NEIVA: Environment Identification based Video Bitrate Adaption in Cellular Networks

Chunyu Qiao
Tsinghua University
qiaocy16@mails.tsinghua.edu.cn

Gen Li
Tsinghua University
g-li17@mails.tsinghua.edu.cn

Jiliang Wang
Tsinghua University
jiliangwang@tsinghua.edu.cn

Yunhao Liu
Michigan State University & Tsinghua University
yunhao@greenorbs.com

ABSTRACT

With the popularization of advanced cellular networks, mobile video occupies nearly three quarters of cellular network traffic. While previous adaptive bitrate (ABR) algorithms perform well under broadband network, their performance degrades in cellular networks due to throughput fluctuation. Through real world 4G/LTE network measurement, we find that throughput in cellular networks exhibits high fluctuation. It follows Markov behaviors with different states and different transition probability among states. We further find that the transition probability is stable along time but varies significantly under different environments. This inspires us to design ABR algorithms by improving throughput prediction in cellular networks. We propose NEIVA, a network environment identification based video bitrate adaption method in cellular networks. NEIVA trains a network environment identifier based on throughput data and trains a hidden Markov model (HMM) based throughput predictor for different environments. In online video bitrate selection, NEIVA utilizes the environment identifier to select the model for corresponding environment. Then NEIVA predicts future network performance by combining offline model and online throughput data. We implement NEIVA with MPC and evaluate it in real environment. The evaluation results show that with manually identifying environment, NEIVA improves 20% – 25% bandwidth prediction accuracy and 11% – 20% QoE improvement over the baseline predictors. With online environment identification, online NEIVA achieves 3.8% and 11.1% average QoE improvement over MPC and HMM, respectively.

CCS CONCEPTS

• **Networks** → **Network algorithms**; **Network measurement**; • **Theory of computation** → **Online algorithms**; • **Computing methodologies** → *Machine learning*.

KEYWORDS

4G/LTE bandwidth measurement, quality of experience, adaptive bitrate selection, environment identification, throughput prediction

ACM Reference Format:
Chunyu Qiao, Gen Li, Jiliang Wang, and Yunhao Liu. 2019. NEIVA: Environment Identification based Video Bitrate Adaption in Cellular Networks. In *IEEE/ACM International Symposium on Quality of Service (IWQoS '19), June 24–25, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3326285.3329038

1 INTRODUCTION

The HTTP-based video streaming has become the major traffic on the Internet [28]. With the popularization and rapid development of cellular networks (e.g., 4G/LTE), more and more people watch videos via cellular networks easily. According to cisco report, mobile video will generate more than three quarters of mobile data traffic by 2021 [9]. Under this trend, engaging users watching videos achieve significant benefits [11] for video providers. Thus delivering video streaming of high quality of experience (QoE) becomes increasingly important. ABR algorithm is one of the most important techniques to control the bitrate selection of each video chunk according to the network condition and thus improve the QoE.

With the rapid development of cellular network, the throughput under cellular network e.g., 4G/LTE bandwidth achieves up to tens of *Mbps*. This benefits video streaming application under cellular networks, such as live streaming by smart phones for anchors (e.g., Douyu [2], Twitch [4], requiring up to 10 Mbps downloading network throughput.). Despite of the increasing trend for mobile videos, many video providers still use ABR algorithms regardless of cellular network property [7, 10, 23]. The algorithms differ in strategies for video bitrate adaption to network conditions. Such difference also reflects the key challenge in ABR algorithms design: adapting to various and dynamic network conditions (i.e., challenged to design efficient and accurate network models for throughput prediction) [16, 31, 32, 34].

The wild throughput fluctuation in cellular network makes it challenging for inferring actual network state [9, 13], leading to QoE degradation. For example, state-of-the-art approach MPC [33] predict bandwidth by harmonic mean of history bandwidth, which results in low prediction accuracy when the network fluctuates over time. CS2P [31] finds that Markov property exists in broadband throughput and changes over time, while in cellular networks the property may not hold. OBOE [8] models the network trace as piece-wise stationary process, but it fails to correctly segment the throughput trace into pieces of stationary trace under cellular network.

To address the issue, we first measure real world bandwidth of cellular networks for 4 months. We find that the fluctuation of $4G$/LTE throughput makes it difficult to predict network environment state, e.g., low bandwidth prediction accuracy in MPC [33] and invalidation of piece-wise stationary network model in OBOE [8]. From the measurement data, we further find that the bandwidth throughput exhibits Markov stateful behaviors for different environments (indoor, bus and the open air). The characteristic can be modeled as Markov process to infer network state, which motivates us to predict network states by modeling.

Accordingly, we propose NEIVA, a network environment identification based video bitrate adaption approach for ABR improvement. NEIVA utilizes Markov property under specific environment to improve ABR performance, i.e., improving throughput prediction accuracy in MPC [33] and neural model training in Pensieve [22].

Utilizing Markov property online requires model updating to predict bandwidth accurately, like CS2P [31]. It however induces huge time cost for training and updating the model. We find that the Markov property under a certain network environment remains relatively stable over time, which means the network model training can be processed offline and does not have to update online. NEIVA offline trains HMM of specific environment, such as indoor, bus and open air. For online adaption, NEIVA infers the environment from network bandwidth and selects corresponding HMM model for throughput prediction, which is used to decide video bitrate selection in ABR algorithms. To accurately specify corresponding model, NEIVA conducts multi-layer perceptron classifier for online inference. By utilizing both the network and identification model trained offline, NEIVA does not require online model updating, which makes it feasible for deployment at server side.

We implement NEIVA based on MPC [33], i.e., NEIVA incorporates offline model training under different network environment scenarios and predicts throughput by online preferring corresponding model. We conduct HTTP based video emulator mahimahi [24] for video playback and evaluate NEIVA under $4G$ traces with manual model selection. For throughput prediction, NEIVA improves the overall accuracy by $20\% - 25\%$ over the existing throughput predictors. In QoE evaluation under MPC, NEIVA achieves $11\% - 20\%$ overall QoE improvement. Further, we design online NEIVA by integrating MLP classifier into NEIVA for online environment identification. Online NEIVA achieves $87\% - 99\%$ performance of offline NEIVA across different environments and outperforms MPC and overall HMM (trained over the whole dataset) by 3.8% and 11.1%, respectively.

The remainder of the paper is organized as follows: Section 2 presents the background of video adaption and motivation for ABR design under $4G$/LTE networks. Section 3 introduces our $4G$ trace collection and analysis. Section 5 derives formal model design of NEIVA and parameter configuration. Section 4 shows the overview of NEIVA. Section 6 evaluates NEIVA's performance for throughput prediction and QoE metrics. Section 7 summarizes related work on video bitrate selection. Section 8 concludes our work.

2 BACKGROUND AND MOTIVATION

2.1 Video Bitrate Adaption

Recent years have witnessed a rapid increase in mobile applications and advanced techniques in cellular networks such as $4G$/LTE. In 2016, $4G$ has already carried 69% of the total mobile traffic and represents the largest share of mobile data traffic by network type. It is growing faster than other networks to represent 79% of all mobile data traffic by 2021. As the major traffic, mobile video will generate 78% of mobile data by 2021 [9].

The HTTP-based video streaming (DASH) [10] is one of the most popular video delivery techniques. By delivering video via HTTP, DASH is compatible with heterogeneous networks and mobile devices, such as iOS and Android mobile devices. In DASH, the video is chopped into chunks with the same time length, e.g., 4 seconds. Each video chunk is encoded into different bitrates copies and stored in video server or CDN. When the client watches video, the video player requests video chunk of selected bitrate one by one.

The fluctuation of networks may lead to low bitrate video playback or video rebuffering, and thus result in low QoE. For instance, the requested video chunk will take a long time for downloading when bandwidth is poor, leading to stalled video playback (rebuffering). When rebuffering occurs, the DASH video player selects video chunk of small bitrate. When network condition changes, the video player should be able to adapt the bitrate.

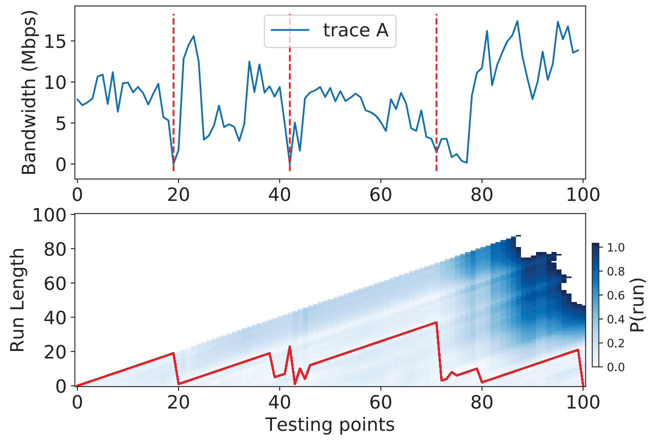
There exists a trade off in promoting QoE, such as low rebuffering time vs. high bitrate. Previous work proposes QoE metrics including video bitrates, rebuffering and changes of bitrates [11, 20]. The QoE metrics have been formulated into a QoE quantization function, each of which is calculated as a linear part [33]. In addition, previous work has proposed network models and prediction methods to maximize QoE. In ABR algorithms design, it has been proved that high bandwidth prediction accuracy can improve QoE [8, 31, 35].

2.2 Limitations of Prior Work under Cellular Networks

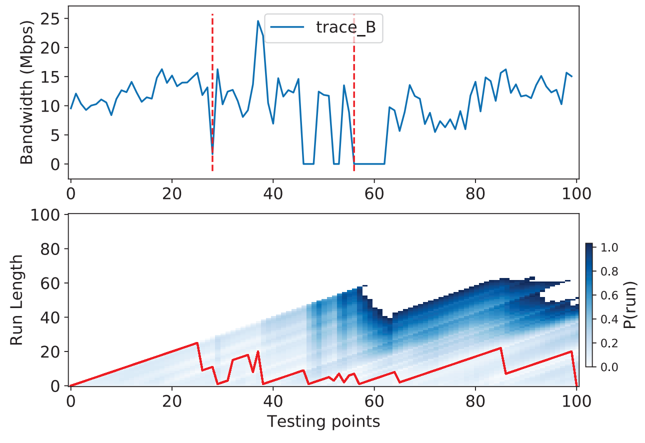
To adaptively select bitrate for video delivery, ABR algorithms are proposed to deal with network throughput fluctuations. These algorithms are designed across different network conditions and improve QoE from the following aspects: (1) online throughput prediction: MPC [33], CS2P [31], (2) neural network model training of ABR: Pensieve [22], and (3) online parameter tuning with capturing change of network states: Oboe [8].

CS2P [31] and OBOE [8] show significant improvement by capturing the network state under broadband networks. However, the emergence of advanced cellular network, such as $4G$, introduces new challenges. The throughput of $4G$ networks can achieve high bandwidth (up to tens of *Mbps*) and fluctuates significantly across environments. Even the environment is the same, e.g., in bus, the network conditions also change as moving of the bus and changing of passenger number.

We collect cellular network traces and show the limitations of prior work as follows.



(a) Stationary trace



(b) Non-stationary trace

Figure 1: Illustration of piecewise stationary model failing to segment a throughput trace into pieces of stationary process: segment stationary trace A, non-stationary trace B into piecewise traces and test their stability. We conduct Bayesian online change point detection technique to segment the traces. The bottom plot shows the probability $P(run)$ over the current run length (or time since the last change point [1]) at each point of a network trace. Notice that the drops of red solid lines to zero run-length correspond well with the abrupt changes in the throughput. We denote the piecewise traces shown between the vertical red dotted lines in the top plots. All the piecewise traces are non-stationary.

(1) Rough prediction methods. Previous algorithm such as MPC [33] conducts simple throughput method, e.g., last sample (LS) and harmonic mean (HM) obtaining low prediction accuracy rates. In our 4G/LTE data set, the 50-percentile prediction errors for harmonic mean and last sample are above 20.0% while 75-percentile error is above 39.0% (as shown in evaluation part, Figure 8).

(2) Heavy overheads for online adaption. Prior work, CS2P [31] collects traces for training Markov model online and induces extra overheads. We compare the time cost between online model CS2P T_{online} (online trace collecting and model training) and offline model $T_{offline}$ (offline training[1]). While CS2P collects traces at the server, we set the collecting trace number range from 200 to 500. As shown in table 1, the time cost of CS2P is hundred to thousand times higher than offline and still grows with increasing number of bandwidth traces. Such limitation makes it unsustainable to proceed at the server side.

trace amount	200	250	300	350	400	450	500
$T_{online}/T_{offline}$	446	518	574	619	705	817	932

Table 1: Time cost comparison for online model training and throughput prediction.

(3) Network state identification based ABR approach, Oboe [8] proposes piecewise-stationary model for network throughput. It assumes network traces behave as piecewise stationary process: each network trace is in one of several states, each of which can be identified according to the stationarity (a stationary process means its mean and variance remain unchanged over time [27]). Oboe shows that a network throughput trace can be segmented into pieces of time stationary series (piecewise traces) and utilize them for network state modeling for further ABR algorithms improvement. To evaluate its performance, we utilize the referenced technique in Oboe: change point detection [1] and AD-Fuller test [14] techniques under our data set.

As shown in Figure 1, we show an example of a segmentation for stationary (traceA) and non-stationary (trace B) throughput traces. The traces of pieces of sub-traces are segmented by Bayesian change point detection [1]. Then we take AD-Fuller test [14] to test the stationarity of each piecewise traces. As a result, they are both segmented into non-stationary piecewise traces which are opposite of OBOE's model.

To study the piecewise-stationary property under cellular networks, we conduct extensive experiments over our dataset. As shown in Table 2, the non-stationary trace proportion is 35.89%, 59.10% and 65.77% under indoor, bus and the open air, respectively. The result shows that after segmenting the original trace into piece traces, the percentage of non-stationary traces changes to 44.95%, 55.62% and 72.43%, respectively. This means that under cellular networks, the piecewise-stationary model does not always hold. Given that Oboe conduct simultaneous traces for evaluation, we infer the reason is that the network is dynamic under cellular networks and cannot be simply segmented into piecewise stationary process.

We show the limitations of previous works under cellular networks. For video bitrate adaption, we need to examine the characteristic of cellular networks, and utilize it for ABR performance improvement.

[1]Offline model can be utilized for bandwidth prediction directly.

Table 2: $4G$ **bandwidth measurement traces summary.**

Test time range	SpeedTest May 25 2018 to September 30 2018[1]		
Env. (test interval)	indoor (5-10s)	open air (1-3s)	bus (1-2s)
Trace number	6249	742	670
5% DL	24.46	17.09	20.56
50% DL	15.07	5.77	7.28
95% DL	2.98	0.23	0.00
Avg. bandwidth	14.45	7.02	8.26
Std. Dev.	6.61	6.19	6.10
Non-stationarity	35.89%	65.77%	59.10%

Trace number: each trace consists of 100 testing points.
DL: downlink throughput (*Mbps*). 5%, 50%, 95% are percentiles.
Non-stationarity: the proportion of non-stationary traces.

3 DATA COLLECTION AND ANALYSIS

In this section, we first show the measurement of cellular network traces, i.e., $4G$/LTE network bandwidth in real world and presents the wild throughput fluctuations. In addition, to study the characteristics of $4G$/LTE network throughput across environments, we conduct time series analysis and clustering techniques and find distinct Markov property under specific environment.

3.1 Collecting 4G/LTE Traces.

Cellular network covers a large range of environments, such as indoor, open air and mobile vehicles. $4G$/LTE network has taken more than three quarters of the mobile traffic and its high bandwidth e.g., up to tens of *Mbps* makes it an ideal medium for high quality video delivery. Thus we focus on measuring $4G$ bandwidth traces in this paper.

Although open cellular network bandwidth data can be accessed on the Internet, the data is not sufficient enough due to lack of environment information and coarse measurement intervals [13, 17]. To study the $4G$/LTE traces, we collect $4G$ bandwidth data with smart phones on our own. We utilize the open source speed-test code [29] and runs QpythonL [6] on smart android phones.

The speed-test codes are designed for average downlink and uplink throughput measurement in $10 - 20$ seconds. To measure $4G$ trace for analysis, we make the following modifications:

- Modify measurement intervals: we modify the function *shell* and set timers to download test samples with intervals range between 1-10 seconds.
- Reduce download time cost: the original downloading sample file number of Speedtest is up to 4 (copies) $*$ 8 (different sizes of the sample files) and result in large time cost for one testing point (e.g., up to tens of seconds). We reduce the testing samples to $1 * 3$ by modifying the parameter *sizes* and *download* in function *get_config*.
- Record environments: we set environment type as one input parameter and record the environment information.
- Save test results: after every test session (100 test points), the smart phone sends the testing results to our *Node.js* server.

For generalization concerns, we measure the $4G$ bandwidth data in two cities Suzhou and Beijing, China across various environments, including indoor (laboratories, offices, bedrooms), moving buses (about 20 different lines) and open air (streets, parks, playgrounds). We find that there sometimes occurs $0Mbps$ bandwidth testing points. This is because (1) network conditions are too poor and result in network failures and (2) testing failures, the server may fail to response for a time when the clients frequently requiring one sample downloading file. However, the testing failures will generate continuous $0Mbps$ testing points even with several retransmission attempts. We process the data by eliminate the invalid traces consisting of continuous $0Mbps$ testing points last for more than 1 minute.

The results are summarized in table 2. Note that the standard deviation of bandwidth is nearly the average bandwidth under open air and bus, the huge fluctuation of network bandwidth also reflects on the $5, 50, 95$ percentile downlink throughput, ranges from $0.0Mbps$ to $24.46Mbps$.

3.2 Data Analysis

3.2.1 Time series analysis. To study the characteristics of the throughput traces, we deal with the network bandwidth in a trace by time series analysis technique. We first make scatter plot of the throughput traces for consecutive testing points, with x-axis testing time T and y-axis testing time $T + 1$. We randomly select 100 traces from the data set each environment and make plots separately for illustration.

As shown in Figure 2, the scatter plots present different characteristics across network environments. Specifically, (1) Bus: the throughput is distributed evenly between $0Mbps$ to $25Mbps$, follows stateful behaviors, e.g., throughput ranges between 3-12$Mbps$ mostly transitions to 3-12$Mbps$. There also exists possible network failing behavior such as throughput 0-23$Mbps$ transitions to $0Mbps$. (2) Indoor: the throughput is more likely centralized with high bandwidth, e.g., the throughput transitions within several states between $14Mbps$ and $25Mbps$. (3) Open air: While the states for the open air range from $0Mbps$ to $20Mbps$, the throughput tends to centralize at lower bandwidth (e.g., 0-8$Mbps$) and basically transitions in a narrow range.

As a result, the throughput exhibits distinct stateful characteristics across different environments. The states can be captured as Markov property, which means the consecutive throughput clusters in several states.[2]. As we randomly choose traces from the data set last for 4 months, such property persists the similar behaviors under the specific environment (indoor, bus, open air) and distinguishes from other environments.

In CS2P [31], it demonstrates that the network throughput traces under the same ISP, city and server may show the stateful characteristic, namely Markov property for the TCP fair-sharing model. However, it points out that the property is unstable over time and demands the network traces collected within a short time period like within 1 hour. Our findings reveal that the Markov property differs across environments, but it persists over time under the same

[1]We measure the bandwidth at Beijing and Suzhou in China, the bandwidth traces of indoor are tested all the time while others are measured partially of the time.
[2]We manually mark the states as circles in the figure for illustration.

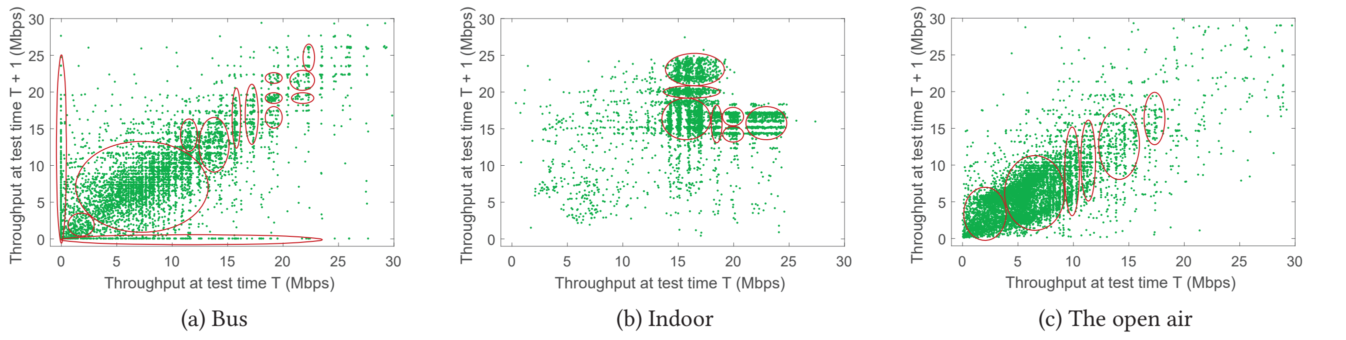


Figure 2: Consecutive throughput scattering plots (Markov property) of $4G$ network traces under bus, indoor and the open air, where red circles indicate similar network state for the inner points.

environment. Since that we measure the throughput by end devices directly, we can intuitively conjecture the difference across environments for the following reasons: (1) For throughput in moving bus, the rapid network environment changes cause it transition in a vast range, even face possible network failures. (2) For throughput in indoor, the network environment is relatively stable and thus may experience good network conditions. (3) For throughput in the open air, the network environment changes, but not as violently as bus in that our tester walks around places like playground.

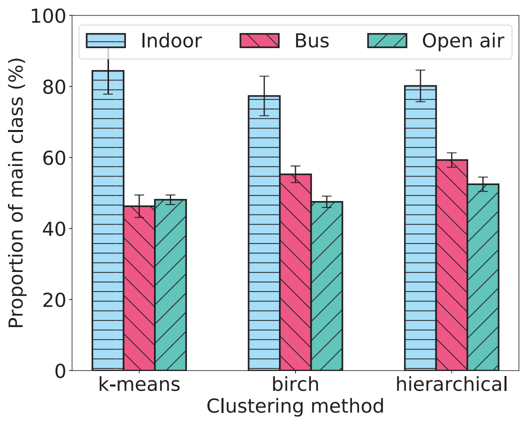


Figure 3: The percentage proportion for the main class. The main class means it contain more than 80 percent throughput traces under the same environment. The error bars represent the standard deviation for the value.

3.2.2 Network environment based clustering property. The Markov property across different environments inspires us conducting clustering techniques from the raw throughput traces. We conduct three common clustering techniques, k-means, birch and hierarchical. For clustering job at one time, we randomly select 500 traces equally from three environments and cluster them into unsupervised groups.

As we repeat the clustering experiment for 100 times, the result is shown in Figure 3. The group of more than 80% throughput traces containing the same environment is counted as the main class. With manually checking out the groups, we learn that the throughput traces under the same environment tend to be clustered into one group (denoted as main class). Though the proportion for bus and outdoor is lower, e.g., 53% − 60% in hierarchical clustering, the overall proportion of main classes is above 65%.

In summary, we find that the wild fluctuation exists in cellular network throughputãĂĆ While the overall Markov property among the whole data set does not persist over time, the Markov property can be distinguished from different network environment and hold stable at long time range. This make it possible for utilization in predicting future throughput. Such property reveals that specifying the characteristics of the specific environment can benefit for network modeling. This inspires us to utilize it for network throughput prediction by inferring the network environment of throughput trace online. Since we measure the throughput in two cities across different environments, we believe that similar temporal persisted Markov property can be captured in more vast space.

4 OVERVIEW OF NEIVA

In this section, we present the design of NEIVA for offline training and online predicting.

At offline training stage, as shown in the top part in Figure 4, we conduct the following steps:

- Data processing. As shown in data analysis, we capture that the throughput traces generate stateful behaviors and such property can be utilized for building prediction models. To maintain the property, we transform the throughput trace into a series of bandwidth pairs (feature extraction).
- HMM model training. Since the Markov property is distinct from different environments, we conduct the processed data for HMM model training each environment, i.e., indoor, bus and the open air, respectively.
- Environment identifier training. Given the pre-trained HMM models, it is however hard to decide which model to use for online throughput prediction, e.g., a user watches video in bus with HMM model trained from indoor will usually get

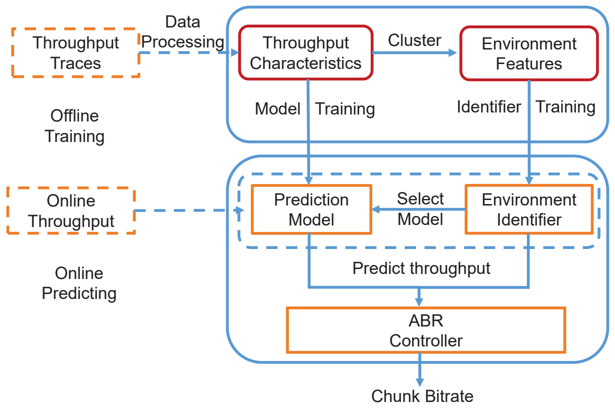


Figure 4: Work flow of NEIVA.

a worse throughput prediction result. To address the issue, we infer environments from throughput traces automatically by deriving supervised classifier technique for online identification.

While at online predicting stage, as shown in the bottom part in Figure 4, we integrate the prediction model and identification classifier as online predictor. The online process for throughput prediction is as the follows: 1) The predictor collects the network throughput online when user watches the video. 2) NEIVA infers the specific environment by the offline classifier and then selects corresponding prediction model. 3) At last it predicts the throughput by the pre-trained HMM model and sends the result to ABR controller.

The video bitrate is determined by the ABR controller. Note that both the environment identifier and prediction model require enough throughput data (e.g., 10 samples) as input, this may result in predictor failing to work at the beginning of video. However, video players such as *dash.js* already record intermediate throughput samples during a chunk download (tens of milliseconds). Thus we can get sufficient throughput samples without incurring overheads.

While we introduce the overview of NEIVA as offline and online stages, it is challenged to implement the offline model and identifier efficiently for online video bitrate adaption. In addition, the parameter configuration for model training need to be further explored. We leave the detail design of NEIVA in the next section.

5 NEIVA DESIGN

In this section, we show how to extract throughput trace features and utilize them for online bandwidth prediction and environment identification.

5.1 Data Processing

As shown in the data analysis, the consecutive throughput of a trace shows stateful behaviors, Markov property. This motivates us to transform the throughput into the stateful points as shown in Figure 2.

We denote the trace $Traces_s$ under a specific network environment as a time series sequence: bandwidth B_i denotes the random

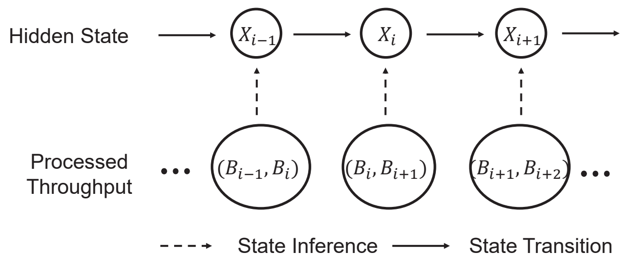


Figure 5: Overview of HMM.

variable of actual throughput b_i, where $i \in \{1, 2, 3, \ldots, n-1\}$, denotes the time sequence and b_i is a non-negative real number. In throughput prediction, we define the absolute prediction error as:

$$Err(\hat{b}_i, b_i) = \frac{|\hat{b}_i - b_i|}{b_i} \tag{1}$$

where $\hat{b}_i$ denotes the predicted value of the actual bandwidth b_i.

Then a trace $Traces$ is transformed into two dimension array $[(b_0, b_1), \ldots, (b_i, b_{i+1})]$, where $i \in \{1, 2, 3, \ldots, n-1\}$, the element pair $p_i = (b_i, b_{i+1})$ of the array can be mapped to the point on $2D$ throughput coordinates. Thus the trace $Traces_s$ is transformed into an array of points Q_s,

$$Traces_s([b_i, b_{i+1}]) \triangleq Q_s([p_i]) \tag{2}$$

5.2 Offline Training

First we introduce modeling Q_s into HMM. The overview of HMM is shown in Figure 5. We assume p_i evolves according to the hidden state variables $X_i \in \mathcal{X}$, where $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, i denotes the time sequence for the random variable of throughput pair $P_i = (B_i, B_{i+1})$ and the number of states $N = |\mathcal{X}|$. The stateful behavior of a certain network environment can be denoted as the hidden states in hidden Markov model (HMM). For example, the passengers in the bus share a bottleneck link and the change of passenger number generate the stateful behaviors. Let $\mathcal{P}$ denotes the probability space. The hidden state behaves as a Markov chain in which the probability distribution of current state only depends on the last state. For the throughput, it can be formulated as:

$$\begin{aligned}\mathcal{P}(X_i = x_i | X_{i-1} = x_{i-1}, \ldots, X_1 = x_1) = \\ P(X_i = x_i | X_{i-1} = x_{i-1})\end{aligned} \tag{3}$$

Given the state X_i, we assume the throughput pair random variable P_i in Equation 2 follows joint Gaussian distribution (the assumption is also used in previous work CS2P [31] and we test for several kinds of distribution, Gaussian fits the best):

$$\begin{aligned}P_i | X_i &\triangleq (B_i, B_{i+1}) | X_i = (b1, b2) \\ &\sim (N(\mu_{b_1}, \sigma^2_{b_1}), N(\mu_{b_2}, \sigma^2_{b_2}))\end{aligned} \tag{4}$$

Given the training data set, typical HMM training process is to solve the optimal transition matrix for the hidden states of number N.

As we transform the throughput traces into sequential point sets, HMM is trained to represent the state transitions. Intuitively, it represents the transitions of the clustered points in Figure 2.

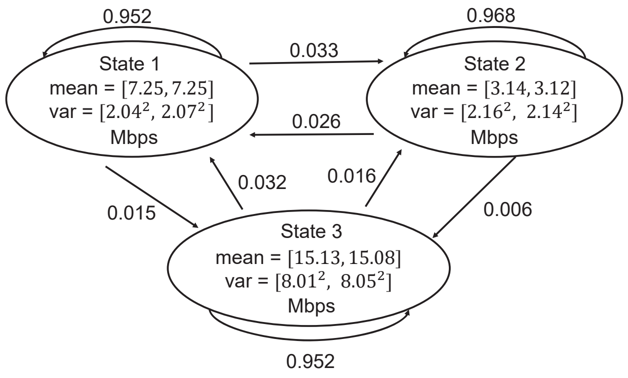


Figure 6: Example of state transition in HMM.

An HMM example with three states is shown in Figure 6. Given the number of states N, we conduct training with throughput traces under the same environment to learn the parameters of HMM. Note that the number of states N needs to be specified beforehand. This leaves a trade-off to select an appropriate N. Smaller N means a simple model but may be insufficient to represent all the possible network behaviors. Larger N models all the network characteristics but however make the model more complicate and even introduces over-fitting problems. We thus conduct experiment to find suitable number of states and discuss the issue in the parameter configuration section.

5.3 Online Throughput Prediction

To predict throughput online, we need to combine HMM and environment identification. As shown in Figure 2, the clustering techniques show that the network states correlates well to the network environments, i.e., indoor, bus, the open air. However, unsupervised clustering cannot infer corresponding environments and thus we conduct classifying techniques to infer the specific network environment and select corresponding model HMM_s.

While the specific model HMM_s is trained offline, the online prediction steps are as follows:

5.3.1 Network environment identification. We conduct multi-layer perceptron (MLP) classifier [26] to identify specific environment. MLP classifier uses back-propagation training over the supervised throughput training set. We denote the classifier trained from the data as CLF. During video playing, NEIVA records the network throughput b_i and infers specific environment by classifier CLF.

$$\hat{s} = CLF_{predict}([b_1, \ldots, b_i]) \tag{5}$$

5.3.2 Throughput prediction. Given the bandwidth data, CLF infers the specific environment s. Then NEIVA selects the specific HMM model HMM_s for bandwidth prediction.

To avoid prediction accuracy degradation due to bandwidth staleness under cellular network [18], we utilize latest bandwidth data for throughput prediction. Let lh denotes the look ahead steps for the hidden state inference. Given previous bandwidth data $b_{i-lh}, \ldots, b_{i-1}$, NEIVA first infers the current hidden state by the maximum likelihood estimate:

$$x_{cur} = arg \max_{x \in \mathcal{X}} P(X_i = x | b_{i-lh}, \ldots, b_{i-1}) \tag{6}$$

Then NEIVA predicts the bandwidth $\hat{b}_{i+1}$ by the first moment estimate from x_{cur}:

$$\hat{b}_{i+1} = \sum_{x \in \mathcal{X}} \mathcal{P}(X_{i+1} = x | x_{cur}) * Sample(\mu_x, \sigma_x^2) \tag{7}$$

where x denotes the possible hidden state of the next bandwidth, function $Sample(\mu, \sigma^2)$ denotes the sample value from Gaussian distribution $N(\mu, \sigma^2)$.

The online prediction algorithm is shown in Algorithm 1, NEIVA infers the network environment from bandwidth, then selects corresponding model HMM_s for hidden state inference according to Equation 6. Finally, NEIVA predicts the future bandwidth by Equation 7.

Algorithm 1 NEIVA Online Prediction

```
Input: prediction $model$, past bandwidth $B_i = [b_{lh}, \ldots b_{i-1}]$
Output: predicted bandwidth $\hat{b_i}$
 1: function PREDICT($B_i$)
 2:     $\hat{b_i} \leftarrow 0$
 3:     $s = CLF_{predict}(B_i)$
 4:     $model = HMM_s$
 5:     $x_{cur} = model.infer(B_i)$
 6:     $\hat{b_i} = GetBandwidth(model, x_{cur})$
 7:     return $\hat{b_i}$
 8: end function
 9: function GETBANDWIDTH($model, x_{cur}$)
10:     $\hat{b_i} \leftarrow 0$
11:     for $i = 1$ to $N$ do
12:         $x_{next} = model.state[i]$
13:         $mean = model.mean[x_{next}]$
14:         $var = model.var[x_{next}]$
15:         $sample_i = Sample(mean, var)$
16:         $b_{next} = model.transmat[x_{cur}][x_{next}] * sample_i$
17:         $\hat{b_i} = \hat{b_i} + b_{next}$
18:     end for
19:     return $\hat{b_i}$
20: end function
```

5.4 Parameter Configuration

There are still two problems in our design: (1) derive the number of hidden states in HMM and (2) learning neural network parameters in MLP. We answer these questions in this section.

5.4.1 Number of hidden states. To configure a suitable number of hidden states N, we compare the fitness among different number of states. Given a network trace, we can calculate the probability of a pre-trained Markov model. This represents the model's fitting degree over the data [19]. We normalize the probability as model scores, with larger score indicating a better fitting.

We separate the data set by network environment and evaluate the performance score correspondingly. The result is shown in Figure 7. We can see that the performance is improved when the number of state increases. Meanwhile, the score increases slowly when the number of states is larger than 30. Hence we select $N = 30$ for offline model training and online bandwidth prediction. Note that our HMM model trains the data over 4 months, we believe that the 30−state HMM model can be deployed easily for online prediction.

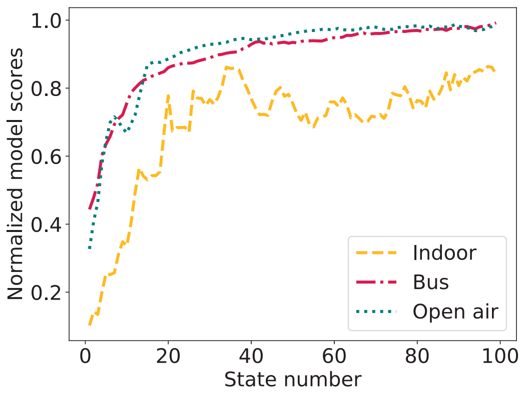


Figure 7: Configuring hidden state number for specific models.

To train the offline Markov model, we randomly select 500 traces equally for three network environments. The data set is partitioned into train set of 70% data and 30% for test set. We choose Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) [21] as the solver. L-BFGS approximates the Hessian matrix which represents the second-order partial derivative of a function. Further it performs parameter updates by approximating the inverse of the Hessian matrix. Thus L-BFGS reduces its iteration cost which deals well with large dimension data calculation and suits our data. We conduct three-layer neurons with size (10, 50, 40) and *relu* as activation function. The experiments are taken over 100 times for generalization concerns. Outperforming clustering results as shown in Figure 3, the result shows that the overall accuracy of MLP achieves 71.6%.

6 EVALUATION

In this section, we first evaluate offline NEIVA's performance on throughput prediction accuracy and QoE metrics score offline. Further, to configure specific HMM model automatically, we integrate environment classifier *CLF* into NEIVA for online adaption.

6.1 Experiment Methodology

Experiment tools: to evaluate NEIVA's performance, we use the following tools for experiments:

- Mahimahi emulator (HTTP-based video player) [24].
- Python machine learning package sklearn for environment identification [12].
- Python HMM learning package for Markov model training [19].

Metrics: we have the following metrics for evaluation:

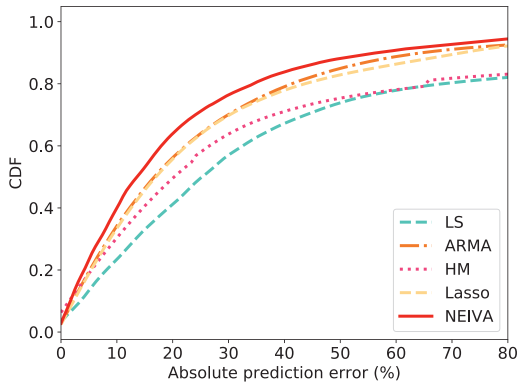


Figure 8: Throughput prediction error comparison.

- Throughput prediction accuracy: we conduct absolute prediction error to denote the accuracy as shown in Equation 1.
- QoE scores: To evaluate the overall QoE, MPC designs a QoE function by combining the QoE metrics, average bitrate, rebuffering time and bitrate switch linearly.

$$QoE_{avg} = \sum_{i=1}^{n} \frac{bitrate[i]}{n} - \beta * \sum_{i=1}^{n} \frac{rebuff[i]}{n} - \gamma * \sum_{i=1}^{n-1} \frac{|bitrate[i] - bitrate[i+1]|}{n-1} \tag{8}$$

where $bitrate[i]$ denotes the bitrate of video chunk i, $rebuff[i]$ denotes the rebuffering time, and β, γ denotes the weight on rebuffering and smoothness penalty, respectively. Similarly to MPC, we set $\beta = 4.3$, $\gamma = 1$.

Video parameters: The video from *DASH* reference client shows that the video bitrate ranges from 250$kbps$ to 15$Mbps$ [5]. In our experiment, the video bitrate levels are 900$kbps$, 2250$kbps$, 3600$kbps$, 5400$kbps$, 8000$kbps$, 12000$kbps$. There are 48 chunks of 4 seconds video. Packet payload portion and link RTT are set to 95% and 80ms respectively.

6.2 Throughput Prediction Improvement

We first evaluate offline NEIVA predictor (with manually environment identification) with baseline algorithms. We train HMM models of 30 states offline specifically for different network environments: indoor, bus and open air.

We evaluate NEIVA under our 4G/LTE throughput data set. The HMM models are trained under 70% throughput traces (training set) of the whole data set and test under the remaining 30% traces (test set). We compare our predictors to the following baseline approaches: Last Sample (LS), Harmonic Mean (HM [33]) and machine-learning based approaches Auto Regression Moving Average (ARMA) and Lasso Regression (Lasso [18]).

The result of the comparison is shown in Figure 8. Comparing to other predictors, NEIVA decreases at least 20% prediction error. More specifically, the 75-percentile error of NEIVA is 27.0%, and the 50-percentile error is 13.5%. Note that the wild fluctuation in

4G/LTE networks makes it challenging for throughput prediction. We believe that NEIVA effectively improves the prediction accuracy over the baseline predictors.

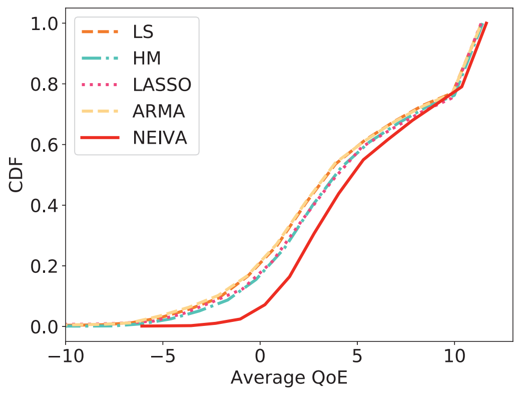


Figure 9: Comparing NEIVA with existing throughput predictors on the overall QoE. Average QoE values are shown for each predictor.

6.3 Video QoE Improvement

While NEIVA performs well in throughput prediction, we evaluate its performance on video QoE in this section. We conduct experiments on 1500 network traces for each environment scenario. We implement the baseline predictors into MPC [33] and record the QoE metrics of each trace. The result is shown in Figure 9, the quantized QoE is calculated according to Equation 8, and the average QoE is the average of all chunks under the same trace. As we can see, NEIVA achieves better QoE over the baseline predictors, i.e., 11.3% – 20.0% overall QoE improvement.

To further study NEIVA's performance, we record the individual component in QoE function and compare the results across different environments. The result is shown in Figure 10. From the result, NEIVA achieves the lowest rebuffering penalty across different environment, especially in bus and open air, where the throughput fluctuation is wild. Note that these predictors are implemented in MPC, we can learn the reason for NEIVA's improvement over existing predictors. With superior throughput prediction accuracy, NEIVA effectively detects network throughput drops and bursts. Thus NEIVA reduces significant rebuffering time with little bitrate loss.

6.4 Online NEIVA Evaluation

While offline NEIVA (manually environment identification) outperforms previous predictors, we also implement NEIVA with network environment identification automatically. We integrate *CLF* for automatic environment identification into NEIVA, namely online NEIVA. Note that for online identification, fast switching of network environment may reduce online NEIVA's performance. However, we think that such situation seldom occurs in real world.

Online NEIVA adapts to video bitrate selection as the following steps: First, *CLF* identifies network environment from history bandwidth. Next, NEIVA selects corresponding HMM model to predict the hidden state of current throughput. Then NEIVA calculates the expected throughput according to the transmission matrix, as shown in Algorithm 1.

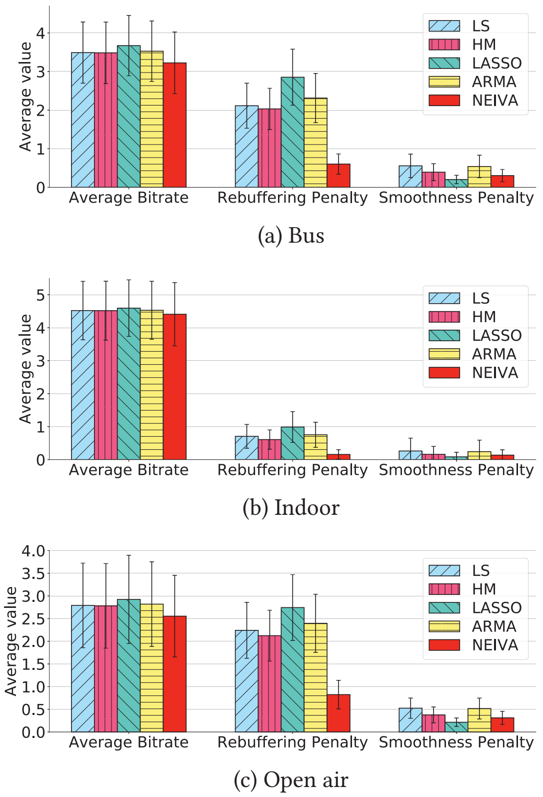


Figure 10: Comparing NEIVA with existing throughput predictors through the individual QoE metrics across different networks. Error bars denote the standard deviation for the average value.

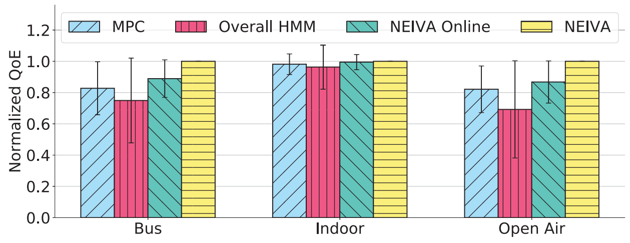


Figure 11: Comparing online NEIVA to offline NEIVA (with manually environment identification), original MPC and overall HMM. Normalized QoE of each ABR algorithm is listed. Error bars denote the standard deviation on the normalized QoE.

To show the necessity for environment identification, we also evaluate the performance of overall HMM (training over the whole data set for all network environments). We integrate NEIVA into MPC and compare its performance with offline NEIVA, original MPC (HM predictor) and overall HMM. The result is presented in

Figure 11. The NEIVA Online shows better performance than MPC and overall HMM. NEIVA Online achieves 86.8% – 99.7% quantized QoE of offline NEIVA across different environments.

7 RELATED WORK

Bandwidth measurement: There are approaches use packet-level probing to estimate the available bandwidth and the capacity of Internet paths (e.g., [3], [15]). CS2P [31] uses network trace set directly from video server, which does not require path information like trace route. NEIVA collects cellular network traces with smart phones by modifying speed-test code [29]. Besides the end to end downlink throughput, NEIVA also collects bandwidth with several measurement intervals and environment information (e.g., indoor, bus and open air).

Throughput prediction in ABR algorithms: Prior work MPC, BOLA [30, 33] conduct simple throughput prediction methods, such as moving average and harmonic mean. CS2P [31] proposes hidden Markov based models to cluster clients at server side and conduct it to improve throughput prediction. When a client receives video chunk from server, the server is required to collect all the throughput traces of the cluster (e.g., consisting clients of a city) in every hour. Cellscope [18] conducts Lasso regression for throughput prediction in cellular networks. Oboe [8] models network throughputs as piecewise-stationary process. It tunes the parameter in ABR algorithms according to the network state inferred under its model.

QoE metrics: The metrics of video QoE have been popular studied recently. The metrics such as video rebuffering, average bitrate, rendering quality, bitrate switch, video flickering can be perceived by users and sensitive to users QoE, which directly relates to user engagement [11] [25]. In ABR algorithms design, the metrics for rebuffering, average bitrate and bitrate switch are usually considered for they are affected by network condition.

8 CONCLUSION

With the increasing popular 4G/LTE network techniques, video streaming over cellular networks occupies nearly three quarters of the Internet traffic. However, the wild fluctuation of network throughput induces challenges for prior ABR algorithms, for QoE degradation caused by low throughput prediction accuracy. To address the issue, we propose NEIVA, a network environment identification based approach to effectively capture network state online. Namely, we propose NEIVA, an online network environment identification based approach for ABR improvement. We evaluate NEIVA with existing methods and the results show that NEIVA achieves 11.3% – 20.0% QoE improvement.

9 ACKNOWLEDGMENTS

This work is in part supported by National Natural Science Fund China for Excellent Young Scholars (No. 61722210), NSFC key program No. 61532012, NSFC No. 61572277, 61529202.

REFERENCES

[1] Aug, 2018. Bayesian Changepoint Detection. https://github.com/hildensia/bayesian_changepoint_detection. (Aug, 2018).
[2] Aug, 2018. Douyu. https://www.douyu.com/. (Aug, 2018).
[3] Aug, 2018. Pathchar. http://www.caida.org/tools/utilities/others/pathchar/. (Aug, 2018).
[4] Aug, 2018. Twitch. https://www.twitch.tv/. (Aug, 2018).
[5] May, 2018. Dash reference page. https://reference.dashif.org/dash.js/nightly/samples/dash-if-reference-player/index.html. (May, 2018).
[6] May, 2018. Python on Android. http://www.qpython.com/. (May, 2018).
[7] Adobe. 2017. Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/hds-dynamic-streaming.html. (2017).
[8] Zahaib Akhtar. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 44–58.
[9] Cisco. Mar, 2018. Visual networking index: Global mobile data traffic forecast update. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.
[10] DASH. 2017. DASH Industry Forum. http://dashif.org/. (2017).
[11] Florin Dobrian. 2011. Understanding the impact of video quality on user engagement. In *ACM SIGCOMM Computer Communication Review*, Vol. 41. ACM, 362–373.
[12] Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. 12 (2011), 2825–2830.
[13] FCC. 2016. Measuring Broadband America. https://www.fcc.gov/general/measuring-broadband-america. (2016).
[14] Wayne A Fuller. 1996. Introduction to Statistical Time Series. *Technometrics* 20, 2 (1996), 211–211.
[15] Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. 2004. Locating Internet bottlenecks: Algorithms, measurements, and implications. In *ACM SIGCOMM Computer Communication Review*, Vol. 34. ACM, 41–54.
[16] Te-Yuan Huang. 2012. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the 2012 Internet Measurement Conference*. ACM, 225–238.
[17] IDLAB. Mar, 2018. 4G/LTE Bandwidth Logs. http://users.ugent.be/~jvdrhoof/dataset-4g/. (Mar, 2018).
[18] Padmanabha Iyer. 2018. Mitigating the Latency-Accuracy Trade-off in Mobile Data Analytics Systems. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 513–528.
[19] Sergei Lebedev. Nov, 2018. hmmlearn. https://github.com/hmmlearn/hmmlearn. (Nov, 2018).
[20] Zhi Li. 2016. Toward a practical perceptual video quality metric. *The Netflix Tech Blog* 6 (2016).
[21] Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* 45, 1-3 (1989), 503–528.
[22] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 197–210.
[23] Microsoft. 2017. Microsoft Smooth Streaming. https://www.iis.net/downloads/microsoft/smooth-streaming. (2017).
[24] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP.. In *USENIX Annual Technical Conference*. 417–429.
[25] Pengpeng Ni, Ragnhild Eg, Alexander Eichhorn, Carsten Griwodz, and Pål Halvorsen. 2011. Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 463–472.
[26] F. Pedregosa and G. Varoquaux. Aug, 2018. Scikit-learn: Multi-layer Perceptron. https://scikit-learn.org/dev/modules/neural_networks_supervised.html. (Aug, 2018).
[27] Hossein Pishro-Nik. 2016. Introduction to probability, statistics, and random processes. (2016).
[28] Sandvine. 2016. Global Internet Phenomena-Latin American & North America. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.
[29] Sivel. May, 2018. speedtest-cli. https://github.com/sivel/speedtest-cli. (May, 2018).
[30] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.
[31] Yi Sun. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 272–285.
[32] Keith Winstein. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks.. In *NSDI*, Vol. 1. 2–3.
[33] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 325–338.
[34] Yasir Zaki. 2015. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 509–522.
[35] Xuan Kelvin Zou. 2015. Can accurate predictions improve video streaming in cellular networks?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 57–62.