Karma: Adaptive Video Streaming via Causal Sequence Modeling

Bowei Xu
Nanjing University
Nanjing, China
xubowei@smail.nju.edu.cn

Hao Chen
Nanjing University
Nanjing, China
chenhao1210@nju.edu.cn

Zhan Ma
Nanjing University
Nanjing, China
mazhan@nju.edu.cn

ABSTRACT

Optimal adaptive bitrate (ABR) decision depends on a comprehensive characterization of state transitions that involve interrelated modalities over time including environmental observations, returns, and actions. However, state-of-the-art learning-based ABR algorithms solely rely on past observations to decide the next action. This paradigm tends to cause a chain of deviations from optimal action when encountering unfamiliar observations, which consequently undermines the model generalization.

This paper presents Karma, an ABR algorithm that utilizes causal sequence modeling to improve generalization by comprehending the interrelated causality among past observations, returns, and actions and timely refining action when deviation occurs. Unlike direct observation-to-action mapping, Karma recurrently maintains a multi-dimensional time series of observations, returns, and actions as input and employs causal sequence modeling via a decision transformer to determine the next action. In the input sequence, Karma uses the maximum cumulative future quality of experience (QoE) (a.k.a, *QoE-to-go*) as an extended return signal, which is periodically estimated based on current network conditions and playback status. We evaluate Karma through trace-driven simulations and real-world field tests, demonstrating superior performance compared to existing state-of-the-art ABR algorithms, with an average QoE improvement ranging from 10.8% to 18.7% across diverse network conditions. Furthermore, Karma exhibits strong generalization capabilities, showing leading performance under unseen networks in both simulations and real-world tests.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; **Information systems applications**.

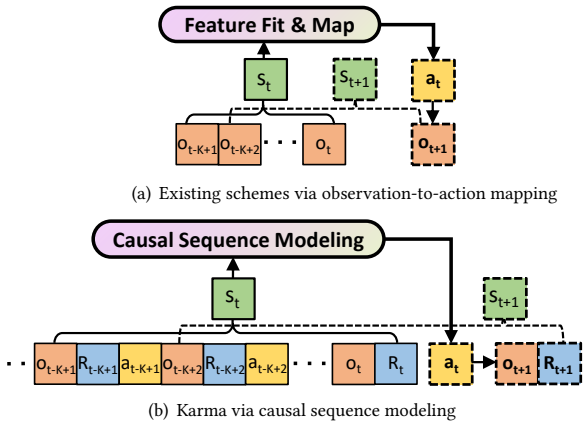
KEYWORDS

Sequence Modeling, Decision Transformer, Adaptive Bit Rate, Video Streaming

ACM Reference Format:
Bowei Xu, Hao Chen, and Zhan Ma. 2023. Karma: Adaptive Video Streaming via Causal Sequence Modeling. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612177

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612177



(a) Existing schemes via observation-to-action mapping

(b) Karma via causal sequence modeling

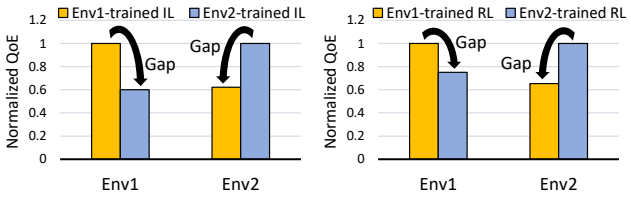
Figure 1: *ABR Decision In Use* **: Karma vs. existing algorithms.**

1 INTRODUCTION

In recent years, a remarkable surge of HTTP-based video traffic [9, 21] has been propelled by the proliferation of video streaming applications. Adaptive Bitrate (ABR) algorithms [6, 12, 28] have emerged as prominent tools and have been employed by content providers to optimize the video quality of streaming services. Typically implemented on the client-side video player, the ABR algorithm dynamically adjusts the video bitrate in response to the underlying network conditions, with the primary objective of maximizing users' quality of experience (QoE).

Early rule-based ABR algorithms rely on fixed control rules for bitrate decisions. However, they usually require careful tuning (e.g., rate-based [24, 42] and buffer-based algorithms [19, 41]) or prior knowledge of network conditions (e.g., MPC [46]), making them incapable of being generalized well in most dynamic networks [18, 42, 49]. Although learning-based ABR approaches (e.g., imitation learning (IL) [17, 35] and reinforcement learning (RL) [22, 31, 32, 43] algorithms) have shown superior performances, they solely stack past observations to decide the next action, as illustrated in Figure 1(a). When familiar observations cannot be encountered in unexperienced environments, these learning-based algorithms tend to fall into a chain of deviations from optimal actions, which ultimately undermines the model generalization [11, 45]. A detailed analysis of existing ABR algorithms and their limitations are revealed in §2.

This paper proposes Karma, a novel ABR decision system that aims to enhance generalization by comprehending the interrelated causality among past observations, returns, and actions and executing timely action refinement when deviation occurs. As shown in Figure 1(b), it recurrently uses a sequence of multi-dimensional elements, including observations, returns, and actions over time as the state input to determine the next action. In Karma, The return



(a) IL-based ABR algorithms (b) RL-based ABR algorithms

Figure 2: Generalization issues of both IL- and RL-based ABR algorithms. Env1 and Env2 have an average bandwidth of 1.3Mbps and 2.0Mbps, respectively. Results are normalized using the model performance measured in the same training environment.

signal is formulated as a "QoE-to-go" representing the *maximum cumulative future QoE* to better assess the current state from a long-term perspective. To thoroughly characterize the cross-time and inter-modality causality in each input sequence, i.e., causal sequence modeling, for better action decisions (e.g., bitrate of the next video chunk), Karma proposes the use of causal decision transformer [8, 44] to fulfill the purpose. We describe the principle and superiority of the proposed causal sequence modeling in §3.

To train Karma, we first construct training samples by generating numerous extended expert ABR trajectories. Each trajectory comprises serial chunk-level tuples, each of which consists of observation, estimated QoE-to-go, and optimal action at each chunk. A pair of observations and optimal action for each chunk is obtained via dynamic programming. It serves as the expert guide for the pursuit of optimality. And the corresponding QoE-to-go is estimated using the current network condition and playback status via a pre-trained estimator. Using these expert ABR trajectories, Karma trains a causal decision transformer via supervised learning by minimizing the cross-entropy between the optimal actions (labeled in trajectories) and Karma predictions. We describe the design and implementations of Karma in §4.

Karma is compared with state-of-the-art ABR algorithms under a wide range of network conditions in both trace-driven simulations and real-world field tests. Our results indicate that Karma achieves an average QoE improvement ranging from 10.8%-18.7% over the best-performing solution in each scenario under consideration (§5.2). It is also worth mentioning that Karma consistently outperforms prevalent learning-based ABR algorithms under networks that have never been experienced in both simulations and real-world tests (§5.3). All of these studies not only report the superior performance of Karma but also reveal its wide generalization. Additional deep dive studies are also conducted to offer in-depth insights into Karma (§5.4).

In general, we summarize our contributions as follows:

- To the best of our knowledge, Karma is the *very first solution* that recurrently maintains a multi-dimensional time series of observations, returns, and actions and employs causal sequence modeling via a decision transformer to determine the next action in the ABR system, which significantly ameliorates the weakness in generalization for existing learning-based approaches that solely rely on the observation-to-action mapping;
- We utilize a maximum cumulative future QoE (i.e., QoE-to-go) as an effective return signal to reflect the long-term assessment of the current state and devise a learning-based QoE-to-go estimator in Karma, which is also different from the instant returns used in existing approaches;
- Extensive experimental results in both trace-driven simulations and real-world field tests demonstrate the proposed Karma's remarkable generalization and superior performance across a broad set of network conditions.

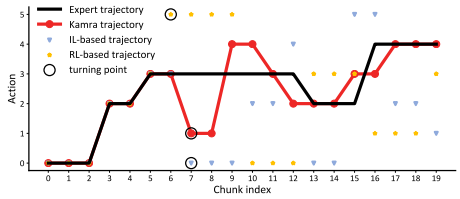


Figure 3: Comparing the action trajectories using the expert policy, the IL-based model, the RL-based model, and our proposed Karma.

2 BACKGROUND AND MOTIVATION

HTTP-based adaptive streaming is a dominant tool used to deliver video content across the entire Internet today. Relevant techniques have also been approved as the international standard like DASH [1] to assure service interoperability across various heterogeneous users. As the underlying network conditions usually fluctuate unexpectedly, ABR algorithms are usually devised to sustain uncompromised QoE.

Existing ABR algorithms can be grouped into two classes: rule-based algorithms and learning-based algorithms. As extensively analyzed in [32], rule-based algorithms cannot easily generalize themselves to different networks because deliberate tuning of parameters or a precise understanding of system dynamics is required. Unfortunately, unsatisfactory generalization is also troubling recent learning-based ABR algorithms. Xia et al. [45] reported that existing learning-based ABR algorithms tend to better adapt to the same environment used in training, and significant degradation is observed when they operate in a new environment. The same observation is also reproduced in our preliminary experiment. We prepare two separate environments (i.e., Env1 and Env2) by dividing a part of network traces from Norway 3G/HSDPA mobile dataset [39] into two corpora with different average bandwidths. Representative IL- and RL-based ABR models are respectively trained in Env1 and Env2 and then tested in both of them as well. As shown in Figure 2, significant performance degradation is observed when testing the model in a different environment for both IL- and RL-based approaches. More details will be provided in §5.

To investigate the reason for this generalization issue, we further analyze the action trajectories generated using the expert policy, the IL-based ABR model (Comyco), the RL-based ABR model (Pensieve), and the causal sequence modeling based ABR model (our proposed Karma) in the inference. Figure 3 shows an example of these trajectories recorded in the same unexperienced networks. We find that there is a turning point in the trajectory (at chunk 6 or 7 in Figure 3) that IL- and RL-based models start to take an action that deviates from the optimal action by the expert policy. We speculate that it is because an unfamiliar observation is encountered when deciding on an action. More seriously, a bad chain reaction is observed for existing IL- and RL-based algorithms: a deviated

action (from the expert one) is likely to transit the environment to a state with more unfamiliar observations, which in turn makes them take more deviated (sub-optimal) actions. As these algorithms only use past observations as input and utilize a typical neural network to fit a direct observations-to-action map for decision-making, they cannot break the vicious circles themselves, ultimately leading to significant performance degradation in the new environment. This may be a major cause of poor generalization for existing learning-based algorithms.

Therefore, a new ABR algorithm that can effectively control the deviation between its actions and the optimal trajectory is highly desired to achieve better generalization. Considering that an ABR task is essentially a causal sequential decision problem, we have an intuition that this new ABR algorithm should be able to capture the interrelated causality among observations, returns, and actions. For example, how past actions affect observations and returns, and how received observations and returns affect the decision of the next action. By taking the observations and returns as feedback signals, the ABR algorithm can analyze past actions and subsequently refine its next action for better feedback to catch up with the optimal action trajectory. As shown in Figure 3, our proposed Karma equipped with causal sequence modeling can make timely action refinement after the occurrence of action deviation at chunk 7.

3 ABR ALGORITHM VIA SEQUENCE MODELING

To this aim, we propose Karma, which distinguishes itself from existing ABR algorithms via causal sequence modeling on past observations, returns, and actions to make decisions. This is achieved in two correlated ways:

1) Providing a multi-dimensional causal sequence as input. We dynamically maintain a multi-dimensional causal sequence of past observations, returns, and actions to preserve the cross-time and inter-modality causality in state transitions. Karma uses this sequence as the input in both the training and inference stages. Specifically, the observation is a multidimensional signal, consisting of network throughput, buffer occupancy, and video information. The action signal indicates the bitrate to use for the next chunk. The return signal, i.e., QoE-to-go, is defined as the maximum cumulative future QoE that can be attained to download all remaining chunks. Instead of a single-step instant QoE, such QoE-to-go allows Karma to comprehensively to better assess the current state from a long-term perspective. For example, the action with a high instant QoE may force the agent to take more conservative actions for subsequent chunks, which cannot be deemed a good one as the overall QoE for the entire video streaming session is degraded.

2) Applying a causal decision transformer for sequence modeling. Empirical evidence suggests that a sequence modeling approach can model widely distributed behaviors, leading to a better generalization in the sequential-decision problems [20, 38]. In Karma, a causal decision transformer is utilized to accurately characterize the state transition by modeling long-term dependencies among all three modalities, which inherently reside in sequential-decision ABR tasks. Compared to normal neural networks, this transformer architecture derives considerable advantages for effective sequence modeling from both the positional encoding and causal self-attention

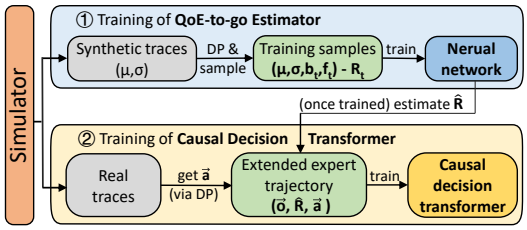


Figure 4: The logical diagram of the training pipeline used by Karma.

mechanism. The positional encoding provides unique position information for each input token, facilitating the transformer to discriminate between tokens at different positions. The causal self-attention mechanism empowers the transformer to concurrently attend to different tokens in the input sequence, capturing their causal relationship while ensuring that only the previous input information is utilized to predict the next chunk's action in line with the principle of causality. In this way, Karma learns to choose actions based on a comprehensive understanding of the causality among all modalities rather than just observations.

4 DESIGN OF KARMA

In this section, we describe the design and implementation of Karma. We first introduce the basic training algorithm of Karma. Then, we describe how to apply the trained Karma for adaptive bitrate selections during a video streaming session. Finally, we provide the implementation details of Karma.

4.1 Training Karma

To train Karma, we must provide a set of extended expert trajectories as training samples, each containing a tuple of observations, corresponding QoE-to-go, and optimal action. To this aim, we introduce a simulator, as widely-used in [30, 32], to simulate the video streaming environment faithfully, which largely accelerates the process of producing extended expert trajectories. As the accurate QoE-to-go is unavailable until the video streaming session ends, we first train a QoE-to-go estimator under synthetic network traces to generate the estimated QoE-to-go modality of extended expert trajectory recurrently based on current observations. Then we use the dynamic programming (DP) algorithm as an ABR method under real traces to generate the observation and corresponding action modalities of extended expert trajectory. Finally, a causal decision transformer is trained based on these extended expert trajectories. The logical diagram of the training pipeline is illustrated in Figure 4.

Training QoE-to-go estimator: As stated in the definition, the QoE-to-go at chunk t (denoted as R_t) can be formulated as $R_t = max(\lambda \sum_{t'=t}^{T} QoE(t'))$. Once given the current buffer size b_t and the chunk number T, R_t is only determined by the network condition when downloading the remaining chunks. Based on the observation made by previous works [4, 48] that end-to-end throughput is piecewise stationary on a short-term time scale, we assume the network condition in the near future keeps the same as that currently. In this paper, we use the mean μ_t and the variance σ_t of throughput values to characterize the network condition at time t by following prior works [2, 42]. Hence, we suggest estimating the QoE-to-go

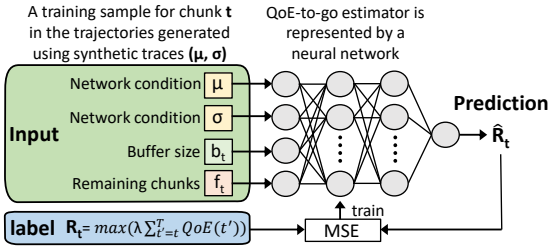


Figure 5: The training algorithm for QoE-to-go estimator.

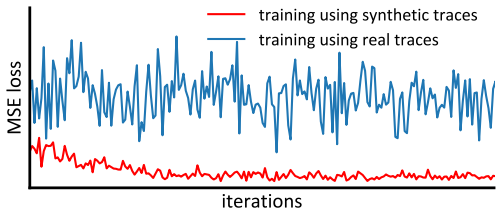


Figure 6: Comparing the training loss of QoE-to-go estimator over time using synthetic traces with that using real traces.

based on the current observations, including the playback status and the network condition (e.g., the mean μ_t and the variance σ_t of recent throughput measurements) as shown in Figure 5, and update the estimation chunk-by-chunk to adapt to possible new network conditions in the future.

To train the QoE-to-go estimator, we first produce a set of stationary synthetic traces, in which the throughput sample is generated following a Gaussian distribution with a predetermined mean μ and variance σ. This design ensures that the network condition remains relatively constant for all chunk downloads and guarantees the effectiveness of training samples for learning the mapping of current observations to QoE-to-go. It is worth noting that real traces with non-stationary bandwidth distributions do not meet the requirement of similar network conditions currently and in the future, so they are not suitable for the use of constructing training samples. Figure 6 illustrates that the training of the QoE-to-go estimator can converge more effectively using synthetic traces than that using real traces.

Specifically, we generate training samples for the QoE-to-go estimator using these synthetic traces. Using DP as the ABR algorithm, the maximum cumulative QoE for all remaining chunks can be achieved, which is actually the truth value of QoE-to-go. As shown in Figure 5, we take a set of current observations, including the network condition μ and σ (i.e., the settings for generating a synthetic trace), buffer size b_t, and the percentage of remaining chunks f_t as the input, and use corresponding QoE-to-go R_t as the label. Karma represents its QoE-to-go estimator as a two-layer fully connected network, and we train it by minimizing the mean squared error (MSE) loss between the output and the label.

Training causal decision transformer: To ensure that Karma can gain experiences from the real environment, we use a broad set of network traces collected in the real world to train the causal decision transformer. For each observation $\vec{o}_t$, Karma uses network throughput measurements for the past L chunks to compute μ_t and σ_t. Given the current observations of μ_t, σ_t, b_t and f_t, an estimated QoE-to-go $\hat{R}_t$ can be output by the well-trained QoE-to-go estimator. And an optimal action $\vec{a}_t$ is produced by using the dynamic programming algorithm. Once $\vec{a}_t$ is executed, Karma gets the next observation $\vec{o}_{t+1}$ and generates $\hat{R}_{t+1}$, which initiates a new round of action decision. In this way, the observation, estimated QoE-to-go, and optimal action together constitute an extended expert trajectory. For a video with a total of T chunks, an extended expert trajectory can be expressed as a 3-modality sequence with $3T$ tokens in total:

$$\tau = \left((\vec{o}_1, \hat{R}_1, \vec{a}_1), (\vec{o}_2, \hat{R}_2, \vec{a}_2), \ldots, (\vec{o}_T, \hat{R}_T, \vec{a}_T)\right), \tag{1}$$

where $\vec{o}_t$, $\hat{R}_t$, and $\vec{a}_t$ respectively represent a set of observations, the estimated QoE-to-go, and the action for chunk t.

- Observation: At chunk t, the agent gets an observation vector $\vec{o}_t = (b_t, c_t, d_t, \vec{e}_t, f_t)$ from the environment. b_t is the buffer size; c_t is the network throughput measurement; d_t is the download time; $\vec{e}_t$ is a vector of all available sizes for the next video chunk; f_t is the percentage of remaining chunks.
- Estimated QoE-to-go: $\hat{R}_t$, the estimation of maximum cumulative QoE for the remaining chunk downloads, is used in Karma to represent the return signal, which is generated from the trained QoE-to-go estimator.
- Action: $\vec{a}_t$ is a vector that indicates the probability distribution to select one from several available discrete bitrates for chunk t.

Then the causal decision transformer [8], a generative pre-trained transformer (GPT) [13, 37] model with causal self-attention masking, is introduced to execute sequence modeling and predict the bitrate for the next chunk. A training sample $(o, \hat{R}, a)$ is constructed by sampling a segment for K consecutive tuple of observation, estimated QoE-to-go, and action from an extended expert trajectory. During each training epoch, a mini-batch of training samples is input to the transformer. Initially, we acquire token embeddings for timesteps and three modalities using a linear layer [3]. Then, we perform positional encoding by embedding timestep to each token. In contrast to the conventional positional encoding scheme, where a single token corresponds to one timestep, Karma maps one timestep to three distinct tokens. These embeddings are then merged into interleaved tokens, which are further processed by multi-encoders and multi-decoders of the transformer to derive the hidden states. These encoders and decoders, equipped with causal self-attention masking, ensure that the current prediction in ABR tasks is not influenced by any information beyond the current timestep, i.e., an action a_k can only be predicted using the $3k-1$ tokens before it in the sequence. Finally, the actions are predicted via another linear-layer decoder. The training loss comes from the cross-entropy of predicted actions and optimal actions, and the parameters of the Transformer are updated using the gradient descent algorithm. The training algorithm of the causal decision transformer is summarized in Algorithm 1.

4.2 Inference

Bitrate decision: To select a bitrate for video chunk t, Karma first utilizes a multi-dimensional sequence of observations, estimated QoE-to-go, and actions for the past K chunks as input, which totally possess $3K-1$ tokens (not including a_t token which is to be predicted). These tokens are then combined with their timesteps to

Algorithm 1:

// $o, \hat{R}, a, t$: observations, estimated QoE-to-go, actions, and timesteps
// $embed_t$:embedding for positional encoding
// $embed_o, embed_{\hat{R}}, embed_a$:linear embedding layers
// GPT: a transformer architecture with causal self-attention masking
// $pred_a$:linear action prediction layer

Create an initialized GPT
while *not reach max training episode* **do**
  Randomly pick a minibatch of $sample(o, \hat{R}, a, t)$ from extended expert trajectories
  for *sample* $(o, \hat{R}, a, t)$ *in minibatch* **do**
    // embeddings and positional encoding
    // per-timestep with 3 tokens
    $pos_{emb} = embed_t(t)$
    $o_{emb} = embed_o(o) + pos_{emb}$
    $\hat{R}_{emb} = embed_{\hat{R}}(\hat{R}) + pos_{emb}$
    $a_{emb} = embed_a(a) + pos_{emb}$
    $input_{emb}$ = stack($\hat{R}_{emb}$, o_{emb}, a_{emb})
    // get hidden states and predict next action
    hidden_states = GPT($input_{emb}$)
    a_{hidden} = unstack(hidden_states).actions
    $a_{preds} = pred_a(a_{hidden})$
    // Cross-Entropy for discrete actions
    loss = CrossEntrophy(a_{preds}, a)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
  end
end

execute positional encoding. Finally, Karma passes the positional-encoded tokens into the trained causal decision transformer to predict the bitrate $\vec{a}_t$ of the video chunk t. Figure 7 illustrates the logical diagram of the inference pipeline used by Karma.

The update of input sequence: Once the video player executes $\vec{a}_t$, the environment transits to a new state with the observation $\vec{o}_{t+1}$ and an estimated QoE-to-go $\hat{R}_{t+1}$. Similar to training, $\hat{R}_{t+1}$ is generated via the trained QoE-to-go estimator using b_{t+1} and f_{t+1} from $\vec{o}_{t+1}$, as well as mean μ_{t+1} and variance σ_{t+1} calculated using the past L network throughput measurements. Obviously, the estimated QoE-to-go is updated chunk-by-chunk recurrently, which can improve estimation accuracy when the network condition or playback status changes. Subsequently, $\vec{a}_t$, $\vec{o}_{t+1}$, and $\hat{R}_{t+1}$ are all added into the multi-dimensional sequence, and the oldest $\vec{o}_{t-K+1}$, $\hat{R}_{t-K+1}$, and $\vec{a}_{t-K+1}$ are removed from the sequence. This update process continues until the end of the video.

4.3 Implementation

In the implementation of Karma, we use the past $L = 4$ chunks to characterize the current network condition for the QoE-to-go estimator and use a multi-dimensional sequence for the past $K =$ 4 chunks as the input of the transformer. The impact of L and K on Karma's performance will be discussed in §5.4. Similar to Oboe[2, 15], we generate synthetic traces with mean μ ranging from 0.5Mbps to 6Mbps in increments of 0.1Mbps and variance σ ranging from 0 to 3 in increments of 0.1, to produce training samples for QoE-to-go estimator. The scale factor λ is empirically set to 0.01. The transformer uses a neural network structure of 3 hidden layers, one attention head, and a 128-dimensional embedding operation. During training, we use an initial learning rate of 0.001 and then dynamically adjust it using the cosine decay manner. We update the parameters using the AdamW optimizer [27]. Relu is used as the activation function, and the dropout is set to 0.1. The mini-batch size is set to 128. We use PyTorch to implement the QoE-to-go estimator and the Transformer architectures. After Karma generates an ABR algorithm, it is necessary to apply it to real-world video streaming sessions. For this purpose, we deploy Karma on a separate ABR server for real-world applications, which is implemented using the Python *BaseHTTPServer*.

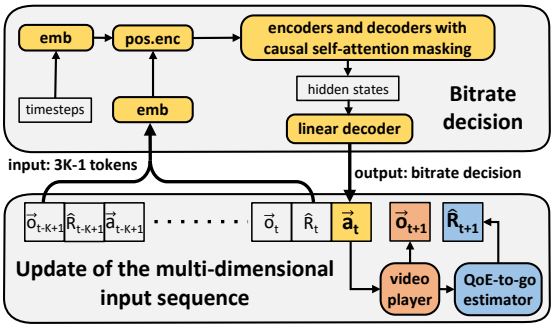


Figure 7: The logical diagram of the inference pipeline used by Karma.

5 EVALUATION

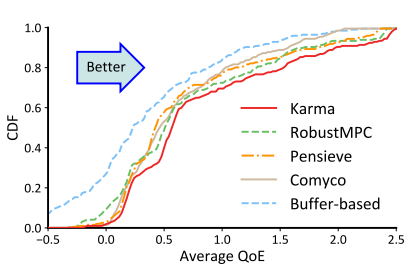
5.1 Methodology

Network traces: To evaluate Karma on realistic network conditions, we use real network traces from several popular public datasets, including FCC's broadband dataset [10], and 3G/HSDPA mobile dataset [39] collected in Norway. We use 70% of these traces as a training set and the remaining 30% as a test set. Another Oboe dataset [2], which collects traces from wired, WiFi, and cellular network connections, is introduced only for validation. Each trace is meticulously filtered to satisfy the requirement of average throughput below 6Mbps and minimum throughput above 0.2Mbps. Moreover, all traces were formatted to be compatible with the Mahimahi network simulation tool.

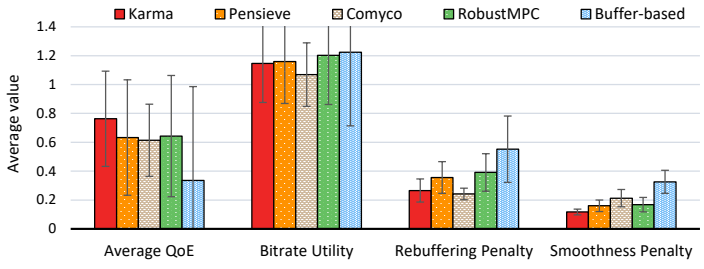
QoE metric: Due to the preferences of different users [26, 33, 34, 36], we use a general form of linear QoE metric for video chunk t, which is defined as

$$QoE(t) = q(r_t) - \eta T_t - \gamma \left| q(r_t) - q(r_{t-1}) \right| \quad (2)$$

where r_t represents the selected bitrate for chunk t, $q(r_t)$ maps the bitrate to the quality perceived by a user, and T_t represents the rebuffering time. η and γ are penalty factors for rebuffering and quality smoothness loss. In accordance with recent researches [25, 32, 47], we choose to set $q(r_t) = 0.001 r_t$ (r_t in kbps), $\eta = 4.3$ and $\gamma = 1.0$.

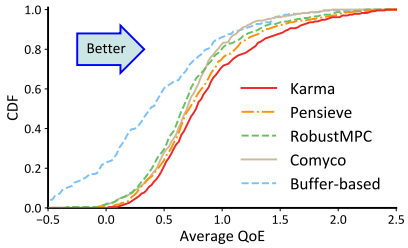


(a) The CDF distributions of general QoE

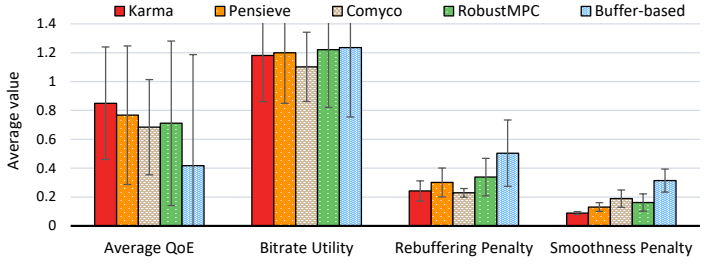


(b) The average value of general QoE metric and its individual components

Figure 8: Comparing Karma with existing ABR algorithms on FCC broadband networks. The mean and variance of throughput in the FCC dataset are 1.30 and 0.99 respectively.



(a) The CDF distributions of general QoE



(b) The average values of general QoE and its individual components

Figure 9: Comparing Karma with existing ABR algorithms on Norway 3G/HSDPA networks. The mean and variance of throughput in the Norway dataset are 1.56 and 0.97 respectively.

ABR Baselines:

- Pensieve [32]: firstly introduces RL-based techniques into adaptive video streaming applications. Its return modality is only used in the training stage. In this paper, we use the pre-trained Pensieve model provided by the authors.
- Comyco [17]: a typical IL-based ABR algorithm. It learns the control policy for bitrate adaptation by imitating the observation-to-action behavior of an expert. In this paper, we retrain Comyco using the QoE metrics defined in Equation (2).
- RobustMPC [46]: MPC solves a QoE maximization problem over a horizon of several future chunks based on buffer occupancy and throughput prediction to select bitrates. Furthermore, RobustMPC introduces a normalized error between the predicted and actual throughput to make the algorithm more robust.
- Buffer-based (BB) [19, 41]: dynamically selects the next bitrate according to the current buffer occupancy.

Experimental setup: The experimental setup is consistent with Pensieve [32]. Here we give a brief description. The "Envivio- Dash3" video was used for evaluation, which was divided into 48 chunks. Each chunk represents approximately 4 seconds of video playback time. The video provides six discrete bitrates as {300, 750, 1200, 1850, 2850, 4300} kbps, which pertain to video resolutions in {240, 360, 480, 720, 1080, 1440}p. A playback buffer capacity of 60 seconds was configured for the player. Please refer to Pensieve [32] for details.

5.2 Karma versus Existing ABR Algorithms

To evaluate Karma, we compare it with different types of state-of-the-art ABR algorithms. Figure 8 and Figure 9 depict the results in the form of cumulative distribution function (CDF) distributions and average values of general QoE on the FCC broadband dataset and the Norway 3G/HSDPA dataset respectively. The average values of individual components in the general QoE definition (Equation (2)) are also provided, including bitrate utility, rebuffering penalty, and smoothness penalty.

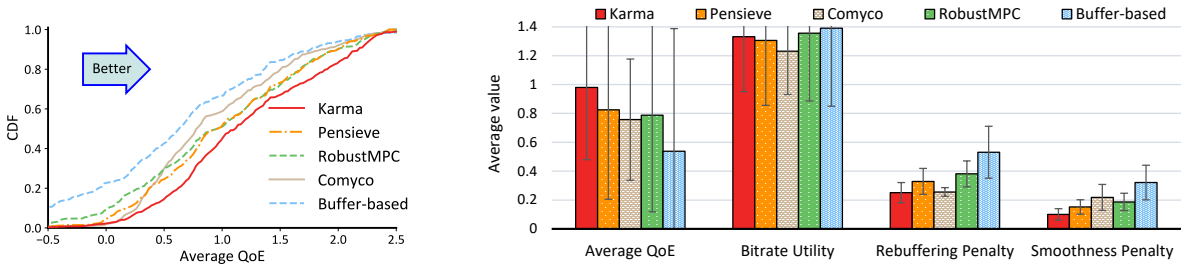
In general, Karma has shown to be a superior ABR algorithm compared to the existing solutions, with an average QoE improvement ranging from 10.8% (Pensieve on Norway networks) to 18.7% (RobustMPC on FCC networks). As evidenced by the CDF distribution, Karma is able to adapt to various network conditions, which is a strong demonstration of robustness. Karma does not lead in every underlying metric when compared to other ABR algorithms. For example, Pensieve and RobustMPC usually exhibit a better average bitrate utility. However, Karma exhibits better control of rebuffering events and frequent bitrate switching, which helps it stand out from all schemes.

The closest competing schemes are Pensieve and RobustMPC. Comyco employs a policy that tends to choose a lower bitrate and perform frequent bitrate switches to ensure smooth video playback. This conservative policy only helps to reduce rebuffering events but does not necessarily lead to a satisfactory overall QoE. BB falls behind other schemes significantly because it makes bitrate decisions only based on a fixed rule of current buffer occupancy, making it struggle to adapt to different network conditions.

5.3 Generalization

To evaluate the generalization of Karma, we first conduct several experiments across a wide range of network conditions in both simulation and the real world and across multiple video properties. Second, we retrain Karma with an expert policy that is sub-optimal but more easily available and evaluate its effectiveness.

Unexperienced network traces: We compare Karma with existing ABR algorithms under the network traces in the Oboe dataset, which has never been experienced for all learning-based ABR algorithms in the training stage. The Oboe dataset has a different distribution of network conditions, bringing more challenges for



(a) The CDF distributions of general QoE

(b) The average values of general QoE and its individual components

Figure 10: Comparing Karma with existing ABR algorithms on unexperienced Oboe network traces. The mean and variance of throughput in the Oboe dataset are 1.79 and 1.36 respectively.

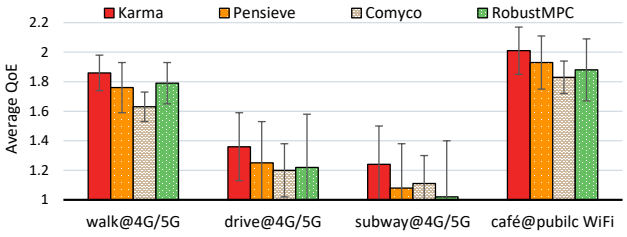


Figure 11: Comparing Karma with existing ABR algorithms in the wild. Results are collected on the 4G/5G cellular and public WiFi networks in the scenes of walking, driving, subway, and stationary café.

these ABR algorithms to retain their performance. As shown in Figure 10, we find that Karma achieves the best generalization under the unexperienced network conditions, with a stable improvement of 19.2% over the second-best Pensieve.

Real-world field tests: We conduct several experiments in the wild to evaluate Karma and several state-of-the-art ABR algorithms (Pensieve, Comyco, and RobustMPC) on 4G/5G cellular and public WiFi networks in the scenes of walking, driving, subway, and stationary café. The evaluation is carried out on a client running on Ubuntu 18.04 who performs actual video requests and downloads from a video server hosted on a node of a 3rd-party Internet cloud. In these experiments, we load the test video ten times in random order using each ABR algorithm. As depicted in Figure 11, Karma always achieves the best performance in the form of average QoE on different networks and in different scenes. Over relatively stationary network connections, e.g., walking@4G/5G cellular network and café@public WiFi network, Karma has a slight improvement of about 4% compared with the second-place scheme (RobustMPC in walking@4G/5G and Pensieve in café@public WiFi). While over more dynamic network connections, e.g., driving@4G/5G cellular network and subway@4G/5G cellular network, Karma significantly outperforms other ABR algorithms by 7.1%-11.7%. The network dynamics are mainly caused by high mobility and frequent handovers [29, 40]. Interestingly, we also find that existing ABR algorithms show inconsistent performances in different scenes. For example, Pensieve, RobustMPC, and Comyco respectively rank second in the scene of driving@4G/5G (and café@public WiFi), walking@4G/5G, and subway@4G/5G. It reveals that Karma can generalize well to different networks while existing ABR algorithms cannot.

Multiple videos: We also evaluate Karma on other videos different from the one in training, which are very common in real life. For each test, a video was generated synthetically with different properties including bitrate ladders, number of chunks, and chunk size. Specifically, the number of available bitrates was randomly selected from $[4, 8]$, and the number of chunks was randomly selected from $[30, 60]$. Other properties, including the chunk size, were determined using the method presented in Pensieve [32]. For the video not with 6 bitrate ladders, Kamra just maps its predicted action down to an available bitrate without retraining. The Multi-video Pensieve, a retrained ABR model on the above-mentioned synthetic videos, is used for comparison. Figure 12 shows that Karma outperforms Multi-video Pensieve with an improvement of 5.5%. This result demonstrates that Karma can generalize well across different video properties.

Training using sub-optimal expert trajectories: Considering that the absolute optimal expert trajectories are sometimes unavailable (needing known network traces) or costly (high complexity in dynamic programming) to acquire, we seek to validate the effectiveness of Karma when only a feasible sub-optimal expert policy can be provided. Specifically, we use Pensieve to generate a series of sub-optimal expert trajectories (compared to that generated via dynamic programming) for training Karma as well as IL-based Comyco. Figure 13 shows that Karma outperforms Comyco significantly with the improvements on average QoE of 28.8%. It implies that Karma can generalize effectively even if the expert policy is sub-optimal. Notably, Karma even exceeds the expert trajectories themselves (i.e., Pensieve's performance) with a slight improvement of 2.7%. An explanation is that by generating reasonable QoE-to-go tokens using the estimator, Karma can execute timely action refinement to get close to or even exceed the desired target. Thus, even if trained with the guidance of a sub-optimal policy like Pensieve, this inherent mechanism helps Karma partly alleviate the deviation from optimal action in the inference, which existing ABR algorithms cannot achieve.

5.4 Karma Deep Dive

In this section, we set up a series of experiments to gain a thorough understanding of Karma. Specifically, we investigate the optimal input sequence length (i.e., K) for the transformer, the optimal window size of past chunks (i.e., L) used to generate network condition features for the QoE-to-go estimator and the most effective form of the return signal in Karma.

The input sequence length: We have tested Karma's performances using different K. As shown in Table 1, the best performance is achieved when K is 4. However, Karma does not deteriorate significantly when K varies within a reasonable range. For instance,

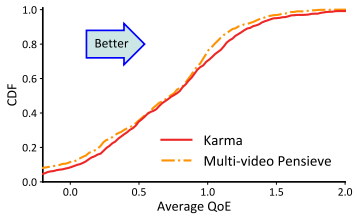


Figure 12: Comparing Karma with multi-video Pensieve across multiple video properties.

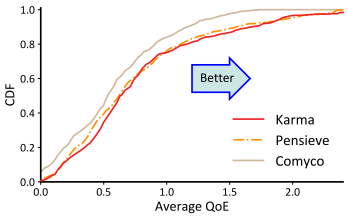


Figure 13: Comparing Karma with Comyco when trained using Pensieve as the expert policy.

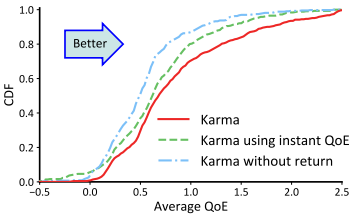


Figure 14: Comparing Karma with its variants using different forms of the return signal.

the performance drop is limited to 5% when K is either 3 or 5. But if the input sequence is too short (e.g., $K = 1$), Karma fails to capture the effective causality, leading to a severe performance collapse. Meanwhile, a higher value of K (e.g., $K = 8$) also does not necessarily guarantee a performance improvement, implying that excessive, unnecessary historical information may interfere with Karma's decision.

The window size for QoE-to-go estimator: We also study the effect of using varying L for the QoE-to-go estimator on the performance of Karma. Results are listed in Table 2. The best performance is attained for Karma when L is set to 4. Because the network condition usually fluctuates over time, it's insufficient only to use a single throughput sample (e.g., $L = 1$) to represent the current network condition. On the contrary, a throughput series that contains information from a long time ago (e.g., $L = 8$) also fails to accurately represent the current network condition, which even results in a worse performance of Karma.

The form of return signal: To verify how QoE-to-go benefits Karma, we develop two variants of Karma for comparison: one using instant QoE as the return signal and the other learning from the expert trajectories without any return signal. The results are illustrated in Figure 14. We find that the average QoE for Karma is 25.4% higher than that using instant QoE. We conjecture that the instant QoE can hardly establish proper causality among past modalities for sequence modeling, finally preventing Karma from learning appropriate policy in a causal sequence modeling way. Similarly, the scheme that ignores the return signal seems like a kind of simple behavior cloning and falls behind Karma with a vast gap of 42.7%.

Karma Overhead: We train and test Karma on a 12-core, AMD R5-4600H 3.00Hz CPU. Because of the direct expert policy, Karma can be proficiently trained within 2 hours (similar to Comyco and faster than Pensieve). The model size is 5.3 MB, and the decision-making time is in milliseconds. In short, we believe that Karma does not incur significant computational overhead and is highly feasible for practical implementation.

Table 1: Different settings of K on Karma's performance.

K	Average QoE
1	0.564 ± 0.095
3	0.762 ± 0.031
4	**0.793 ± 0.021**
5	0.774 ± 0.028
8	0.690 ± 0.036

Table 2: Different settings of L on Karma's performance.

L	Average QoE
1	0.687 ± 0.144
3	0.774 ± 0.023
4	**0.793 ± 0.021**
5	0.744 ± 0.042
8	0.511 ± 0.109

6 RELATED WORKS

ABR algorithms: Most early ABR algorithms develop fixed rules based on environmental observations. For example, buffer-based (BB) [19, 41] and rate-based (RB) [24, 42] choose bitrate based on buffer occupancy and estimated network throughput, respectively. MPC[46], as the state-of-the-art rule-based approach, uses both buffer occupancy information and estimated throughput to select bitrate by solving a QoE maximization problem over a horizon of several future chunks. Emerging learning-based ABR algorithms can be classified into IL-based algorithms [17] and RL-based algorithms [14, 16, 32]. They benefit from the excellent fitting ability of neural networks and can learn a better ABR policy from expert trajectories or through exploration. However, they typically rely on a direct observations-to-action map for decision-making and tend to generalize poorly in a new environment with unfamiliar observations.

Sequence modeling in decision problem: Recently, a new technology paradigm that applies sequence modeling using the transformer [44] to solve decision-making problems appears. It stands out from traditional learning approaches due to its remarkable capability of modeling long sequences. For instance, in some essential RL decision scenarios, such as Gym [7] and Atari [5], Decision Transformer (DT) [8] meets or even exceeds the best temporal difference learning-based traditional RL. Similarly, Trajectory Transformer (TT) [23] models the distribution over trajectories as a planning algorithm. However, as far as we know, there is no research that directly applies causal sequence modeling to the ABR task.

7 CONCLUSION

This paper proposed Karma, an ABR algorithm that applied causal sequence modeling for ABR optimization. Karma first maintained a multi-dimensional causal sequence of past observations, QoE-to-go, and actions as input and then applied a causal decision transformer for causal sequence modeling and final bitrate decision. Experimentally, Karma outperformed existing fixed rule-based and learning-based ABR algorithms, with an average QoE improvement of 10.8%-18.7%. Karma also proved its ability to generalize well across various networks and multiple video properties in both simulations and real-world field tests.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (62101241, 62231002) and the Jiangsu Provincial Double-Innovation Doctor Program (JSSCBS20210001). *(Corresponding Author: Hao Chen.)*

REFERENCES

[1] Akamai. 2016. dash.js. https://github.com/Dash-Industry-Forum/dash.js/.. (2016).

[2] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 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*. 44–58.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[4] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, and Randy H Katz. 1997. Analyzing stability in wide-area network performance. In *Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. 2–12.

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[6] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[9] Visual Network Index Cisco. 2017. Cisco visual networking index: forecast and methodology 2016–2021. *CISCO White paper* (2017).

[10] Ronald H Coase. 2013. The federal communications commission. *The Journal of Law and Economics* 56, 4 (2013), 879–915.

[11] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. 2019. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems* 32 (2019).

[12] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. *ACM SIGCOMM computer communication review* 41, 4 (2011), 362–373.

[13] Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512* (2019).

[14] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, and Andrea Zanella. 2017. D-DASH: A deep Q-learning framework for DASH video streaming. *IEEE Transactions on Cognitive Communications and Networking* 3, 4 (2017), 703–718.

[15] Tianchi Huang. 2022. oboe-reproduce. https://github.com/godka/oboe-reproduce. (2022).

[16] Tianchi Huang, Xin Yao, Chenglei Wu, Rui-Xiao Zhang, and Lifeng Sun. 2018. Tiyuntsong: A Self-Play Reinforcement Learning Approach for ABR Video Streaming. *arXiv preprint arXiv:1811.06166* (2018).

[17] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2019. Comyco: Quality-aware adaptive video streaming via imitation learning. In *Proceedings of the 27th ACM International Conference on Multimedia*. 429–437.

[18] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. 2012. Confused, timid, and unstable: picking a video streaming rate is hard. In *Proceedings of the 2012 internet measurement conference*. 225–238.

[19] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 187–198.

[20] Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. 2019. Optimizing agent behavior over long time scales by transporting value. *Nature communications* 10, 1 (2019), 5223.

[21] Cisco Visual Networking Index. 2015. Cisco visual networking index: Forecast and methodology 2015-2020. *White paper, CISCO* (2015).

[22] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* (2016).

[23] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems* 34 (2021), 1273–1286.

[24] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 97–108.

[25] Nuowen Kan, Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2022. Improving Generalization for Neural Adaptive Video Streaming via Meta Reinforcement Learning. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3006–3016.

[26] István Ketykó, Katrien De Moor, Toon De Pessemier, Adrián Juan Verdejo, Kris Vanhecke, Wout Joseph, Luc Martens, and Lieven De Marez. 2010. QoE measurement of mobile YouTube video streaming. In *Proceedings of the 3rd workshop on Mobile video delivery*. 27–32.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] S Shunmuga Krishnan and Ramesh K Sitaraman. 2012. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*. 211–224.

[29] Yuxiang Lin, Yi Gao, and Wei Dong. 2022. Bandwidth Prediction for 5G Cellular Networks. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[30] Hongzi Mao. 2017. pensieve. https://github.com/hongzimao/pensieve. (2017).

[31] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*. 50–56.

[32] 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*. 197–210.

[33] Ricky KP Mok, Edmond WW Chan, and Rocky KC Chang. 2011. Measuring the quality of experience of HTTP video streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, 485–492.

[34] Ricky KP Mok, Edmond WW Chan, Xiapu Luo, and Rocky KC Chang. 2011. Inferring the QoE of HTTP video streaming from user-viewing activities. In *Proceedings of the first ACM SIGCOMM workshop on Measurements up the stack*. 31–36.

[35] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. 2018. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics* 7, 1-2 (2018), 1–179.

[36] Kandaraj Piamrat, Cesar Viho, Jean-Marie Bonnin, and Adlen Ksentini. 2009. Quality of experience measurements for video streaming over wireless networks. In *2009 Sixth International Conference on Information Technology: New Generations*. IEEE, 1184–1189.

[37] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*. PMLR, 8821–8831.

[39] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. 114–118.

[40] Bojan Rikic, Dragan Samardžija, Ognjen Čadovski, and Tomislav Maruna. 2021. Cellular network bandwidth prediction in consumer applications. In *2021 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 1–3.

[41] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions On Networking* 28, 4 (2020), 1698–1711.

[42] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 272–285.

[43] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[45] Zhengxu Xia, Yajie Zhou, Francis Y Yan, and Junchen Jiang. 2022. Genet: automatic curriculum generation for learning adaptation in networking. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 397–413.

[46] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.

[47] Danfu Yuan, Yuanhong Zhang, Weizhan Zhang, Xuncheng Liu, Haipeng Du, and Qinghua Zheng. 2022. PRIOR: deep reinforced adaptive video streaming with attention-based throughput prediction. In *Proceedings of the 32nd Workshop on Network and Operating Systems Support for Digital Audio and Video*. 36–42.

[48] Yin Zhang and Nick Duffield. 2001. On the constancy of Internet path properties. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. 197–211.

[49] Xuan Kelvin Zou, Jeffrey Erman, Vijay Gopalakrishnan, Emir Halepovic, Rittwik Jana, Xin Jin, Jennifer Rexford, and Rakesh K Sinha. 2015. Can accurate predictions improve video streaming in cellular networks?. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. 57–62.