learning

Meng Shen*¶, Jinpeng Zhang*, Ke Xu†¶,, Liehuang Zhu*, Jiangchuan Liu‡, Xiaojiang Du§
*School of Computer Science, Beijing Institute of Technology, Beijing, P. R. China
†Tsinghua University & BNRist, Beijing, P. R. China
‡Simon Fraser University, British Columbia, Canada
§Department of Computer and Information, Temple University, USA
¶Peng Cheng Laboratory, Shenzhen, P. R. China
{shenmeng, zhangjinpeng, liehuangz}@bit.edu.cn; xuke@tsinghua.edu.cn; jcliu@cs.sfu.ca; dxj@ieee.org

***Abstract*—With the dramatic increase of video traffic on the Internet, video quality of experience (QoE) measurement becomes even more important, which provides network operators with an insight into the quality of their video delivery services. The widespread adoption of end-to-end encryption protocols such as SSL/TLS, however, sets a barrier to QoE monitoring as the most valuable indicators in cleartext traffic are no longer available after encryption. Existing studies on video QoE measurement in encrypted traffic support only coarse-grained QoE metrics or suffer from low accuracy.**

In this paper, we propose DeepQoE, a new approach that enables real-time video QoE measurement from encrypted traffic. We summarize critical fine-grained QoE metrics, including startup delay, rebuffering, and video resolutions. In order to achieve accurate and real-time inference of these metrics, we build DeepQoE by employing Convolutional Neural Networks (CNNs) with a sophisticated input and architecture design. More specifically, DeepQoE only leverages packet Round-Trip Time (RTT) in upstream traffic as its input. Evaluation results with real-world datasets collected from two popular content providers (i.e., YouTube and Bilibili) show that DeepQoE can improve QoE measurement accuracy by up to 22% over the state-of-the-art methods.

***Index Terms*—Encrypted traffic analysis, video QoE, network measurement, deep learning, convolutional neural networks**

I. Introduction

Recent years have witnessed the prosperity of video-based services and applications, which contribute greatly to the dramatic increase in network traffic. The global internet phenomena report published by Sandvine [1] shows that video content accounts for an impressive 58% of the Internet traffic. Monitoring video quality of experience (QoE) provides network operators with a deep insight into the quality of their network services on video delivery. In particular, real-time tracking of video QoE enables network operators to dynamically optimize their network bandwidth provisioning and traffic routing strategies.

For cleartext video traffic, it is easy to obtain common QoE metrics (e.g., startup delay and stalls) through deep packet inspection. However, an increasing number of network services have adopted end-to-end encryption protocols for consideration of user privacy. Currently, more than 70% of Internet traffic has been encrypted by SSL/TLS [1, 2]. YouTube, Netflix and Bilibili, the popular video providers, have encrypted a large portion of their video streaming traffic. This poses a great challenge to real-time measurement of video QoE as most valuable indicators are no longer available in encrypted traffic.

Existing studies mostly rely on machine learning models such as Naive Bayes and random forests [3–5] to extract features from encrypted video traffic for inferring critical QoE metrics. Features that are commonly used range from packet level to flow level, such as packet counts, TCP flag counts, and statistical information. It usually takes a relatively long time to extract and calculate the desired features from encrypted traffic, which cannot meet the requirement of real-time monitoring. In addition, they only enable coarse-grained video QoE measurements. Take for example the video resolution monitoring, existing studies usually classify all types of resolutions into two or three classes, which are labeled as high, medium, and low qualities. Thus, the problem of measuring video resolutions are turned into a bi-classification or three-classification problem. This measurement granularity is inadequate for a comprehensive understanding of video delivery quality, as current adaptive streaming usually supports a group of video qualities, varying from 144p to 1080p.

The goal of this paper is to develop an accurate method for real-time measurement of fine-grained video QoE metrics from encrypted traffic. Motivated by the literature in QoE assessment [6], we select three types of critical metrics, namely startup delay, rebuffering, and video resolution. Although these metrics can be easily obtained from cleartext traffic, it remains a challenging task to accurately monitor them in encrypted scenarios. Another challenge lies in the time efficiency of video QoE measurement approaches. Utilizing the information of packets can be realized by the network devices such as routers in a time window of 5 or 10 seconds. Therefore, QoE measurement should be conducted in a real-time manner with segmented streaming information.

Corresponding author: Liehuang Zhu (email: liehuangz@bit.edu.cn).

978-1-7281-6887-6/20/$31.00

In this paper, we propose DeepQoE, a new method for real-time video QoE measurement on encrypted traffic. To achieve the goal that infers video QoE metrics in a real-time manner, we use only the Round-Trip Time (RTT) of upstream packets as a crucial feature. This greatly reduces the computational complexity in feature extraction and is more suitable for real-time video QoE measurement. To enable accurate and fine-grained monitoring, DeepQoE leverages Convolutional Neural Networks (CNNs) with sophisticated designs, including the architecture, activation function, and loss function. Compared with the traditional machine learning methods used in existing studies (cf. Table I), CNNs solve the problem of manually selecting and fine-tuning features.

The main contributions of this paper are summarized as follows:

We observe that the RTT information of upstream packets can act as a crucial feature for inferring fine-grained video QoE metrics. We identify fine-grained measurement metrics, which enable network operators to identify specific startup delay and concrete video resolutions. Comparing with existing video QoE inferring models, our method does not need to calculate traffic statistics, such as mean, maximum, minimum, etc. It takes a quite short time (i.e., around 4 milliseconds) to extract features from packets, which greatly reduces the calculation cost of the front-end server and is more suitable for real-time inferring.

We design a deep learning-based model for accurate QoE measurement. In order to apply CNNs in our scenario, we design an appropriate neural network structure, an activation function, and a loss function. The neural networks generate features automatically by convolution functions from the input data. CNNs extract high latitude features to achieve the goal that infers fine-grained video QoE metrics.

We demonstrate the effectiveness of DeepQoE with real-world datasets collected from two popular content providers. We select YouTube [7], the largest content provider around the world, and Bilibili [8] as the experiment data sources. More than 10 thousands video sessions are collected in total, and the pcap file size reaches 480GB in our experiments. The evaluation results show that compared with the state-of-the-art methods, DeepQoE can improve the measurement accuracy by up to 22%.

To the best of our knowledge, this is the first study that applies CNNs to infer video QoE metrics from encrypted traffic. The rest of this paper is organized as follows. We introduce the background and related work in Section II and describe the motivation of employing RTT information as model input in Section III. Then, we present the design details of DeepQoE in Section IV and evaluate its performance in Section V. We discuss several future research issues in Section VI and conclude the paper in Section VII.

II. Background and Related Work

In this section, we briefly introduce the adaptive video streaming mechanism that is increasingly adopted by video websites. Then, we elaborate on the QoE metrics considered in this paper and describe the application scenario. Finally, we summarize the existing studies on measuring video QoE metrics from encrypted traffic.

A. Adaptive Video Streaming

HTTP Adaptive Streaming (HAS) combines the traditional streaming media technology with the characteristics of HTTP progressive download and play, which transmits media content to users in HTTP mode. The video content is broken into a sequence of HTTP-based file segments by HAS, each of which hold a short interval of playback time of an original video that is with a duration of several hours, such as a documentary or a live concert. A video player requests a sequence of fixed video segments from the server and stores the downloaded video segments in its local buffer so that it can continue to play smoothly while waiting for the arrival of the follow-up video segments. Multiple quality levels, usually varying from 144p to 1080p, are prepared for each video segment. The quality of each video segment ranges from 144p to 1080p and the video player request the video segment with a specific resolution. The video resolution adaptive logic of the video player determines the video resolution of the request according to the network condition [15, 16].

B. QoE Metrics

QoE acts as an important measurement of the overall performance of video streaming services perceived by end users. Traditionally, subjective evaluation solicits users' evaluation scores to measure the level of QoE. However, it takes a lot of time and requires a large amount of human resources. Objective quality models are developed to provide faster evaluation results.

In order to develop efficient video QoE prediction models that are independent of specific video contents, network statistics (e.g. packet delay) and spatiotemporal features of the distorted video are involved. The video QoE are characterized most commonly by video quality metrics containing startup delay, rebuffering events, and video resolution (i.e., encoding bitrate) [17].

Motivated by existing studies, we also employ these QoE metrics as the criteria to evaluate video QoE. In addition, we also pay special attention to fine-grained metrics that enable a much better understanding of video delivery quality.

Startup delay. It measures the time spent by the video player when it requests the initial video segments until meeting its requirements. This delay consists of the network delay and the initial buffering delay. Network delay is influenced by DNS lookups, network latency, longer server response times and CND redirection. The initial buffering delay indicates the time required to perform the initial filling of the buffer, which has sufficient local video data to allow layer for smooth playback.

Startup delay plays an important role in the data-driven QoE evaluation [6]. As the startup delay increases, video popularity often drops significantly. The coarse-grained metric of startup delay is commonly used in existing studies, where a binary

TABLE I: Summary of existing studies on QoE metric measurement

Methods	Online	Classifiers	QoE Metrics		
			Startup delay	Rebuffering	Video resolution
Dimopoulos et al. [9]		Random Forest	$\surd$	$\surd$	$\surd$
Hammad et al. [3]	$\surd$	Decision Tree	$\surd$	$\surd$	$\surd$
Orsolic et al. [5]		Naive Bayes, Decision Tree and Random Forest	$\surd$	$\surd$	$\surd$
Mangla et al. [10, 11]		Session Modeling-based Approach	$\surd^*$	$\surd^*$	$\surd^*$
Krishnamoorthi et al. [12]	$\surd$	Decision Tree and SVM		$\surd^*$	
Gutterman et al. [13]	$\surd$	Random Forest		$\surd^*$	$\surd^*$
Schmitt et al. [14]	$\surd$	Regression Model and Random Forest	$\surd^*$		$\surd^*$
DeepQoE	$\surd$	CNNs	$\surd^*$	$\surd^*$	$\surd^*$

$\surd^*$ represents fine-grained QoE metrics.

parameter is learned to indicate whether the startup delay is below a certain threshold (e.g., 5 seconds) or not.

Fine-grained startup delay. This metric has two representations that one directly measures the video's initial buffering time to seconds. Although the precision of it is high, the prediction accuracy is relatively low. Mangla et al. [11] have only got about 70% of the prediction accuracy. The other metric is proposed in this paper, which measures whether the video has started to play within a specific second.

Rebuffering events. When the content locally stored in the buffer of the video player cannot meet the playback requirement, rebuffering will appear until the video segments are buffered enough to satisfy the playing requirements. More specifically, playback is forcibly suspended until more video segments are downloaded and the buffer is refilled.

From the perspective of user experience, rebuffering is also known as *stalling*. It has been proved that frequent rebuffering events have a high correlation with poor QoE. For instance, a video with 2 rebuffering events in a 3-second session will result in a significantly lower Mean Opinion Score (MOS) [18]. The coarse-grained rebuffering metric is usually abstracted as a binary parameter, indicating whether rebuffering events happen in each video session or not.

Fine-grained rebuffering metric. As indicated by existing studies, the frequency of rebuffering events in video playback can severely affect the user viewing experience. Thus, we refer a fine-grained rebuffering metric to the specific number of rebuffering events during each time window. In this paper, we adopt a 10-second time window as the commodity routers can usually buffer network traffic for 10 seconds.

Video resolution. It is a parameter used to measure the amount of data in one frame. In general, the higher the resolution of a video, the better the users viewing experience. Existing researches on subjective evaluations demonstrated the relationship between video resolution and QoE [19]. They take experiments in mobile networks, and the results show that video streams with higher resolution are linked to better overall QoE.

In a real network environment, content providers dynamically adapt video resolution to network conditions. The typical inference target of video resolution is the average resolution during a period of time, such as 10 seconds. Coarse-grained resolution metric has been employed in previous studies, where identifying video resolution is abstracted as a binary (i.e., low or high) or three-category (i.e., low, medium, or high) classification problem.

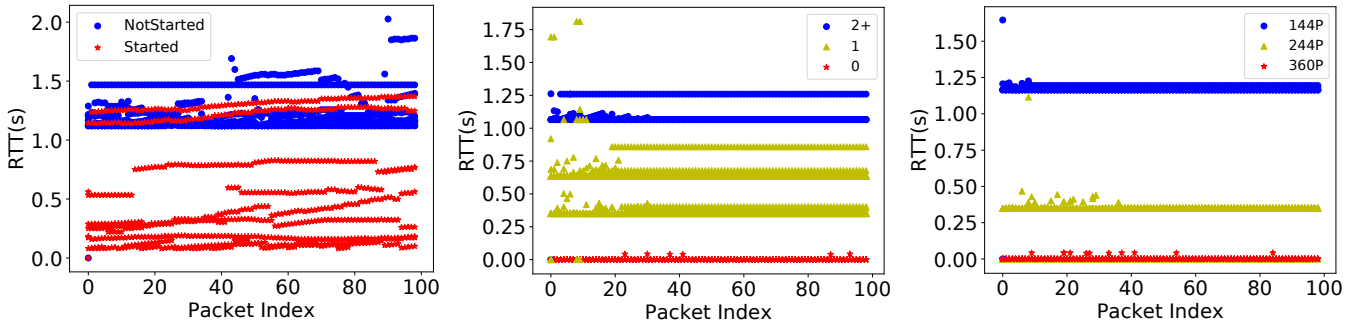
Fine-grained video resolution. It aims to identify the specific resolution of each video session. Fine-grained video resolution is more desirable as it provides a deep insight into the concrete display quality on the user side and also enables the understanding of resolution changes (e.g., active resolution adjustment by users). As a result, it turns to a multi-category classification problem, where each category corresponds to a specific resolution (e.g., 144p).

C. Related Work

Table I shows cutting-edge researches on QoE metrics for encrypted video traffic. The QoE metrics inferring methods mainly contain Machine Learning (ML) based methods and Session Modeling (SM) based methods.

ML-based approaches. This approach predicts video QoE by associating the network observable metrics like packet length, delay, and loss with the video QoE metrics using traditional machine learning algorithms[20]. Dimopoulos et al. [9] extracted features from encrypted flows, and then they used Random Forest to identify the initial time delay, stalls, average representation quality and representation quality variation about the video QoE. Their method needs to calculate features from the whole video session, which is not for online video QoE measurement. Hammad et al. [3] proposed a machine learning-based solution to predict QoE metrics using J48 based on statistics of network flow. Features they extracted from packets and a time window, which contained statistics such as mean, minimum, maximum, median and etc. The machine they used is J48, a traditional decision tree classification algorithm. Their method can't infer video QoE in a fine-grained, which is not significant for network operators to make network operator strategy. Orsolic et al. [5] proposed a machine learning approach to infer YouTube QoE metrics based on encrypted traffic analysis. Their method needs to calculate the number of statistic features, which is not suitable for real-time video QoE identification.

SM-based approaches. QoE metrics are inferred by modeling the video session according to the properties of the



(a) Startup delay within 3 seconds (b) Number of rebuffering events (c) Video resolution

Fig. 1: The correlation between RTT and video QoE metrics on YouTube

streaming protocol used by the video. In prior work, Mangla et al. [10] proposed a QoE inference approach for the HTTP-based Adaptive Streaming (HAS) video, called MIMIC. The basic idea was to calculate the average bitrate and MIMIC relies on application layer information. Mangla et al. [11] proposed a QoE inferring method for encrypted video, named eMIMIC. Their method uses packet headers of network traffic to represent HAS sessions and estimates the QoE metrics. However, their method is based on video session modeling, which means that model training can only be performed offline, and real-time QoE recognition cannot be achieved.

Different from the existing studies discussed above, in this paper, we apply CNNs to build the classifier for inferring fine-grained QoE metrics, which can automatically and adaptively learn valuable features and thus avoids empirical and hand-crafted feature selection process.

III. Motivation

In this section, we describe the observations that motivate us to leverage RTT information to infer fine-grained QoE metrics.

In web applications, RTT measures the propagation delay of the path between a client and the corresponding server. It usually refers to the duration of sending a packet by the client to receiving its acknowledgment. In this paper, we regard the value extracted by `tcp.analysis.ack_rtt` in Tshark as the packet-level RTT information, which is a time offset on the client-side between sending a packet and the arrival of its acknowledge (ACK) packet.

High RTT in a video session can have a negative effect on video delivery quality, thus impairing user experiences. To verify the impact of RTT on QoE metrics listed in Section II, we conduct experiments to explore the relationship between RTT and QoE metrics, as illustrated in Figure 1. For ease of exhibition, we select 10 YouTube traffic flows and visualize the RTT distribution of the first 100 packets in each flow.

For startup delay shown in Figure 1(a), we simply set a 3-second threshold and present the RTT of packet sequences that started (i.e., red stars) or not started (i.e., blue dots) within 3 seconds. It shows clearly that RTTs of the majority of *NotStarted* and *Started* flows spread above or below the cutting line of 1 second, respectively. For rebuffering and video resolution in Figures 1(b) and 1(c), we observe similar or even more clear spread of RTTs for fine-grained metrics.

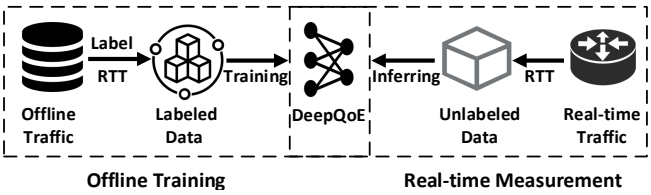


Fig. 2: System overview

Packet RTT can reflect the state of network condition. The smaller the RTT value, the better the network environment. Operators can infer the required QoE metrics from network conditions which are reflected by RTTs. Note that HTTP adaptive video streaming always has a segment organization due to the adaptive characteristics. Though the packet delay can be bursty due to segment structure, packet-level RTT in each segment remains relatively stable.

IV. The Proposed DeepQoE

In this section, we present the design of DeepQoE. An overview of the QoE metrics measurement process with DeepQoE is illustrated in Figure 2, which mainly consists of two steps: *offline training* and *real-time QoE measurement*.

In the training process, we collect encrypted traffic of the target video provider and extract the input information (i.e., packet-level RTTs and labels) to train a CNN-based classifier. We leverage CNNs to construct DeepQoE as it does not require manually selecting and fine-tuning features. The data labeling strategy will be described in Section V. In the online QoE measurement process, real-time RTTs are extracted from encrypted traffic of the videos we want to appraise and then fed to DeepQoE for quick identification.

A. Model Input

In order to satisfy the real-time measurement requirements, the classifier needs to extract as few input information as possible from the original encrypted traffic [21]. We take advantage of the RTT information of the upstream packets. Now, we provide a formal description of the model input.

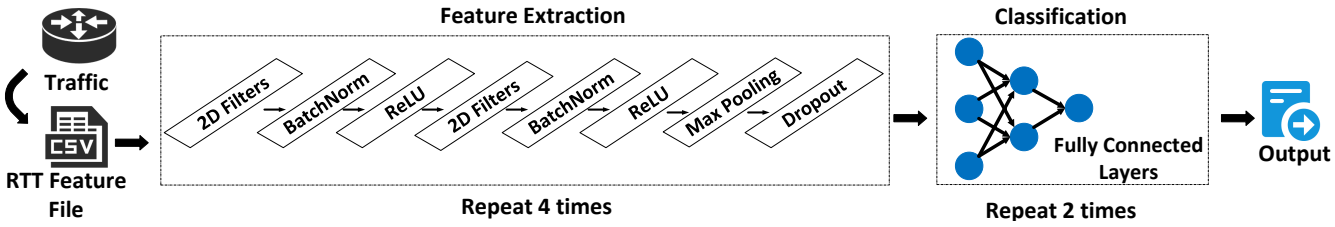


Fig. 3: Structure of the Convolutional Neural Network in DeepQoE

TABLE II: List of notations

Notation	Meaning
t	The video has been played in t second
N_s	The number of RTTs used in the startup delay estimation
T	Time window
T_f	The first half of the time window
T_s	The last half of the time window
N_r	The number of RTTs used in the rebuffering event or video resolution estimation
RIN	The model input
Θ	The set of CNNs parameters
$\mathcal{L}_i$	The fully-connected layers
C	The number of QoE metrics
$\mathcal{L}_a$	Cross entropy function
$\mathcal{L}_b$	Confidence control constraint
$\mathcal{L}$	Loss function

For the *startup delay* measurement, we use t to represent the video that has been played in t seconds. If we want to infer whether the video has been played in t seconds, we only need to extract $t*N_s$ RTTs in the first t seconds. In our experiments, N_s equals 50, as we get an average of 50 RTTs per second in our experimental environment. If the number or RTTs in the packet sequence exceeds $t * N_s$, we only use the first $t * N_s$ information. If the vector has fewer than $t * N_s$, we pad it to $t * N_s$ by appending zeros. The notations used in this paper are summarized in Table II.

For measuring *rebuffering* and *video resolution*, we adopt a T-second interval as the identification time window. T is set as 10 in this paper, as the network operator is capable of utilizing packet data in a moving time window of 10 seconds for each network flow. We divide T into two parts on average, which are defined as the first and last half of each time window. Model inputs are derived separately for the first and last half of each time window. No matter for the first T_f time window or the last T_s time window, we only need to extract the first N_r RTTs in the packet sequence. If the number of RTTs in each time window is fewer than N_r, we pad it to N_r by appending zeros. N_r is set to be 150, as 150 RTTs are enough to infer the QoE metrics.

We represent the three kinds of model input by RIN, for startup delay RIN_s:

$$RIN_s = [RTT_{t*N_s}] \tag{1}$$

for rebuffering and video resolution RIN_r:

$$RIN_r = [RTT_f; RTT_s] \tag{2}$$

where RTT_f is the RTT vector of T_f, and RTT_s is the RTT vector of T_s.

B. Network Architecture

We use CNNs to construct the classifier, which can extract features automatically from the input information. The structure of a four-layer stacked CNNs used in DeepQoE is illustrated in Figure 3. The main body of the neural network consists of a convolution layer and a fully connected layer.

Convolution layer. In this layer, we repeat the convolution operation four times. We use 2D convolutional functions twice in each convolution operation. After the convolutional function, we use a BatchNorm function to makes the input of each layer of the neural network keep the same distribution during the training of the neural network. After the BatchNorm function, we use a rectified linear unit (ReLU). A max-pooling layer are added behind the ReLU function to reduce the size of representation. We add a max-pooling layer to reduce the size of representation. At the end of each Convolutional Layer, we use a dropout function to avoid over-fitting by reducing the interdependence among nodes in the neural network.

Θ represents the set of parameters in CNNs. Give the input data RTT, the feature representations can be obtained as shown in Eq. (3).

$$F = CNN\left(RIN, \Theta\right) \tag{3}$$

Fully connected layer. We use a linear function for the linear transformation of the input data calculated by the convolution layers. Then, a Batchnorm function is applied to the linear transformation data. Following the activation function V_i, there is a dropout function to avoid over-fitting. Based on the output of features extractor (i.e., F), an activation function is used before fully-connected layers to learn the representation $\mathcal{L}_i$ of RTT as shown in Eq. (4),

$$\mathcal{L}_i = Softplus\left(W_f F_i + b_f\right) \tag{4}$$

where W_f and b_f are the parameters to be learned in the activation function.

In order to infer the labels of video QoE metrics, the feature representation $\mathcal{L}_i$ are mapped into a new latent space $H_i \in \mathbb{R}^C$, where C is the number of QoE metrics we want

to identify. Furthermore, a softmax function is used to get the probability vector of QoE metrics as shown in Eq. (5).

$$\hat{y_i} = Softmax\left(H_i\right) \tag{5}$$

H_i is calculated as follows:

$$H_i = W_v \mathcal{L}_i + b_v \tag{6}$$

where W_v and b_v are parameters to be learned in the fully connected layer.

Loss function. The cross entropy function is used to calculate the loss between the identification and the ground truth, as defined in Eq. (7),

$$\mathcal{L}_a = -\frac{1}{|X|}\sum_{i=1}^{|X|}\sum_{c=1}^{C} y_{ic} log\left(\hat{y_{ic}}\right) \tag{7}$$

where $|X|$ is the number of instances.

One problem of the over-fitting is that the model is over-confident when it predicts all the probabilities on one class on the training process [22]. The model may get stuck in a local optimum when the model is overfitted on the training stage, which may reduce the classification accuracy of the model on the testing stage. To solve this problem, a confidence control constraint is proposed in this paper, which is penalized $\hat{y_{ic}}$ when it is over confident. It introduces the prior knowledge of label distribution into the training data to improve the stability of the training stage. The loss of the confidence control constrain is shown in Eq. (8).

$$\mathcal{L}_b = -\frac{1}{|X|}\sum_{i=1}^{|X|}\sum_{c=1}^{C}\left(log\left(\hat{y_{ic}}\right) + log\left(1-\hat{y_{ic}}\right)\right) \tag{8}$$

In this way, if $\hat{y_{ic}}$ reaches 0 or 1, the penalty will go to infinity. Based on Eqs. (7) and (8), we define the loss function in Eq. (9),

$$\mathcal{L} = \mathcal{L}_a + \alpha \mathcal{L}_b \tag{9}$$

where α is the weighting parameter.

Optimizer. As for the optimizer, we adopt Adam optimizer in DeepQoE. Adam algorithm is a first-order gradient optimization algorithm for the stochastic objective function, which is based on adaptive low-order moment estimation. Adam is an effective stochastic optimization method, which requires only a first-order gradient and very little memory.

C. Real-Time Measurement of QoE Metrics

When all the parameters of the CNNs classifier are learned at the training stage, the classifier is saved locally. The trained classifier can then be used to infer QoE metrics based on a real-time acquisition of RTT information from network egress nodes (e.g., the exit nodes of the campus network). More specifically, to identify the startup delay, we only need to extract $t_i * N_s$ RTTs from the packet sequence in the first t_i seconds; for rebuffering and video resolution, we separately extract the first N_r RTTs from the first T_f seconds and the last T_s seconds.

TABLE III: Datasets for Experiments

Video Source	# of Sessions	Pcap File Size
YouTube	5,833	180GB
Bilibili	6,303	302GB

V. Performance Evaluation

In this section, we are devoted to evaluating the effectiveness of DeepQoE, in terms of measurement accuracy and time overhead, by comparing it with the state-of-the-art methods.

A. Preliminary

Methods to compare. In order to fully understand the efficacy of DeepQoE, we leverage two typical methods for comparison: one uses a traditional J48 classifier to predict video QoE metrics and the other leverages deep learning for website fingerprinting, which are summarized as follows:

- J48, which uses statistical features generated from network and transport layers, such as mean, minimum, and maximum, to infer video QoE metrics [3].
- DeepFingerprinting (DF), which uses the packet direction as the input of CNNs model to obtain website fingerprints [23]. We adjust hyperparameters to achieve its best performance. DF acts as a counterpart that leverages a CNN-based classifier for encrypted traffic analysis.

Hardware. A server equipped with an Intel Core Duo i5-3230M 2.60GHz and 4GB of memory to capture video traffic in our experiment. The traffic capture script runs on Ubuntu 16.04. Video QoE metrics inferring experiments are conducted on a server, which is equipped with an Inter Core Duo 3.6GHz and 16GB of memory. The CNNs module runs on a GPU with 8GB of memory.

Cross-validation. In order to prove the effectiveness of DeepQoE, we use the collected datasets to establish a 10-fold cross-validation, i.e., 9 of the 10 portions are used as the training data and the remaining one as the testing data. The mean result is used to evaluate the performance of inferring models.

Precision, recall and F1-score are used to measure the performance of an inferring model. Recall is the number of true positive instances divided by the sum of true positive and false negative instances. Precision is the number of true positive instances divided by the true positive and false positive instances. F1-score considers both the precision and recall to measure the overall performance of a classifier.

B. Dataset Collection

In our experiments, we consider two popular network content providers, namely Youtube [7] and Bilibili [8]. Based on the global internet phenomena report published by Sandvine [1], more than 1.9 billion users log on to YouTube every month in general. Bilibili is one of the most popular video websites in China, of which user views are over 90 millions per month. Choosing different video websites can prove the universality of the proposed method.

TABLE IV: Hyperparameters selection for DeepQoE from extensive candidates search method

Hyperparameters	Search Range	Final
Optimizer	[Adam, Adamax, RMSProp,SGD]	Adamax
Learning Rate	[0.001 ... 0.01]	0.0015
Training Epochs	[10 ... 50]	30
Batchsize	[50 ... 300]	150
[Kernel, Stride, Pool] Sizes	[1 ... 16]	[3, 1, 2]
Activation Functions	[Tanh, ReLU, ELU]	ReLU
Each Convolution Layer		
Block1 [Conv1, Conv2]	[D ... 3D]	[D, 2D]
Block2[Conv3, Conv4]	[2D ... 6D]	[2D, 4D]
Block3[Conv5, Conv6]	[2D ... 10D]	[4D, 8D]
Block4[Conv7, Conv8]	[2D ... 20D]	[8D, 16D]
Pooling Layers	[Average, Max]	Max
Number of FC layers	[1 .. .4]	2
Dropout [Pooling, FC1, FC2]	[0 ... 0.3]	[0.01,0.01,0.01]

The data capturing process is shown in Figure 4. We set a list of video URLs, which contain short interviews, game videos, short documentaries and so on. The video resolutions range from 144p to 1080p. YouTube videos are automatically played by Selenium WebDriver in a Chrome browser, which runs on a server with Ubuntu 16.0.4 OS. Selenium WebDriver is controlled by a Python script, which loads the selected video URL to the browser, opens the video player button and closes the browser.

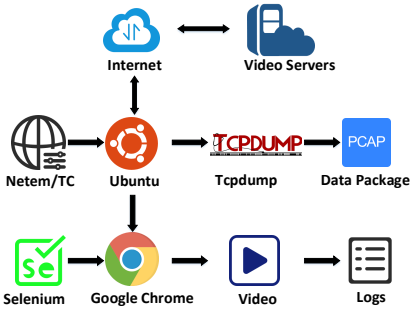


Fig. 4: Process of data collection

To get the ground truth of video QoE metrics, we use the IFrame API provided by YouTube and Bilibili to get the playing log reported by the video player. We can get the states of the player (e.g., started, not started, paused, playing, buffering) and the exact video resolution (ranging from 144p to 1080p) from the IFrame API. `Tcpdump` is used to capture traffic and extract `tcp.analysic.ack_rtt` information.

Network emulation tools containing `netem` and `tc` are used to emulate a dynamic network environment. Taking advantage of these tools in the traffic capturing server, we can emulate different network conditions by randomly setting different network parameters, such as bandwidth, delay, and packet loss rate. We divide the network condition into 10 types, which represent different network conditions. The network bandwidth ranges from 100kbps to 1000kbps, the delay ranges from 10ms to 100ms, and the packet loss ranges from 0 to 30%. Before every video playback, we randomly set network environment relying on these tools.

The video traffic of YouTube and Bilibili is collected for three months. To avoid data contingency, we collect data from 6 servers deployed in different areas. The total datasets are shown in Table III. We collect 5833 sessions of YouTube and 6303 sessions of Bilibili.

C. Hyperparameter Tuning of DeepQoE

In this subsection, we illustrate the hyperparameters used in DeepQoE.

Tuning the hyperparameters of neural networks is of great significance for achieving desirable inferring performance. Adjusting the trade-off between bias, variance and inferring accuracy is contained in this process. A large number of training data and hyperparameters make it very difficult to train DeepQoE. We build each layer of the neural network block by block by an extensive search of the hyperparameter space. In each layer of DeepQoE, we change the hyperparameters to anticipate the gradient of the parameter and decide whether its value should be increased or decreased. After completing this process, the best Top-N parameters are selected as the initial parameters. When all layers are set up, we choose the best combination of these hyperparameters. The hyperparameters search space and the final selected values are shown in Table IV. We use D to represent the dimensions of input data. After 4 blocks with 2 layers of convolutional function, we will get 16 D features extracted from the network. We use a fully connected layer to output the inferred labels.

Although the deep neural network is a powerful supervised classification model, like most machine learning models, they are also prone to over-fitting. *Dropout* [24] and *Batch Norm* [25] are applied to prevent over-fitting during the training stage of DeepQoE. Hidden units such as incoming and outgoing connections are randomly selected to remove by *Dropout* function in the training stage. *Batch Norm* normalizes the output of the fully connected layer and the convolution layer. It helps the deep learning model to accelerate learning and reduce over-fitting at the same time.

D. Evaluation of Measurement Accuracy

In this subsection, we evaluate the performance of DeepQoE from the identification accuracy of video QoE metrics.

Startup delay. To infer the startup delay time of YouTube and Bilibili, we range t_i from 1 to 8. Table V shows the inferring result of the different inferring models on YouTube. No matter which metrics, DeepQoE performs best comparing with the other two methods. The precision of DeepQoE reaches more than 90%, which is more than 10% higher than J48 and DF. At the start up delay time of 6 seconds, the recall of DeepQoE reaches 94.98%, which is more than 22% higher than J48. The recall and F1 score of DeepQoE also reach more than 90%. Table VI shows the inferring result of the different inferring models on Bilibili. DeepQoE is superior to the counterparts on the Bilibili dataset. The precision of DeepQoE reaches more than 90%, which is higher than DF

TABLE V: Startup delay inferring results with J48, DF and DeepQoE on YouTube

	J48			DF			DeepQoE		
Startup Delay Time t_i (s)	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
1	0.7732	0.7744	0.7738	0.7868	0.8165	0.8014	0.9276	0.9289	**0.9283**
2	0.6960	0.6967	0.6963	0.7766	0.7650	0.7708	0.9027	0.9026	**0.9027**
3	0.7085	0.7086	0.7085	0.7766	0.7699	0.7732	0.9376	0.9370	**0.9373**
4	0.7129	0.7135	0.7132	0.8140	0.7886	0.8011	0.9190	0.9191	**0.9190**
5	0.7082	0.7090	0.7086	0.8240	0.8029	0.8133	0.9203	0.9204	**0.9204**
6	0.7270	0.7292	0.7281	0.8248	0.7927	0.8085	0.9507	0.9498	**0.9502**
7	0.7395	0.7418	0.7406	0.8241	0.7920	0.8078	0.9158	0.9072	**0.9115**
8	0.7614	0.7631	0.7623	0.8092	0.7676	0.7879	0.9127	0.9075	**0.9101**

TABLE VI: Startup delay inferring results with J48, DF and DeepQoE on Bilibili

	J48			DF			DeepQoE		
Startup Delay Time t_i (s)	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
1	0.8484	0.8529	0.8506	0.8256	0.8360	0.8308	0.9017	0.9044	**0.9030**
2	0.8373	0.8421	0.8397	0.8014	0.8153	0.8082	0.9496	0.9491	**0.9493**
3	0.8168	0.8224	0.8196	0.8054	0.8167	0.8110	0.9420	0.9426	**0.9423**
4	0.8176	0.8222	0.8199	0.8004	0.8127	0.8065	0.9634	0.9632	**0.9633**
5	0.8131	0.8176	0.8153	0.8023	0.8090	0.8056	0.9346	0.9350	**0.9348**
6	0.7943	0.8005	0.7974	0.7900	0.8022	0.7961	0.9149	0.9070	**0.9109**
7	0.7787	0.7849	0.7818	0.7671	0.7800	0.7735	0.9396	0.9376	**0.9386**
8	0.7772	0.7826	0.7799	0.7573	0.7664	0.7618	0.9155	0.9165	**0.9160**

TABLE VII: Rebuffering event inferring results with J48, DF and DeepQoE on YouTube

	J48			DF			DeepQoE		
Rebuffering time	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
0	0.7751	0.7962	0.7855	0.7514	0.9794	0.8504	0.7814	0.9657	**0.8638**
1	0.4910	0.4663	0.4783	0.7759	0.3194	0.4525	0.8158	0.4330	**0.5657**
2+	0.1032	0.0919	0.0973	0.0000	0.0000	0.0000	0.9002	0.2082	**0.3382**
Weighted AVG	0.6683	0.6752	0.6717	0.7287	0.7254	0.7270	**0.7954**	**0.7875**	**0.7914**

TABLE VIII: Rebuffering event inferring results with J48, DF and DeepQoE on Bilibili

	J48			DF			DeepQoE		
Rebuffering time	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
0	0.8495	0.8593	0.8543	0.8230	0.9362	0.875971	0.9001	0.9923	**0.9440**
1	0.5923	0.5828	0.5875	0.7055	0.4822	0.572844	0.9685	0.7557	**0.8490**
2+	0.1436	0.1234	0.1328	0.2609	0.0792	0.121501	0.9494	0.5214	**0.6731**
Weighted AVG	0.7640	0.7682	0.7661	0.7763	0.7979	0.7869	**0.9208**	**0.9161**	**0.9184**

and J48. As for the recall and F1 score, DeepQoE reaches more than 90%, which performs best among the three methods. The recall and F1 score of J48 and DF are similar, which are around 80%. The results show that our approach is effective for both YouTube and Bilibili traffic.

Rebuffering events. As for rebuffering events, we adopt $T = 10s$ as the inferring time threshold. We label rebuffering events as 3 types: no rebuffering, rebuffering 1 time, rebuffering 2+ times (rebuffering 2 times and more than 2 times).

Table VII and Table VIII show the rebuffering event inferring results of different identification models on YouTube and Bilibili. No matter how many the rebuffering times are, DeepQoE performs the best among different inferring models. The recall of DeepQoE on no rebuffering event reaches more than 95%. The precision of DeepQoE also outperforms among different inferring models.

The precision of DeepQoE reaches 79.54% on YouTube, which is 12.71% higher than J48 and 6.67% higher than DF. DeepQoE performs the best on Bilibili compared to the other two models, whose precision reaches 92.08% and is 15.68% higher than J48 and 14.45% higher than DF. As for the recall, DeepQoE reaches 78.75% on YouTube, which is the highest among the three inferring models. The recall of DeepQoE reaches 91.61% on Bilibili, which is 14.79% higher than J48 and 11.82% higher than DF. The F1 score of DeepQoE reaches 79.14% on YouTube and 91.84% on Bilibili, which is higher than J48 and DF. The results show that our method is superior to other methods.

Video resolution. For the video resolution inferring, we adopt a fine grained inferring model and set the time window

TABLE IX: Video resolution inferring result comparison of J48, DF and DeepQoE on YouTube

	J48			DF			DeepQoE		
Video Resolution	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
144p	0.7320	0.7465	0.7392	0.6999	0.9889	0.8197	0.7301	0.9617	0.8300
240p	0.2735	0.2629	0.2681	0.4159	0.0316	0.0588	0.8242	0.3318	0.4731
360p	0.5721	0.5553	0.5636	0.9310	0.4624	0.6179	0.8447	0.5870	0.6926
Weighted AVG	0.6199	0.6240	0.6218	0.6936	0.7180	0.6458	**0.7736**	**0.7548**	**0.7289**

TABLE X: Video resolution inferring result comparison of J48, DF and DeepQoE on Bilibili

	J48			DF			DeepQoE		
Video Resolution	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
360p	0.8819	0.8981	0.8899	0.8430	0.9519	0.8942	0.9288	0.9932	0.9599
480p	0.2785	0.2554	0.2664	0.2299	0.0806	0.1193	0.8747	0.4784	0.6185
720p	0.3459	0.3147	0.3296	0.2390	0.0692	0.1073	0.9446	0.6592	0.7765
1080p	0.4456	0.4247	0.4349	0.3106	0.3420	0.3255	0.9717	0.8477	0.9055
Weighted AVG	0.7819	0.7902	0.7859	0.7563	0.7981	0.7766	**0.9277**	**0.9285**	**0.9213**

$T = 10s$. DeepQoE can estimate the specific resolution of the videos. Since the connection bandwidth is limited between our server and YouTube server, the resolution of YouTube videos ranges from 144p to 360p (i.e., 144p, 240p, and 360p).

Table IX shows the precision, recall and F1 score of different models on YouTube. DeepQoE performs best in precision, recall and accuracy comparing with other methods. The precision of our method on YouTube reaches 77.36%, which is 15.37% higher than J48 and 8% higher than DF. As for the recall, DeepQoE reaches 75.48%, which is 13.08% higher than J48 and 3.68% higher than DF. The F1 score of DeepQoE is 72.89% on YouTube, which is the highest score among the three inferring models. The video resolution of Bilibili ranges from 360p to 1080p (i.e., 360p, 480p, 720p, and 1080p).

Table X shows the inferring results of video resolution on Bilibili. DeepQoE outperforms other inferring models in precision, recall or F1 score all the time. The precision of DeepQoE reaches 92.77%, which is 14.58% higher than J48 and 17.14% higher than DF. The recall of DeepQoE also reaches more than 90%, which is the highest among three different methods. The F1 score of DeepQoE is the highest score among these inferring models, which reaches 92.13%. The experimental results show that our method is applicable to different datasets, which has the best identification results among different methods.

E. Evaluation of Time Cost

We evaluate time cost from three aspects: feature extraction, training, and testing stage. Table XI presents the details of the time cost of the different inferring models.

TABLE XI: Time complexity

Approaches	Training Time (s)	Testing Time (ms)		
		Feature Extraction	Inferring Time	Total Time
J48	10.266	73.961	0.095	74.056
DF	64.095	3.401	1.575	4.976
DeepQoE	35.899	4.057	0.099	**4.156**

Feature extraction stage. As J48 needs to calculate statics from the video flow, it takes the longest time in the feature extraction stage, which is 15 times higher than DeepQoE. DF and DeepQoE takes a similar time to extract features. DeepQoE needs 4.507 milliseconds to extract features from the encrypted flow and it's suitable for real-time video QoE metrics inferring.

Training and testing stage. DF takes the longest time to build a QoE inferring model in the training stage, which takes 64.095 seconds, 28.196 seconds higher than DeepQoE and 53.288 seconds higher than J48. When a video QoE metrics model is constructed, J48 and DeepQoE take a similar time to infer an instance. DeepQoE only takes 0.099 milliseconds in the inferring stage. DF takes the longest time to infer an instance. When inferring the unknown data, our method needs only 4.156 milliseconds in the testing stage, which is the lowest time cost among the three methods and is suitable for real-time identification.

VI. Discussion

In this paper, CNN is selected to construct DeepQoE as it fits our scenario better, e.g., we can quickly retrain the classifier when input characteristics change, and thus can react more quickly to the changes in video delivery strategies such as quality adaption logic. Several deep learning tools, such as Long Short-Term Memory (LSTM) and reinforcement learning, have been used in networking field. We expect that DeepQoE can be extended with these advanced tools. For instance, LSTM is suitable for correlation analysis of time series with sequential connections. As for the video steaming traffic, the RTTs of consecutive segments may have a correlation, making it potentially suitable to incorporate LSTM in constructing high-accurate classifiers. Reinforcement learning focuses on representation learning and goal-directed behavior, which can also be applied to QoE measurement with tailored designs such as reward functions. It would be an interesting issue to explore the possible extension of DeepQoE with other deep learning tools.

In our datasets, we consider only the video traffic of YouTube and Bilibili encrypted by HTTPS. However, Deep-

QoE can be generalized to other streaming protocols (e.g., WebRTC) and transport protocols (e.g., QUIC), as long as the relationship between packet-level RTT and QoE still holds. We will leave these attempts as the future work.

VII. Conclusion

In this paper, we proposed DeepQoE, which is a CNN-based classifier to measure real-time video QoE metrics from encrypted traffic. Several crucial QoE metrics were considered, including startup delay, rebuffering events, and video resolution. We customized an appropriate input format based on the observation of the relationship between RTTs of upstream packets and QoE metrics, designed activation functions tailored to our input format, and proposed effective protection against over-fitting. We evaluated DeepQoE on real-world traffic datasets collected from YouTube and Bilibili. The results demonstrated the superiority of DeepQoE over the state-of-the-art methods, in terms of measurement accuracy and time overhead. In future work, we will further extend DeepQoE with other deep learning tools and evaluate its effectiveness on more video content providers.

Acknowledgments

This work is partially supported by Beijing Natural Science Foundation under Grant 4192050, National Natural Science Foundation of China under Grants 61972039, 61932016, and 61872041, China National Funds for Distinguished Young Scientists under Grant 61825204, Beijing Outstanding Young Scientist Program with No. BJJWZYJH012019100030l1, Beijing National Research Center for Information Science and Technology (BNRist) with No. BNR2019RC01011, Science and Technology Planning Project of Guangdong Province under Grants LZC0023 and LZC0024, PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019), and Technology Innovation Program of Beijing Institute of Technology (3052019023).

References

[1] Sandvine, "Global internet phenomena report," https://www.sandvine.com.phenomena, 2018.

[2] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order markov chains and application attribute bigrams," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 8, pp. 1830–1843, 2017.

[3] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 1331–1339.

[4] M. Shen, J. Zhang, L. Zhu, K. Xu, X. Du, and Y. Liu, "Encrypted traffic classification of decentralized applications on ethereum using feature fusion," in *Proceedings of the International Symposium on Quality of Service, IWQoS 2019, Phoenix, AZ, USA, June 24-25, 2019*, 2019, pp. 18:1–18:10.

[5] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying youtube qoe based on encrypted network traffic," *Multimedia Tools Appl.*, vol. 76, no. 21, pp. 22 267–22 301, 2017.

[6] Y. Chen, K. Wu, and Q. Zhang, "From qos to qoe: A tutorial on video quality assessment," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 2, pp. 1126–1165, 2015.

[7] "Youtube," https://www.youtube.com/.

[8] "Bilibili," https://www.bilibili.com/.

[9] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 ACM on Internet Measurement Conference, IMC 2016, Santa Monica, CA, USA, November 14-16, 2016*, 2016, pp. 513–526.

[10] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "Mimic: Using passive network measurements to estimate http-based adaptive video qoe metrics," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2017, pp. 1–6.

[11] T. Mangla, E. Halepovic, M. H. Ammar, and E. W. Zegura, "emimic: Estimating http-based video qoe metrics from encrypted network traffic," in *Network Traffic Measurement and Analysis Conference, TMA 2018, Vienna, Austria, June 26-29, 2018*, 2018, pp. 1–8.

[12] V. Krishnamoorthi, N. Carlsson, E. Halepovic, and E. Petajan, "Buffest: Predicting buffer conditions and real-time requirements of http (s) adaptive streaming clients," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 76–87.

[13] C. Gutterman, K. Guo, S. Arora, X. Wang, L. Wu, E. Katz-Bassett, and G. Zussman, "Requet: real-time qoe detection for encrypted youtube traffic," in *Proceedings of the 10th ACM Multimedia Systems Conference, MMSys 2019, Amherst, MA, USA, June 18-21, 2019*, 2019, pp. 48–59.

[14] P. Schmitt, F. Bronzino, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *arXiv preprint arXiv:1901.05800*, 2019.

[15] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, 2014.

[16] T. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: buffer-based rate adaptation for HTTP video streaming," in *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, Hong Kong, China, August 16, 2013*, 2013, pp. 9–14.

[17] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, 2007.

[18] T. Hofeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *2011 IEEE International Symposium on Multimedia*, Dec 2011, pp. 494–499.

[19] B. Lewcio, B. Belmudez, A. Mehmood, M. Wltermann, and S. Mller, "Video quality in next generation mobile networks perception of time-varying transmission," in *2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, May 2011, pp. 1–6.

[20] M. Shen, Y. Liu, K. Xu, X. Du, and N. Guizani, "Optimizing feature selection for efficient encrypted traffic classification: A systematic approach," *IEEE Network*, 2019.

[21] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang, "Webpage fingerprinting using only packet length information," in *2019 IEEE International Conference on Communications, ICC 2019, Shanghai, China, May 20-24, 2019*, 2019, pp. 1–6.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 2818–2826.

[23] P. Sirinam, M. Imani, M. Juárez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, 2018, pp. 1928–1943.

[24] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456.