VIDEO QUALITY ASSESSMENT FOR ENCRYPTED HTTP ADAPTIVE STREAMING: ATTENTION-BASED HYBRID RNN-HMM MODEL

Shuang Tang *Xiaowei Qin** *Xiaohui Chen* *Guo Wei*

Key Laboratory of Wireless-Optical Communications, Chinese Academy of Sciences,
University of Science and Technology of China, Hefei, P. R. China
Email: `ts1991@mail.ustc.edu.cn`, {`qinxw`*,`cxh`,`wei`}`@ustc.edu.cn`

ABSTRACT

End-to-end encryption challenges mobile network operators to assess the quality of the HTTP Adaptive Streaming (HAS), where the quality assessment is coarse-grained, e.g., detecting if there exist stalling during the whole playback. Targeting on this issue, this paper proposes an attention-based hybrid RNN-HMM model, which integrates HMM with attention mechanism to predict the player states. The model is trained and evaluated based on the download speed and player state sequences of encrypted video sessions collected from YouTube. Experiment results show that the proposed model is able to recognize the player states with $86.53\% \sim 94.35\%$ accuracy, and thus achieves to assess video quality in a fine-grained manner, where how long the stalling lasts and when the stalling occurs can be evaluated effectively from the download speed sequence even when encryption is employed.

Index Terms— Hybrid RNN-HMM, Attention, Encryption, HTTP Adaptive Streaming (HAS), Quality Assessment.

1. INTRODUCTION

As HTTP adaptive streaming (HAS) is widely used in mobile network such as YouTube, Netflix and Youku, it's critical for network operators to monitor the Quality of experience (QoE). Studies point out that the Key Quality Indicators (KQIs) for QoE mainly consists of Initial Buffering Delay (IBD), Stalling Frequency (SF) and Stalling Duration (SD) [1–4]. To this end, QoE of HAS is usually evaluated in terms of these KQIs, and video quality assessment for HAS becomes the main concern.

Generally, video quality assessment approaches can be classified into two categories: Player Model (PM) based and Machine Learning (ML) based approaches. PM based approaches usually rely on video content-relevant information to reconstruct the player buffer [5–9], then KQIs are obtained by comparing the buffered playtime with playing and stalling threshold. By monitoring the player buffer in real time, PM based approaches can evaluate IBD, SF and SD effectively in a fine-grained manner. However, PM based approaches could induce complexity and privacy issues, and end-to-end encryption also invalidates the PM based approaches. ML based approaches depend on a reduced number of content-relevant features [10, 11] or only content-independent network layer features [12–15] for the assessment. However, ML based approaches are coarse-grained, where IBD is usually ignored [10–13, 15] and stalling is usually evaluated in terms of classification problems, e.g., if there exists stalling [10–14].

Targeting on these issues, we propose an Attention-based hybrid Recurrent Neural Network and Hidden Markov Model (A-RNN-HMM), which borrows the idea of sequence-to-sequence (seq2seq) to predict the player states, i.e., initial buffering, playing and stalling, only using network layer information. Different from HMM based speech recognition [16], the transition probabilities for HMM hidden states (the player states) here are time-dependent, which are estimated using a modified attention mechanism dynamically. The contributions can be summarized as follows: (1) we model the playing process with dynamic HMM, where the transition probabilities are calculated at each time step dynamically. (2) We introduce and modify the attention mechanism for estimating the dynamic transition probabilities. As A-HMM-RNN a Download Speed (DS) sequence to player state sequence model, it can evaluate IBD, SF and SD in a fine-grained manner even when HAS is encrypted, where the accuracy for player state recognition reaches 94.35%, and the performance is comparable with the previous methods [14] in terms of the coarse-grained quality assessment.

The remainder of this paper is organized as follows. The next section provides the problem statement. In Section 3, we detail the proposed model. In Section 4, experiment results are discussed. At last, we draw our conclusions.

2. PROBLEM STATEMENT

Denote the DS sequence $\boldsymbol{X} = (\boldsymbol{x}_1, ...\boldsymbol{x}_t, ...\boldsymbol{x}_T)$, where $\boldsymbol{x}_t \in R^n$ is n dimensional real valued vector. Denote the HAS player state sequence $\boldsymbol{l} = (l_1, l_2, ...l_t, ...l_T)$, where $\boldsymbol{l}$ is a sequence

This work was supported in part by the National Science and Technology Major Project of China MIIT under Grant 2017ZX03001011-004 and in part by the Fundamental Research Funds for the Central Universities.

over a finite set $\mathcal{L}$ of HAS player states and $l_t \in \mathcal{L}$. Let $\mathcal{S}$ be a set of training examples, each example in $\mathcal{S}$ consists of a pair of sequences $(\boldsymbol{X}, \boldsymbol{l})$.

Network operators can extract $\boldsymbol{X}$ easily from the core network, however the player state sequence $\boldsymbol{l}$ remains unknown. Thus, **the problem:** Given a training set S, the aim is to train a model M: $\boldsymbol{X} \rightarrow \boldsymbol{l}$ over $\mathcal{S}$ to predict the player states in a way that minimises some task specific error measure.

3. ATTENTION-BASED HYBRID RNN-HMM

As with traditional seq2seq model [17, 18], A-RNN-HMM consists of RNN as encoder and HMM as decoder, and the graphical illustration is presented in Fig. 1. To get a better

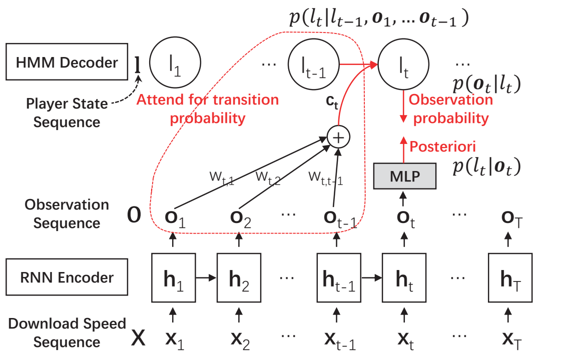

Fig. 1. Architecture of A-RNN-HMM. The model is transmitting to state l_t given the observation sequence $\boldsymbol{O}_1^{t-1}$ and l_{t-1}.

representation, $\boldsymbol{X}$ is first fed to RNN encoder and is encoded to a sequence of vectors $\boldsymbol{O} = (\boldsymbol{o}_1, ...\boldsymbol{o}_t, ...\boldsymbol{o}_T)$, which is later regarded as the observation sequence of HMM.

3.1. Modeling the Playing Process with HMM

During the playing process, HAS player state changes from state l_{t-1} to state l_t at Markov process, and then generates observational vector $\boldsymbol{o}_t$ in the current state l_t. The observation sequence and state sequence are given in Eq. 1

$$\begin{aligned} \boldsymbol{O} &= (\boldsymbol{o}_1, \boldsymbol{o}_2, ...\boldsymbol{o}_t, ...\boldsymbol{o}_T), \quad & \boldsymbol{o}_t \in R^m \\ \boldsymbol{l} &= (l_1, l_2, ...l_t, ...l_T), \quad & l_t \in \mathcal{L}, \end{aligned} \tag{1}$$

where $\boldsymbol{o}_t$ is m dimensional vector at time t and is encoded from $\boldsymbol{X}$, and $\mathcal{L}$ is the player state set with K states as Eq. 2

$$\mathcal{L} = \{L_i, i = 1, 2, ..., K\}. \tag{2}$$

Thus, according to the definition in [19], the playing process can be modeled using a first-order HMM, where $\boldsymbol{O}$ and $\mathcal{L}$ are the observations and hidden states for HMM respectively.

Given the observation sequence $\boldsymbol{O}$, the output of HMM decoder should be the most probable state sequence for $\boldsymbol{O}$:

$$M(\boldsymbol{O}) = \arg\max_{\boldsymbol{l}} p(\boldsymbol{l}/\boldsymbol{O}). \tag{3}$$

Using the terminology of HMM, we define a forward variable α_t as

$$\alpha_t = p(\boldsymbol{l}_1^t, \boldsymbol{O}_1^t), \tag{4}$$

i.e., the joint distribution of the partial observation and state sequence until time t, where $\boldsymbol{l}_1^t = (l_1, l_2, ..., l_t)$ and $\boldsymbol{O}_1^t = (\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_t)$. Then, we can solve for α_t inductively:

$$\begin{aligned} \alpha_t &= p(\boldsymbol{l}_1^t, \boldsymbol{O}_1^t) = p(l_t, \boldsymbol{o}_t, \boldsymbol{l}_1^{t-1}, \boldsymbol{O}_1^{t-1}) \\ &= p(l_t, \boldsymbol{o}_t/\boldsymbol{l}_t^{t-1}, \boldsymbol{O}_1^{t-1}) * p(\boldsymbol{l}_1^{t-1}, \boldsymbol{O}_1^{t-1}) \\ &= p(\boldsymbol{o}_t/l_t, \boldsymbol{l}_t^{t-1}, \boldsymbol{O}_1^{t-1}) * p(l_t/\boldsymbol{l}_1^{t-1}, \boldsymbol{O}_1^{t-1}) * \alpha_{t-1}, \end{aligned} \tag{5}$$

where $p(\boldsymbol{o}_t/l_t, \boldsymbol{l}_t^{t-1}, \boldsymbol{O}_1^{t-1})$ and $p(l_t/\boldsymbol{l}_1^{t-1}, \boldsymbol{O}_1^{t-1})$ are called observation (or emission) and transition probabilities in HMM. Observation probability is assumed dependent on the current state only and so is equivalent to $p(\boldsymbol{o}_t/l_t)$ [16]. Thus, the initialization, recursion and termination for α_t are as follows:

$$\begin{aligned} \alpha_1 &= \pi_{l_1} * p(\boldsymbol{o}_1/l_1), \ t = 1, \\ \alpha_t &= p(\boldsymbol{o}_t/l_t) * p(l_t/l_{t-1}, \boldsymbol{O}_1^{t-1}) * \alpha_{t-1}, \ 1 < t < T, \\ \alpha_T &= p(\boldsymbol{l}, \boldsymbol{O}), \ t = T, \end{aligned} \tag{6}$$

where π_{l_1} is drawn from the initial state distribution $\pi = \{p(l_1 = L_i), L_i \in \mathcal{L}\}$. The probability of the state sequence given the observation sequence can be calculated as Eq. 7

$$p(\boldsymbol{l}/\boldsymbol{O}) = \alpha_T / p(\boldsymbol{O}). \tag{7}$$

It has been the proved that outputs of a standard Multi-Layer Perceptions (MLP) for classification can be interpreted as estimates of a posteriori probabilities [20]. Thus, we use the output of an MLP with a K-unit softmax output layer to estimate the posteriori probabilities, i.e., $p(l_t/\boldsymbol{o}_t) = g(\boldsymbol{o}_t)$, as shown in Fig. 1, and the observation probabilities can be estimated from $\boldsymbol{o}_t$ based on Bayes formula as follows:

$$p(\boldsymbol{o}_t/l_t) = \frac{g(\boldsymbol{o}_t) * p(\boldsymbol{o}_t)}{p(l_t)}. \tag{8}$$

where $p(\boldsymbol{o}_t)$ is the priori probability for $\boldsymbol{o}_t$ and is a constant, and $p(l_t)$ is the priori probability for l_t which can be counted from the dataset directly.

However, it should be noted that according to HAS player model [14], the transition probability to state l_t $p(l_t/l_{t-1}, \boldsymbol{O}_1^{t-1})$ not only depends on the state l_{t-1}, but depends on the previous observation sequence $\boldsymbol{O}_1^{t-1}$, which results in dynamic transition probabilities.

3.2. Attend for the Transition Probability

It is different from the traditional HMM that the transition probabilities $p(l_t/l_{t-1}, \boldsymbol{O}_1^{t-1})$ are dynamic, i.e., time-dependent. We define the time-varying state transition probability distribution as follows:

$$\boldsymbol{A}(t) = \{a_{ij}(t), 1 \leq i, j \leq K\}, \tag{9}$$

where $a_{ij}(t) = p(l_t = L_j / l_{t-1} = L_i, \boldsymbol{O}_1^{t-1})$, and $\sum_{j=1}^{K} a_{ij}(t) = 1$. According to the definition of $a_{ij}(t)$, the HMM decoder can be trained to predict the next state l_t given l_{t-1} and $\boldsymbol{O}_1^{t-1}$.

To this end, we extend the idea of attention in [18], and the attention mechanism for transition probabilities is shown in Fig. 1. Instead of attending on the whole sequence at every time step [18], attention here at time t only attends on the previous observations, i.e., $\boldsymbol{O}_1^{t-1}$, which enables the capability for real time implementations.

Firstly, we calculate a score at time t over each of the previous observations $\boldsymbol{O}_1^{t-1}$ as follows:

$$e_{tk} = f_1(l_{t-1}, \boldsymbol{o}_k), 1 \leq k \leq t-1, \tag{10}$$

where f_1 is an alignment model which scores how well the observation at time k matches the state l_t. Secondly, the scores are normalized using softmax function and we get

$$w_{tk} = \frac{exp(e_{tk})}{\sum_{k=1}^{t-1} exp(e_{tk})}. \tag{11}$$

Then, a context vector is computed as a weighted sum of the previous observations $\boldsymbol{O}_1^{t-1}$ as follows:

$$\boldsymbol{c}_t = \sum_{k=1}^{t-1} w_{tk} \boldsymbol{o}_k. \tag{12}$$

At last, the transition probability at time t can be calculated using a transition probability model f_2 as follows:

$$p(l_t / l_{t-1}, \boldsymbol{O}_1^{t-1}) = f_2(\boldsymbol{c}_t, l_{t-1}), \tag{13}$$

As l_{t-1} and l_t can be any value in the state set $\mathcal{L}$, we get a $K * K$ transition probability matrix $\boldsymbol{A}(t)$ at time t.

During the implementation, we simply parameterize the alignment model f_1 and the transition probability model f_2 as MLPs, which can be jointly trained with all the other components of the proposed A-RNN-HMM model.

3.3. Maximum Likelihood Training

The probability of the state sequence $\boldsymbol{l}$ given the observation sequence $\boldsymbol{O}$ in Eq. 7 can be calculated recursively using Eq. 8 and 13. As the recursion in Eq. 6 may lead to underflows, natural log is applied to the joint distribution α_T.

The aim of maximum likelihood training is to simultaneously maximise the log probabilities of all the observation sequence $\boldsymbol{O}$ and state sequence $\boldsymbol{l}$. In our case, this means minimising the following objective function:

$$C_{ML}(\mathcal{S}, M) = -\sum_{(\boldsymbol{X},\boldsymbol{l}) \in \mathcal{S}} ln[p(\boldsymbol{l}/\boldsymbol{O})] = -\sum_{(\boldsymbol{X},\boldsymbol{l}) \in \mathcal{S}} ln[\alpha_T / p(\boldsymbol{O})]. \tag{14}$$

where $\boldsymbol{O}$ is encoded from $\boldsymbol{X}$, and $p(\boldsymbol{O})$ is a constant during training, thus $C_{ML}(\mathcal{S}, M)$ can be simplified as follows:

$$C_{ML}(\mathcal{S}, M) = -\sum_{(\boldsymbol{X},\boldsymbol{l}) \in \mathcal{S}} ln(\alpha_T). \tag{15}$$

Obtaining the objective function, all the components can be trained jointly using back propagation through time (BPTT).

It should be noted that as l_{t-1} is known from $\boldsymbol{l}$ during the training phase, e.g., L_i, thus instead of calculating the whole transition probability matrix $\boldsymbol{A}(t)$, we only need to calculate the transition probabilities in Eq. 13 conditioned on L_i, which reduces the computational and memory burden.

3.4. Viterbi Decoding

The decoding problem is the same as problem 2 of HMM [16]: given the observation sequence $\boldsymbol{O}$, how to choose a state sequence $\boldsymbol{l}$ which is optimal in some meaningful sense, and can be solved using Viterbi algorithm, i.e., find the best state sequence $\boldsymbol{l}$ to maximize $p(\boldsymbol{l}/\boldsymbol{O})$ which is equivalent to maximizing $p(\boldsymbol{l}, \boldsymbol{O})$. Viterbi algorithm can achieve a real-time or quasi real-time quality assessment by controlling the backtracking depth [16], i.e., $depth = 1$ or $depth > 1$.

According to Section 3.2, the transition probabilities are time-dependent, thus we should calculate $\boldsymbol{A}(t)$ in Eq. 9 at each time step, which results in a $T * K * K$ matrix $\boldsymbol{A}$.

4. EVALUATION

In this section, we evaluate the performance of A-RNN-HMM. We first introduce the dataset, followed by the model setup and experiment results.

4.1. Overview of the Dataset

We develop a data acquisition platform, which mainly consists of mobile phone and VPN proxy. We develop YTMonitor based on Android YouTube API [21] for mobile phone and C++ HTTP server for VPN proxy. Working together, player states including start, playing and stalling, and the corresponding mobile traffic can be collected simultaneously.

We extract DS from the traffic every 0.1 seconds, i.e., $\boldsymbol{DS} = (v_1, v_2, ...)$, which is then cut into DS frames $\boldsymbol{X}$ with a window size $wnd = 10$ and step $step = 5$ as Eq. 16. Three player states including initial buffering (L_1), stalling (L_2) and playing (L_3) are considered, i.e., $\mathcal{L} = \{L_1, L_2, L_3\}$. Obtaining $\boldsymbol{X}$ and $\mathcal{L}$, each frame $\boldsymbol{x_t}$ can be labeled using the player states and we get the state sequence $\boldsymbol{l}$ as Eq. 16.

$$\begin{aligned} \boldsymbol{X} &= (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T), \\ \boldsymbol{x}_t &= (v_{(t-1)*step+1}, ..., v_{(t-1)*step+wnd}), \\ \boldsymbol{l} &= (l_1, ..., l_T), l_t \in \mathcal{L}. \end{aligned} \tag{16}$$

At last, 3,543 samples are collected by crowdsourcing. More details about the platform and dataset can be found in [14].

4.2. Model

A-RNN-HMM consists of RNN encoder and HMM decoder, as shown in Fig. 1. RNN encoder has 3 hidden layers, and each layer consists of 96 long-short term memory (LSTM) [22] units. HMM decoder mainly consist of HMM, MLP for posteriori probabilities g, MLP alignment model f_1 and MLP transition probability model f_2. There are 3 hidden states for HMM model, i.e., L_1, L_2 and L_3 described in Section 4.1, and the state sequence for the samples always starts with L_1, thus the initial state distribution is $\pi = \{1, 0, 0\}$. The number of hidden units for MLP g, f_1 and f_2 are all set to 64. The backtracking depth for Viterbi decoding is set to T, i.e., the length of the whole state sequence.

4.3. Experiment Results

During the evaluation, 5-fold cross-validation is employed and the results are presented in the following.

Each state in state sequence $\boldsymbol{l}$ corresponds to $0.1 * step$ seconds of download speed. Thus, once the state sequence is predicted by the proposed model, video quality can be evaluated in terms of IBD, SF and SD, where IBD and SD can be calculated by multiplying the number of corresponding states with $0.1{*}step$, and SF can be evaluated by the number of discrete periods of stalling. Fig. 2 shows an example of the true (upper part) and predicted (lower part) state sequence in S. IBD and SD are predicted to be 5.0 and 6.0 seconds, SF is 1,

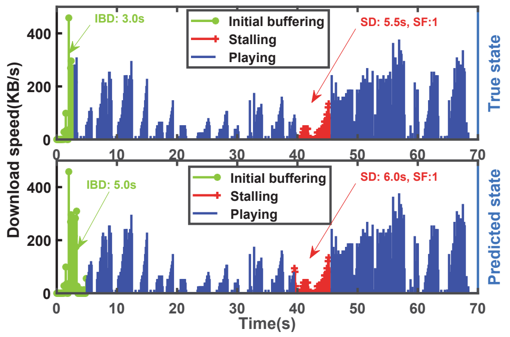

Fig. 2. An example of the true and predicted state sequence.

which are close to the true KQIs. Additionally, the model can also locate when the stalling occurs.

The performance of A-RNN-HMM is compared with RNN-HMM, where RNN-HMM is the same as A-RNN-HMM except for traditional HMM is used for RNN-HMM and the transition probabilities are fixed. The confusion matrix of the player states are presented in TABLE. 1. A-RNN-HMM outperforms RNN-HMM to evaluate the KQIs, where A-RNN-HMM has led to 14.52% and 36.27% recognition error rate reduction for initial buffering L_1 and stalling L_2.

Obtaining the state sequence from A-RNN-HMM, IBD, SD and SF can be evaluated, and the results are compared to the previous work in [14]. For comparison, stalling are calculated from the predicted state sequence to answer: (1) do there exist any stalling, (2) are there multiple stallings, (3) do there exist long stalling during the playback, and for IBD: (4) what is the proportion of the absolute estimation error for IBD within 1 second? The performance is evaluated in terms of True Positive Rate (TPR), False Positive Rate (FPR) and Accuracy for $(1) \sim (3)$, and the results are presented in TABLE 2. Compared with the previous work in [14], although TPR and Accuracy for question $(1) \sim (3)$ and Proportion for question (4) of A-RNN-HMM are a little lower, i.e., $1.1\% \sim 4.4\%$ lower for $(1) \sim (3)$ and 8% lower for (4), A-RNN-HMM is able to evaluate stalling in a more fine-grained manner, i.e., is able to evaluate how long the stalling lasts and locate when the stalling occurs.

Table 1. Player state confusion matrix for A-RNN-HMM and RNN-HMM, (A-RNN-HMM/RNN-HMM).

		Predicted state		
True state		L_1	L_2	L_3
	L_1	**0.9435/0.7983**	0.0382/0.0148	0.0183/0.1869
	L_2	0.0017/0.0076	**0.8653/0.5026**	0.1330/0.4898
	L_3	0.0156/0.0015	0.0815/0.0127	**0.9029/0.9857**

Table 2. Performance comparison with the method proposed in [14], (proposed model/method in [14]).

Question	TPR(%)	FPR(%)	Accuracy(%)
(1)	83.57/88.58	13.33/11.40	85.12/88.60
(2)	89.09/92.24	13.57/7.84	87.76/92.16
(3)	88.89/89.79	13.85/8.12	87.52/91.73
(4)	**Proportion(%)**		72/80

5. CONCLUSION

In this paper, we propose an attention-based hybrid RNN-HMM model to assess HAS quality in a fine-grained manner. We first model the playing process of the playback using HMM with dynamic transition probabilities. Then, based on the output of RNN encoder, observation probability is estimated according to Bayes formula. Thirdly, to estimate the dynamic transition probabilities, we introduce attention mechanism to attend on the observation sequence at each time step. At last, The model is trained and evaluated based on the encrypted videos collected from YouTube. Experiment results show that, A-RNN-HMM contributes to the performance for state recognition, where it is able to recognize player states with $86.53\% \sim 94.35\%$ accuracy. A-RNN-HMM is also able to evaluate KQIs in a fine-grained manner, i.e., evaluate how long the stalling lasts and locate when the stalling occurs from the network layer even when end-to-end encryption is employed, while the performance is comparable with the previous methods in terms of the coarse-grained quality assessment.

Additionally, by controlling the backtracking depth for Viterbi algorithm during the decoding process, the proposed model can be applied to real-time or quasi real-time scenarios.

References

[1] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hobfeld, and Phuoc Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[2] Han Hu, Yuanlong Li, and Yonggang Wen, "Toward rendering-latency reduction for composable web services via priority-based object caching," *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1864–1875, 2018.

[3] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[4] Han Hu, Yongang Wen, and Dusit Niyato, "Spectrum allocation and bitrate adjustment for mobile social video sharing: potential game with online qos learning approach," *IEEE journal on selected areas in communications*, vol. 35, no. 4, pp. 935–948, 2017.

[5] Shuang Tang, XiaoWei Qin, and Guo Wei, "Analysis on the state of mobile http video streaming at the client-side," in *Wireless Communications & Signal Processing (WCSP), 2015 International Conference on*. IEEE, 2015, pp. 1–6.

[6] Pablo Ameigeiras, Juan J Ramos-Munoz, Jorge Navarro-Ortiz, and Juan M Lopez-Soler, "Analysis and modelling of youtube traffic," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 4, pp. 360–377, 2012.

[7] Han Hu, Yonggang Wen, and Shanshan Feng, "Budget-efficient viral video distribution over online social networks: Mining topic-aware influential users," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 3, pp. 759–771, 2018.

[8] Florian Wamser, Pedro Casas, Michael Seufert, Christian Moldovan, Phuoc Tran-Gia, and Tobias Hossfeld, "Modeling the youtube stack: From packets to quality of experience," *Computer Networks*, vol. 109, pp. 211–224, 2016.

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

[10] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*. ACM, 2016, pp. 513–526.

[11] Tingyao Wu, Stefano Petrangeli, Rafael Huysegems, Tom Bostoen, and Filip De Turck, "Network-based video freeze detection and prediction in http adaptive streaming," *Computer Communications*, vol. 99, pp. 37–47, 2017.

[12] Vengatanathan Krishnamoorthi, Niklas Carlsson, Emir Halepovic, and Eric 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*. ACM, 2017, pp. 76–87.

[13] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov, "A machine learning approach to classifying youtube qoe based on encrypted network traffic," *Multimedia tools and applications*, vol. 76, no. 21, pp. 22267–22301, 2017.

[14] Shuang Tang, XiaoWei Qin, and Guo Wei, "Network-based video quality assessment for encrypted http adaptive streaming," *IEEE Access*, vol. 6, pp. 56246–56257, 2018.

[15] Ran Dubin, Ofer Hadar, Amit Dvir, and Ofir Pele, "Video quality representation classification of encrypted http adaptive video streaming.," *KSII Transactions on Internet & Information Systems*, vol. 12, no. 8, pp. 3804–3819, 2018.

[16] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[17] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] Lawrence R Rabiner and Biing-Hwang Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[20] Herve Bourlard and Nelson Morgan, "Continuous speech recognition by connectionist statistical methods," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 893–909, 1993.

[21] YouTube, "Youtube developer documentation," `https://developers.google.com/youtube/`, 2016.

[22] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.