GreenABR: Energy-Aware Adaptive Bitrate Streaming with Deep Reinforcement Learning

Bekir Oguzhan Turkkan
University at Buffalo
Buffalo, New York, USA
bekirogu@buffalo.edu

Ting Dai
IBM Research
Yorktown Heights, New York, USA
ting.dai@ibm.com

Adithya Raman
University at Buffalo
Buffalo, New York, USA
araman5@buffalo.edu

Tevfik Kosar
University at Buffalo
Buffalo, New York, USA
tkosar@buffalo.edu

Changyou Chen
University at Buffalo
Buffalo, New York, USA
changyou@buffalo.edu

Muhammed Fatih Bulut
IBM Research
Yorktown Heights, New York, USA
mfbulut@us.ibm.com

Jaroslaw Zola
University at Buffalo
Buffalo, New York, USA
jzola@buffalo.edu

Daby Sow
IBM Research
Yorktown Heights, New York, USA
sowdaby@us.ibm.com

ABSTRACT

Adaptive bitrate (ABR) algorithms aim to make optimal bitrate decisions in dynamically changing network conditions to ensure a high quality of experience (QoE) for the users during video streaming. However, most of the existing ABRs share the limitations of predefined rules and incorrect assumptions about streaming parameters. They also come short to consider the perceived quality in their QoE model, target higher bitrates regardless, and ignore the corresponding energy consumption. This joint approach results in additional energy consumption and becomes a burden, especially for mobile device users. This paper proposes GreenABR, a new deep reinforcement learning-based ABR scheme that optimizes the energy consumption during video streaming without sacrificing the user QoE. GreenABR employs a standard perceived quality metric, VMAF, and real power measurements collected through a streaming application. GreenABR's deep reinforcement learning model makes no assumptions about the streaming environment and learns how to adapt to the dynamically changing conditions in a wide range of real network scenarios. GreenABR outperforms the existing state-of-the-art ABR algorithms by saving up to 57% in streaming energy consumption and 60% in data consumption while achieving up to 22% more perceptual QoE due to up to 84% less rebuffering time and near-zero capacity violations.

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06…$15.00
https://doi.org/10.1145/3524273.3528188

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Reinforcement learning**; • **Hardware** → **Platform power issues**.

KEYWORDS

video streaming, energy efficiency, deep reinforcement learning

ACM Reference Format:
Bekir Oguzhan Turkkan, Ting Dai, Adithya Raman, Tevfik Kosar, Changyou Chen, Muhammed Fatih Bulut, Jaroslaw Zola, and Daby Sow. 2022. GreenABR: Energy-Aware Adaptive Bitrate Streaming with Deep Reinforcement Learning . In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3524273.3528188

1 INTRODUCTION

Global Internet traffic continues to grow tremendously, with an annual growth rate of 35%, and it is expected to exceed 250 Exabytes per month in 2022 [1]. Video streaming has the largest share with 60% of this global Internet traffic. Likewise, mobile video streaming produced 59% of global mobile traffic in 2017, and it is expected to reach 79% by the end of 2022 [10]. Dynamic Adaptive Streaming over HTTP (DASH) [2] is one of the leading video streaming technologies behind this escalating load. DASH encodes each video at different bitrates, quality levels, and resolutions; and stores these different video versions in equal-sized segments. The DASH client uses an adaptive bitrate (ABR) algorithm, which helps to select the most suitable representation for the next chunk based on the dynamically changing network conditions.

Most of the existing ABR algorithms [12, 22, 27, 36, 37, 48] share three main objectives: (1) selecting the highest bitrate available, (2) minimizing the number of stalling events due to depleted client buffer, and (3) minimizing the oscillations in bitrate selections. However, the quality of experience (QoE) models designed for these objectives do not consider the *perceptual quality* and hence do not represent the actual streaming quality perceived by users. This

especially becomes significant for mobile users. Due to the limited screen size, the increase in bitrate becomes unrecognizable for the user after a certain degree. Moreover, perceptual quality provides a better understanding of available video representations. It enables more energy-efficient decisions by avoiding higher bitrates if there is no gain in the perceived quality.

Heuristic-based ABR algorithms [36, 37, 40, 48] use specific streaming parameters to create their strategies, such as client buffer level, recently achieved throughput information, and the recently used video representation. They select a representation with lower or higher bandwidth requirements based on the buffer level or the achieved throughput. However, these approaches rely on very recent information and come short of accurately predicting future chunks' behavior. Learning-based algorithms [20, 27, 39] adopt different machine learning methodologies to learn the optimal ABR decisions. CS2P [39] leverages Hidden-Markov Model to learn network characteristics from historical data and uses the predicted throughput values to select corresponding representations. Pensieve [27] uses a policy-gradient Reinforcement Learning (RL) method to optimize ABR decisions under various network conditions. Comyco [20] proposes an Imitation Learning model to select video representations with expert policies. However, those approaches disregard either video perceptual qualities or energy consumption. Moreover, they usually require more resources for training.

Improving the energy efficiency of ABR algorithms did not draw the same attention as the QoE. Existing works propose hardware-related solutions such as changing screen brightness [44] or switching off the NIC during idle time [23].

However, these approaches do not propose a new ABR algorithm. A context-aware model [16] was proposed to adjust the bitrate selections depending on the device's vibration level. EnDASH [32] employed machine learning techniques to dynamically estimate the optimal buffer level and bitrate selections.

Similarly, [28] targets maximizing the QoE under a predefined power budget for 360° videos. The limitations of these approaches are explained in Section 2.

To retain the high perceptual quality while still preserving substantial power savings, we present GreenABR. Specifically, we make the following contributions:

- We propose an energy-efficient ABR solution that enforces intelligent decisions using Deep Q-Networks (DQN) [30] in pursuit of high perceptual quality by avoiding rebuffering events and quality oscillations to reduce power consumption.
- We develop a power model to capture the power consumption patterns of real video streaming sessions with different encoding parameters and video characteristics.
- We employ both real and synthetic traces to capture more diverse network scenarios.
- We compare GreenABR with state-of-the-art video streaming approaches. Our results show that GreenABR excels all other ABRs by saving up to 57% in streaming energy consumption and 60% in data consumption while achieving up to 22% more perceptual QoE due to up to 84% less rebuffering time and near-zero capacity violations. In terms of energy efficiency, achieved QoE per unit energy consumption, GreenABR outperforms all by up to 47%.

The rest of the paper is organized as follows: Section 2 provides background information and discusses the related work in this area; Section 3 explains the GreenABR model; Section 4 presents experimental results and compares GreenABR to other state-of-the-art ABR algorithms; and Section 5 concludes the paper with a discussion on future work. Appendix explains the training details.

2 BACKGROUND AND RELATED WORK

Video streaming approaches. Video streaming over HTTP has become the preferred way of video delivery. DASH has been one of the leading industry standards for video streaming over HTTP since 2012 [2]. DASH enables streaming applications to design customized ABR algorithms for their specific needs. Early ABR algorithms, such as Buffer-Based (BB) [21], occupy the client buffer level or the available throughput level to make bitrate selections. Using these two parameters, available bandwidth and buffer level form the foundation of most of the existing ABR algorithms like BOLA [37], Festive [22], MPC [48], BOLAE [36], Throughput-based [36], DynamicDash [36], and Dynamic ABR [36].

BOLA [37] prioritizes the buffer occupancy level to avoid stalling events and targets a minimum buffer level throughout streaming. It starts with the lowest available bitrate and fills up the client buffer fast. After the buffer threshold is satisfied, it uses the available bandwidth to select the highest available bitrate. BOLAE [36] improves BOLA [37] with additional rules on the available bandwidth. Festive [22], MPC [48], and CS2P [39] employ throughput estimation techniques to decide the highest bitrate that would not cause stalling. Dynamic ABR [36] and DynamicDash [36] dynamically switch between buffer-based and bitrate-based rules based on the current buffer level and available bandwidth.

Several advanced machine-learning methods have been applied to address the ABR decision problems. Pensieve [27] proposes an RL-based approach to target the overall QoE. Oboe [12] employs self-tuning algorithms to dynamically adapt runtime parameters for different network conditions. However, those approaches model the video quality linearly with the encoding bitrate. Similarly, Fugu [46] builds a supervised learning model with the historical streaming data to estimate the transfer time for a selected video chunk and decides the optimal bitrate selection to avoid rebuffering events. However, for QoE measurements, it uses a standard video quality metric SSIM [49], which falls short to model real user perceptions [13, 17]. Lately, Comyco [20] trains its model with imitation learning and uses the standard perceptual quality metric, VMAF, for the QoE calculations. To generate the expert behavior, it assumes full knowledge of network conditions during training and uses dynamic programming to find the optimal selection for future chunks. Although it requires fewer iterations to learn a good policy, it is computationally expensive due to the dynamic programming component. In addition, the needed computation resources grow tremendously for a larger set of representations, leading to unavoidable scalability issues. More importantly, the above ABRs do not consider corresponding energy consumption and can quickly become a burden for mobile devices.

Energy-aware video streaming. Video streaming significantly impacts mobile devices' battery life, and hardware-related solutions are commonly employed to save power. For example, e-DASH [44]

dynamically adjusts screen brightness with different video contents to save battery life, while eff-HAS [23] turns off network connections when a predefined buffer level is reached. Uitto et al. [41] uses HEVC codec for video encoding to achieve similar qualities compared with other codecs while causing less power and data consumption. Chen et al. [16] consider both video bitrate and vibration levels to optimize their ABR algorithm. They propose to avoid higher bitrates under high vibration levels since that may affect the QoE of users. eff-HAS [23] studies user-preference history to predict user retention based on the video content and uses lower bitrates to stream the first few chunks of videos with lower user retention. The above algorithms are tailored to specific use-cases [16], provided as an additional feature [23, 44], or require hardware support [41], thus do not achieve power savings for general purposes.

Recent work EnDASH [32] employs random forest learning to predict the future throughput and uses RL-based methods to adjust the optimal buffer length with the corresponding bitrate selection. Breitbach et al. [15] adopt a similar approach to save streaming energy by predicting network throughput while traveling in a train where bandwidth may be significantly lower during rush hours. Likewise, Meng et al. [28] propose an energy-aware model for streaming 360° videos to optimize the QoE under an energy budget decided by the user for the streaming session. These approaches exploit the high bandwidth intervals and progressively buffer video chunks in advance. As a result, they save streaming energy consumption but sacrifice perceived quality with high oscillations affected by frequent buffer length or power budget adjustments. Moreover, they may waste data and power when users do not play the entire video. QUAD [33] and DataPlanner [34] propose models to maximize the QoE for a targeted quality level or data consumption budget. In addition, they leverage the perceptual quality metrics and propose solutions to improve existing ABRs.

Modeling Video Streaming Energy Consumption. An accurate video streaming energy consumption model is crucial for designing energy-efficient ABR algorithms. Chen et al. [16] propose a linear model based on encoding bitrate for local playback while using a quadratic function and the network signal strength for streaming intervals with an active network channel. Herglotz et al. [19] analyze data acquisition, video processing, display, audio processing, and speaker components separately and propose a feature selection approach to model the combined power consumption. They identify the display brightness, encoding bitrate, and frame rate as the major power consumption parameters. Similarly, Yue et al. [47] categorize streaming power consumption in CPU, display, network, and residual power. They propose a separate model for each component for regular and 360° videos. All the above approaches target estimating the power consumption value instead of capturing the pattern. Thus, these approaches may only work well for the studied devices.

Limitations of existing streaming approaches. Existing ABR algorithms share the following common limitations: (1) using a small number of representations; (2) calculating the user QoE based on bitrate levels; (3) assuming a linear relation between the video quality and the encoding bitrate without considering the perceptual quality; and (4) achieving energy savings but reducing overall QoE or pursuing higher QoE with significant energy consumption. ABR algorithms developed with prior training like Pensieve [27] and Oboe [12] assume only a limited number of video representations. However, real streaming applications like Netflix use more representations for the served videos [7]. Therefore, learning models adapting to different representation sets are required in real-world video streaming scenarios. In addition, policy-based models like Asynchronous Advantage Actor-Critic (A3C) [29] used in Pensieve is known to be sensitive to the training parameters as evaluated in our training performance experiments in Section 4.2.4.

Encoding bitrate and perceived quality are not linearly related when video contents, mobile devices, and distances between users and devices are considered [13]. In this manner, targeting the highest possible bitrate does not necessarily increase the perceived QoE as expected. In fact, choosing the highest possible bitrate may even hurt the overall QoE as it may lead to stalling events under unstable network conditions due to an insufficient buffer level. Moreover, without considering the high power consumption coupled with the high bitrate choice, streaming videos can quickly drain the battery of mobile devices. Pensieve, Festive, Oboe, BB, BOLA, BOLAE, Throughput-based, and Dynamic ABR all fall in this category. Furthermore, BB, BOLA, and MPC do not consider the oscillations while changing bitrate levels, significantly hurting overall QoE.

Addressing the aforementioned limitations of ABR approaches and getting the balance of perceptual quality and power savings motivate our work on GreenABR.

3 GREENABR DESIGN

GreenABR is an energy-efficient video streaming system that takes energy consumption, network dynamics, video quality, and other video player measurements into consideration to reinforce high-quality video streaming with substantial energy savings. The overall architecture of GreenABR is shown in Figure 1. In this section, we discuss GreenABR design with its energy model (Section 3.1) and the deep reinforcement learning based ABR algorithm (Section 3.2). The reward function and video player metrics are included in Section 3.2 while network dynamics are represented by throughput traces in our data set (Section 4.1).

3.1 Energy Model

Design principle. The fundamental design principle of our energy model is to capture the energy consumption pattern related to video streaming while minimizing device specification influence. We minimize device heterogeneity (and its impact on energy expenditure) by excluding baseline power consumption, which includes the power for OS usage, CPU frequencies, memory allocations, screen consumption, and video player's operation power without streaming.

Herglotz et al. [19] show that five essential components contribute to the overall energy consumption for video streaming: *data acquisition*, *video processing*, *displaying video files*, *audio processing*, and *speaker operations*. We model the first three components' consumption as they are the most relevant to ABR decisions. The audio files are commonly provided with a single version, encoded with a high bitrate due to its negligible file sizes compared to video files. In addition, speaker power consumption is highly affected by user interactions, i.e., users can adjust speaker volumes. Therefore, we excluded these two components in our modeling work to focus

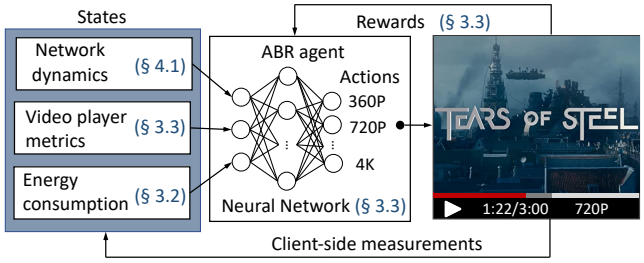


Figure 1: The overview of GreenABR.

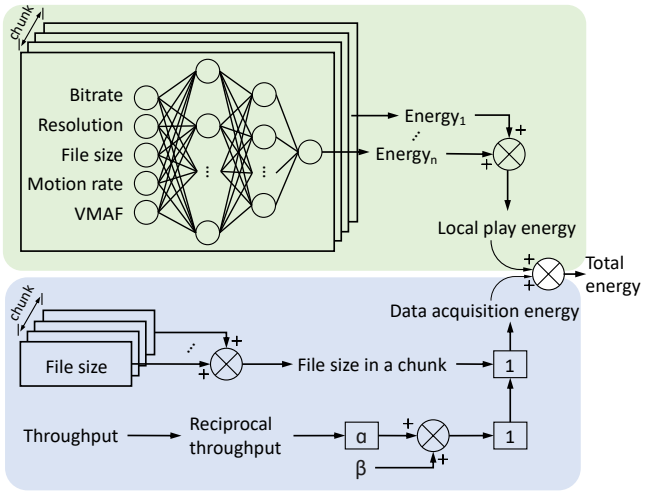


Figure 2: In GreenABR energy model, the total video streaming energy has two major contributors: local playback and data acquisition.

on only ABR-related elements. Other human interactions such as pausing videos, and changing screen brightness levels, can affect the total energy consumption, but they are out of the scope of this paper and excluded in the experiments. We should note that we use the energy model's estimations for GreenABR training while using the real collected measurements from the power meter for energy-saving evaluations. Details will be discussed in Section 4.2.

3.1.1 Energy Consumption Breakdown. As shown in Figure 2, we model video streaming energy consumption in two parts: *local playback* and *data acquisition*. The local playback energy consumption includes the processing and displaying video files components. Such breakdown makes it easy for us to independently measure each part's energy usage and build the corresponding models. Specifically, when measuring local playback energy consumption, we store the videos locally on the device and play them directly without any network involvement. Similarly, to solely measure the data acquisition energy consumption, we download and buffer the videos without playing them.

3.1.2 Local Playback Energy Consumption. Local playback energy consumption is closely related to encoding parameters and video characteristics, including *encoding bitrate*, *resolution*, *frame rate*,

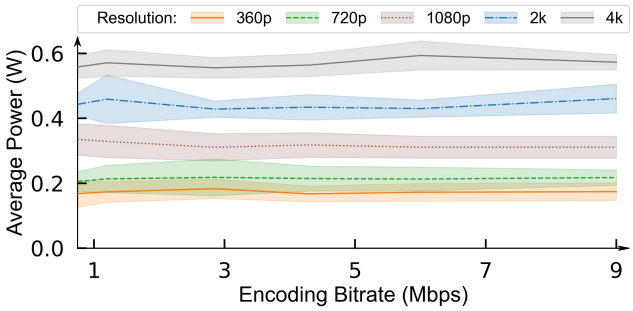


Figure 3: Average local play power consumption for different encoding bitrates and resolutions. Lines are mean values while shaded shapes correspond to the variance.

file size, *motion rate*, and *quality measurements* (i.e., VMAF). Encoding bitrate br decides the amount of video data to be processed, while resolution rs sets the number of pixels to store each frame's video data. Motion rate mr reflects the scene changes resulting in different data sizes of reference frames[1]. VMAF with file size fs is another indicator of video complexity. The parameters above (br, rs, fs, VMAF, mr) are included in our playback energy model, shown in Figure 2. Frame rate decides the number of frames in one second, which is always set by a fixed value for different video representations, thus excluded in our model.

Our measurements suggest that all the aforementioned parameters affect power consumption, and such an effect is not linear to a single parameter but intertwined by multiple factors. Figure 3 shows the local play power consumption for a video with different settings. The average power is highly related to resolutions. However, even with the same resolution, the power consumption is significantly fluctuated, shown by the high standard deviations, which indicates the compound effect of other parameters, such as motion rate and file size. Existing power models [16, 19, 23] solely used the encoding bitrate to estimate the local playback power, which is inaccurate. Other video characteristics such as resolution can affect power consumption more significantly.

Our model's dataset is created by measuring the local playback energy consumption of videos in several genres *w.r.t* different settings of content-related parameters (br, rs, fs, VMAF, mr) in a mobile device, Samsung Galaxy S4. The details of environment setup and measurements are described in Section 4.1.

Training Model. We choose a regression model to build the local playback energy model by training a feed-forward neural network with one input layer, two hidden layers, and one output layer, as shown in Figure 2. We use the root mean squared error (RMSE) for our loss function:

$$\text{RMSE} := \sqrt{\frac{1}{N}\Sigma_{i=1}^{N}(p_i - o_i)^2}, \quad (1)$$

where N is the number of samples, p_i is the predicted value, and o_i is the true observed value.

[1]A video contains multiple chunks. A chunk contains multiple seconds' video information. In each second, there is one keyframe and several reference frames.

To capture the power pattern dynamics (i.e., percentage change), we normalize all parameters with the maximum values in the pre-processing ($br/br_{\max}, rs/rs_{\max}, fs/fs_{\max}, \texttt{VMAF}/\texttt{VMAF}_{\max}, mr/mr_{\max}$).

We split the dataset randomly as 80% training and 20% testing, and train the model until it converges with RMSE less than 0.01 that corresponds to 7% estimation error. We use power measurements collected with Samsung Galaxy S4 for training. To evaluate the generalization of our model, we collect the power measurements for the same videos with another device, Samsung XCover Pro. Our model is accurate, with RMSE less than 0.036 (12% estimation error) for the new device. It indicates that our model successfully captures the power consumption pattern without additional training. We believe the slight degradation in the accuracy is due to different hardware technologies of the devices such as the screen type, although their impact is minimized by excluding base power consumption and pre-processing the data for power attributes.

3.1.3 Data Acquisition Energy Consumption. Data acquisition energy consumption in video streaming is related to *throughput* and *file size* as they decide how frequently and persistently the network interface card is occupied for data transfer. We adopt the existing throughput-based model [38] to calculate the data acquisition energy:

$$E_{data} := (\alpha * th^{-1} + \beta) * \Sigma_{i=1}^{M} fs_i, \tag{2}$$

where E_{data} is the amount of energy to download a video chunk, $\Sigma_{i=1}^{M} fs_i$ is the data size of a video chunk, th is the achieved throughput, and α and β are two constant parameters. In our experiments, we use the recommended values [38] for TCP transfer as $\alpha = 210$ and $\beta = 28$.

The above throughput-based model is suitable for video streaming for two reasons. First, the throughput range for the model is compatible with the network traces we use for the GreenABR agent. Second, the model estimates the energy consumption with around 90% accuracy in our experiments. After the energy consumption of the local playback and data acquisition are calculated, the summation of the two is used as the total energy consumption estimation for a selected video chunk.

3.2 ABR Models with Deep Reinforcement Learning

Why RL for ABR? ABR algorithms target maximizing the overall video streaming QoE by selecting the appropriate version of video chunks to dynamically adapt streaming environments, e.g., network throughput, client buffer size, power consumption. RL can tackle this problem well, as it embodies the same goal– intelligently taking the desired actions in user environments to maximize the cumulative reward. Furthermore, unlike supervised learning, RL-based approaches do not make assumptions on or pre-label the network or streaming parameters but learn by experience, which is more flexible and labor-ease.

Solving video streaming in an RL-way. RL-based approaches aim to learn an optimal policy by maximizing the expected cumulative reward for an agent interacting with an unknown environment. In designing an ABR algorithm with RL techniques, this corresponds to selecting the correct video version that can produce the best QoE. Consequently, QoE can be naturally treated as the reward function, while network dynamics, streaming parameters, and power consumption form the state space, and different video representations are the elements of the action space. Streaming the entire video can be considered one episode of RL, and the set of decisions for given states at each step/chunk is induced from the optimal policy, which is the goal of the RL-based ABR algorithm.

Two popular learning frameworks of RL are DQN-based methods and policy-gradient-based methods such as A2C and its extensions. Although policy-gradient-based methods are more general than DQN-based models, they are typically more challenging to be trained due to the high variance in gradient estimation [4, 18, 45]. By contrast, DQN-based approaches do not suffer from such a problem. Due to this merit, their training is typically more stable and requires much fewer training iterations than their counterparts. Moreover, DQN-based approaches are less sensitive to hyperparameters and changes in the action space, making their training algorithms easy to adopt without much change. Once trained, they can be easily adapted to different client environments, i.e., mobile devices. For the above reasons, we use DQN to build our ABR agent, shown in Figure 4. The advantage of our DQN based model over policy-gradient-based methods will be demonstrated in the experiments.

States. For each chunk of video, GreenABR's learning agent takes energy consumption, network dynamics, and video player measurements as the input state. Specifically, energy consumption is the estimated value for the last chunk predicted by our power model. For training, we use the power model estimations rather than actual measurements since collecting power measurements for each video is not feasible and does not serve for a general model. Network dynamics include the network throughput and download time for the last video chunk. Video player measurements include the current buffer size, the bitrate at which the last chunk was downloaded, and the corresponding VMAF value.

Actions. In ABRs, videos are fragmented as chunks, and each chunk has multiple representations for adjustive selections. All different representations form GreenABR's action space. A video representation contains the following parameters: *encoding bitrate*, *resolution*, *frame rate*, and *codec*. Typically for the same video, all representations share the same frame rate and codec. In our experiments, we encode all videos with a frame rate of 24fps in H.264 format. Selections about encoding bitrate and resolution in our action space are discussed in Section 4.1.

Rewards. Our QoE reward consists of quality, smoothness, rebuffering, and energy consumption, defined as:

$$\begin{aligned}\texttt{QoE}_i := \alpha * \texttt{VMAF}_i + \left\lfloor \frac{\text{sgn}\,(\alpha * \texttt{VMAF}_i - \beta) + 1}{2} \right\rfloor * 2^{\alpha * \texttt{VMAF}_i - \beta} \\ - r_p * rt_i - \alpha * |\texttt{VMAF}_i - \texttt{VMAF}_{i-1}| \\ - \gamma * E_i - \left\lfloor \frac{\text{sgn}\,(\gamma * E_i - \zeta) + 1}{2} \right\rfloor * 2^{\gamma * E_i - \zeta},\end{aligned} \tag{3}$$

where $\alpha * \texttt{VMAF}_i$ is the video quality, $2^{\alpha * \texttt{VMAF}_i - \beta}$ is the quality amplifier, $r_p * rt_i$ is the rebuffering penalty, $\alpha * |\texttt{VMAF}_i - \texttt{VMAF}_{i-1}|$ is the smoothness penalty, $\gamma * E_i$ is the energy consumption penalty, $2^{\gamma * E_i - \zeta}$ is the energy penalty amplifier, i is the chunk number, $\texttt{VMAF}_i$ is the video multimethod assessment fusion metric [14] for

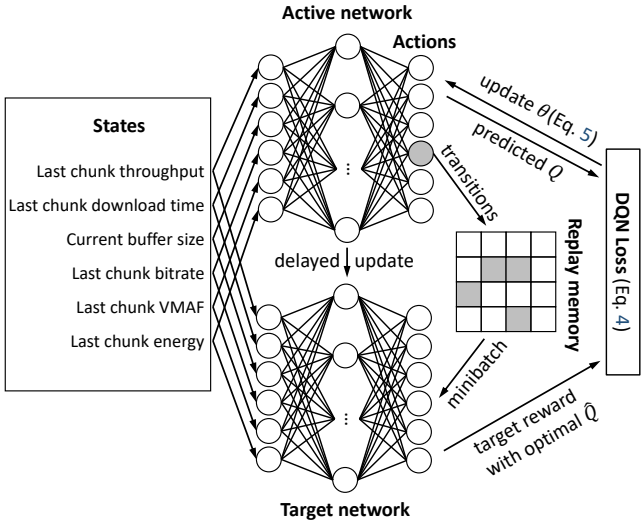


Figure 4: The DQN architecture used by GreenABR.

the chunk i, rt_i is the total rebuffering time while processing the chunk i if stalling occurs, r_p is the penalty constant for the stalling events, and E_i is the energy consumption for chunk i excluding the minimum energy[2]. We set $\alpha = 0.05$, $\beta = 3$, $\gamma = 0.001$, and $\zeta = 2$. We use two sign functions $\left\lfloor \frac{\text{sgn}(\alpha * \text{VMAF}_i - \beta) + 1}{2} \right\rfloor$ and $\left\lfloor \frac{\text{sgn}(\gamma * E_i - \zeta) + 1}{2} \right\rfloor$ to make sure that the quality and energy penalty amplifiers work only when the corresponding quality and penalty metrics $\alpha * \text{VMAF}_i$ and $\gamma * E_i$ exceed their thresholds β and ζ.

Our reward function has multiple components with different ranges of natural magnitudes. To disentangle such magnitudes in the reward calculation and, more importantly, to ease the hyper-parameter tuning [43], we normalize all the reward and penalty components with constant parameters, i.e., α, γ and r_p. Specifically, we set $\alpha = 0.05$, which makes the video quality $\alpha * \text{VMAF}_i$ range from 0 to 5 since VMAF's range is [0, 100] [14]. Likewise, α is also applied on smoothness penalty with an absolute difference in consecutive chunks' VMAF values. We set the first chunk's smoothness penalty as 0 since it does not have a previous chunk. For the rebuffering penalty, we set r_p with the highest bitrate (Mbps) in the action space. For example, with the largest encoding bitrate as 4.3Mbps in a video representation, we set $r_p = 4.3$. For energy penalty, we set γ as 0.001 for the normalization purpose since we use millijoule as the energy unit, and each chunk's energy consumption ranges from 1000 to 4500 millijoules.

We set the quality threshold $\beta = 3$. Videos with quality levels from 2 to 3 are considered to have fair qualities [13]. The quality amplifier $2^{\alpha * \text{VMAF}_i - \beta}$ rewards exponentially for fair, good, and excellent qualities with quality levels as [2, 3), [3, 4), and [4, 5), respectively. Encoding with high bitrates always produces high-quality videos. However, such a relationship is not linear as shown by Figure 5. It

[2]The energy consumed to process a chunk throughout the whole video with minimum settings.

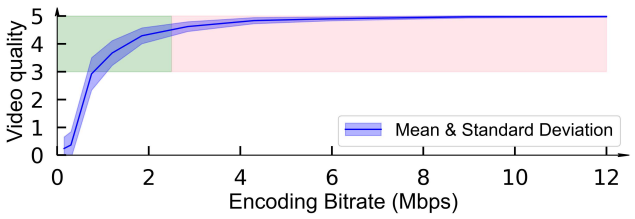


Figure 5: Video perceptual quality ($\alpha * \text{VMAF}$) to encoding bitrate in phone model measurements.

shows that video quality increases logarithmically with encoding bitrate. Increasing encoding bitrate from 0.8 to 2.5 boosts video quality from 3 to 4.5 (green region box), which improves human perceptual quality. However, any further increase is not necessary (pink box), because human eyes cannot distinguish the video quality from 4.5 to 5 [13]. However, the increment of encoding bitrate might increase other video parameters, such as resolution, which can significantly boost energy consumption shown in Figure 3.

To avoid our model greedily choosing the highest bitrate, we introduce the energy penalty amplifier $2^{\gamma * E_i - \zeta}$, which penalizes high energy consumption exponentially after the threshold ζ. We set $\zeta = 2$ based on our measurements. We observe that videos with a quality level 4.5 consume 2 joules on average. Pursuing higher quality than 4.5 with higher bitrate or resolution does not increase overall perceptual quality but instead consumes unnecessary energy. For example, as shown by the top right shaded area in Figure 5, aggressively choosing a bitrate higher than 2.5 Mbps with already high video quality is undesired. Such behavior is punished exponentially by our energy penalty amplifier, which statistically guides our model to make energy-efficient choices.

DQN training. Our ABR agent uses a four-layer feed-forward neural network (Figure 4) as the deep Q-network (DQN) to calculate the expected cumulative reward $Q(s, a)$ for each state-action pair (s, a). The neural network weight parameter θ is learned from the feedback of the client environment in the form of QoE described in Equation 3.

Our training process is described in Algorithm 1. In each training iteration, we use the ϵ-greedy strategy to choose an action to balance the *exploration-exploitation* trade-offs. Specifically, starting from a high ϵ value, we randomly explore the action space (Line #6-8) with selected representation a_t. With more iterations, the ϵ is decayed gradually to a lower value (Line #17), which guides us to exploit the neural network to select the action (Line #9-10), i.e., $a_t = \arg\max_a Q(s_t, a; \theta)$. We leverage a *target network* $\hat{Q}$ to preserve the previously acquired knowledge by avoiding catastrophic forgetting [24] which may result in the model converging to a local minimum. The target network $\hat{Q}$ has the same neural network structure (Line #3) but delays the updates on the weight $\hat{\theta}$ with every C steps (Line #18).

Furthermore, we maintain an *experience replay* memory queue $\mathcal{D}$ to store state transitions, which sustains the stability of our training process [31]. To perform update of the weight parameter θ, we randomly sample a batch of transitions $\mathcal{B} = (s_j, a_j, r_j, s'_j)_{j=1}^{n}$ from $\mathcal{D}$ (Line #14). With the sampled mini-batch, the model is

Algorithm 1: DQN Algorithm with Target Network and Experience Replay [31].

1 Initialize replay memory $\mathcal{D}$ to capacity N ;
2 Initialize Q network with random weights θ ;
3 Initialize target network $\hat{Q}$ with $\hat{\theta} = \theta$;
4 **for** *Each Episode* **do**
5 **while** *not the end of the video* **do**
6 Select random r between 0 and 1;
7 **if** $r < \epsilon$ **then**
8 Select a random action (representation) a_t ;
9 **else**
10 Select action $a_t = \arg\max_a Q(s_t, a; \theta)$;
11 **end**
12 Take action a_t and observe QoE reward r_t (Equation 3 or ??) and state s_{t+1} ;
13 Store transition (s_t, a_t, r_t, s_{t+1}) to memory $\mathcal{D}$;
14 Sample a random batch of transitions $(s_j, a_j, r_j, s'_j)_{j=1}^{n}$ from $\mathcal{D}$;
15 Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1, \\ r_j + \gamma \hat{Q}\left(s'_j, \arg\max_{a'} Q(s'_j, a'; \theta); \hat{\theta}\right) & \text{otherwise.} \end{cases}$
16 Perform gradient descent on $\frac{1}{n}\sum_{j=1}^{n}((y_j - Q(s_j, a_j; \theta)))^2$ *w.r.t* θ;
17 Decay ϵ;
18 Every C steps, reset $\hat{Q} = Q$;
19 **end**
20 **end**

updated periodically by minimizing the loss function:

$$\mathcal{L}(\theta) := \mathbb{E}_{s,a,r,s' \sim \mathcal{B}}\left[(y - Q(s, a; \theta))^2\right], \tag{4}$$

where $y = r + \gamma \hat{Q}\left(s', \arg\max_{a'} Q(s', a'; \theta), \hat{\theta}\right)$ is the target calculated by summing up the current reward r and the optimal Q-value in the subsequent step. The Q-value is estimated by the target network while the action is decided by the up-to-date Q-network. Indexing transitions from $\mathcal{B}$ with j, the parameter θ is learned using gradient descent of the loss function to close the gap between Q-value predicted by the Q-network and the target optimal Q-value (Line #16):

$$\theta \leftarrow \theta + \frac{\alpha}{n}\sum_{j=1}^{n}\left(y_j - Q(s_j, a_j; \theta)\right)\nabla_\theta Q(s_j, a_j; \theta), \tag{5}$$

where α is the hyper-parameter for learning rate, and y_j is the target calculated in Line #15.

4 EVALUATION

This section describes the evaluation methodology and experimental results. GreenABR, including power model and reinforcement learning agent, is implemented in Python. Power measurements are collected from Samsung Galaxy S4 and Samsung XCover Pro smartphones with Android 7.1 and 11.0 operating systems.

4.1 Evaluation Methodology

Video types. In our experiments, we use the first three-minute clips of three videos from three different genres to collect power data and compare the performances of different ABR algorithms. These videos are "Big Buck Bunny", "Tears of Steel", and "Nature" videos in cartoon, sci-fi, and documentary genres, respectively. For all ABRs, we use the "Tears of Steel" video for training and all videos for testing. Since the results are very similar for all genres of videos, we did not exclude the training video from evaluations.

Table 1: Representation sets for the experiments.

Action spaces			Resolution	Bitrate (Kbps)
6 reps	10 reps	10 HD reps		
	✓	✓	320 × 180	150
✓	✓	✓	320 × 180	300
✓	✓	✓	640 × 360	750
✓	✓	✓	768 × 432	1200
✓	✓	✓	1024 × 576	1850
✓	✓	✓	1280 × 720	2850
✓	✓	✓	1920 × 1080	4300
	✓	✓	1920 × 1080	6000
	✓		1920 × 1080	9000
	✓		1920 × 1080	12000
		✓	2560 × 1440	9000
		✓	3840 × 2160	12000

Action spaces. We encoded all videos into 12 representations, as shown in Table 1. All representations use the same frame rate with 24fps and codec H.264, but are distinguished by different resolutions and encoding bitrates. Our representation sets are compatible with the academic [25, 27, 40] and industrial [5, 7, 9] recommendations.

Energy data collection. In our experiments, we powered our phones with a Monsoon power meter [6] by bypassing the internal battery to measure the energy consumption. We played the videos with our instrumented application based on Google's ExoPlayer. Specifically, for the tested three-minute-long videos, we encoded them with 12 representations in table 1. Each representation contains 45 ($3 * 60/4$) chunks with a chunk size of 4 seconds. While playing the videos, the power meter collected the instant power usage of the phone every 200 microseconds and streamed the logs to the connected PC. In total, we have created 1,620 samples with <video index, chunk index, representation index, energy consumption> as our local playback *energy profile*.

To create the local play profile, we calculate energy consumption per second by taking the mean value of the measured 5000 ($10^6/200$) power readings [11]. We then group the mean energy values into chunks, and use the summation in each group to represent that chunk's total consumption.

Network traces. To evaluate ABR approaches under real-world streaming scenarios with mixed network types and wide bandwidth ranges, our network trace benchmarks include 3G traces from HSDPA [3], 4G/LTE traces from Belgium [42], and the FCC broadband traces [8] used in Pensieve and Sabre [36] simulators[3]. To include non-stationary network behaviors such as temporal disconnection and network mode switch, which are not covered by the aforementioned network traces, we have produced synthetic traces to complete our benchmarks while using similar bandwidth range. To produce synthetic traces, we first randomly selected bandwidth levels from predefined ranges, i.e., (0, 1.2], (1.2, 2], (2, 4], (4, 10], and (10, 20] Mbps every 10 seconds. During each time interval, we selected a random throughput within the corresponding bandwidth range every one second. With the random exploration, we were able to capture fluctuated network conditions and changing network types. The distribution of the network traces and their bandwidth levels

[3]Pensieve considers package headers while Sabre does not. To make a fair comparison, we modified Sabre code to make it consider package headers.

Table 2: Distribution of Network Traces.

Trace Type	Bandwidth (Mbps) Avg	Min	Max	>12	Distr. Percent
3G	1.29±0.77	0.0	4.36	0%	29%
4G	31.58±13.59	0.5	64.79	90%	14%
Broadband	3.91±2.37	0.3	6.95	9%	28%
Synthetic	4.41±5.03	0.0	19.02	13%	29%

Table 3: Components of QoE Model.

Component	Description
$\sum_{i=1}^{n} \mathtt{VMAF_i}$	Total video quality
$\sum_{i=1}^{n} rt_i$	Total rebuffering duration
r_c	Total number of rebuffering events
$\sum_{i=2}^{n} (\lvert \mathtt{VMAF_i} - \mathtt{VMAF_{i-1}} \rvert)$	Total smoothness change
$\sum_{i=2}^{n} \lfloor (\lvert \mathtt{VMAF_i} - \mathtt{VMAF_{i-1}} \rvert) /20 \rfloor$	Total number of quality switches

are shown in Table 2. We used fewer 4G traces since they have less diverse throughput levels with very high bandwidth for more than 10% of the time. We randomly selected 70% of all traces for training for all network trace groups and used the rest for testing ABR algorithms.

Compare with other ABR approaches. We compared GreenABR with state-of-the-art ABR schemes[4] including *buffer* based approaches (i.e., BOLA and BOLAE), *bandwidth* based approach (i.e., Throughput-based), *buffer and bandwidth* based approaches (i.e., Dynamic ABR, DynamicDash), and *reinforcement learning* based approach (i.e., Pensieve). Although QUAD and Comyco are similar in using VMAF, the source code of QUAD is not available and training Comyco for ten representations is not stable and causes computational explosion due to its dynamic programming component used for its expert behavior. Similarly, [28] is not included in the evaluations due to the significant differences of power consumption patterns of 360° and regular videos.

We compared the achieved QoE and power consumption of each approach in three sets of representations. As shown in Table 1, the first set contains six representations, which are out-of-the-box actions used by Pensieve. The second set contains ten representations to show the high dynamics and broader range of real-world scenarios. The third set also contains ten representations, including 2K and 4K resolutions with high encoding bitrates. The reason we have two different sets of 10 representations is that our Samsung Galaxy S4 phone does not support 2K (2560 × 1440) or 4K (3840 × 2160) resolutions due to its decoder limit—the maximum supported resolution for the H.264 codec is 1920 × 1088. Our other test phone can play 2K and 4K videos without issues. We used the first ten representation set to evaluate ABRs in broader action ranges without any hardware limit while using the second ten representation set to evaluate ABRs' practicality and generality against capacity violations. The range of the used encoding bitrates is compatible with industry and academic standards [5, 7, 9, 17, 25, 40].

Perceptual quality calculation. RL-based ABR models commonly include essential components of their design (e.g., the energy penalty and the quality amplifier) in their reward function and train their models to maximize the achieved reward. They usually use the same reward function to calculate the QoE in their evaluations. However, ABRs trained with another reward function or ABRs with predefined rules may not consider these components as part of their QoE design. Such differences in QoE design may result in misleading evaluations and unfair comparisons. Thus, a more general QoE calculation is vital for fair evaluations.

For such a general model, we found the achieved video quality, rebuffering duration and frequency, oscillations in the video quality, and number of quality switches to be the dominant components of perceptual QoE in the existing studies [17, 26, 35]. Therefore, to compare the achieved perceptual quality of different ABR approaches fairly, we developed a perceptual QoE model with these components based on the Waterloo Streaming QoE Database III (SQoE-III)[5] [17]:

$$\begin{aligned} \mathtt{QoE} := \alpha * \sum_{i=1}^{n} \mathtt{VMAF_i} - \beta * \sum_{i=1}^{n} rt_i - \gamma * r_c \\ -\sigma * \sum_{i=2}^{n} (\lvert \mathtt{VMAF_i} - \mathtt{VMAF_{i-1}} \rvert) \\ -\mu * \sum_{i=2}^{n} \lfloor (\lvert \mathtt{VMAF_i} - \mathtt{VMAF_{i-1}} \rvert) /20 \rfloor, \end{aligned} \tag{6}$$

where QoE is the quality of experience of a video streaming session, n is the total number of video chunks, $\alpha, \beta, \gamma, \sigma, and \mu$ are coefficients of the components listed in Table 3. As suggested in [13, 33], we consider a difference of 20 in average VMAF values of two consecutive chunks as a quality switch.

We developed our model by using linear regression on the SQoE-III dataset by splitting it as 70% training and 30% testing. As suggested in [17], we repeated the training for 1000 times to avoid unbalanced distribution of data. As a result, we set $\alpha = 0.0771$, $\beta = 1.2497$, $\gamma = 2.8776$, $\sigma = 0.0494$, and $\mu = 1.4365$. We evaluated the performance of different QoE calculations by leveraging the Spearman correlation coefficient (SRCC) as suggested in [17]. The QoE calculation used in Pensieve achieved 0.6563, while our QoE calculation in Equation 6 achieved 0.7845 of SRCC. It indicates one of the best performances among the evaluated QoE models in the SQoE-III dataset, and hence, the proposed QoE calculation is suitable for a standard evaluation. The other ABRs we used in our evaluations do not propose any specific QoE calculation method. Our results show that leveraging VMAF as the video quality metric, including the number of rebuffering events and quality switches lead to better performance to capture the perceived QoE of real users.

We should note that GreenABR shows significantly better performance when we run our evaluations in Section 4 with a simplified version of our training reward function in Equation 3, which achieves 0.7145 of SRCC in the SQoE-III dataset. However, to avoid any unfair comparisons, we used Equation 6 for all of our evaluations due to its higher association with the perception of real users.

[4] All ABRs use their out-of-the-box parameters except action spaces.

[5] SQoE-III is a large realistic dataset for DASH streaming with subjective scores. It consists a total of 450 videos with diverse video content and distortions

Energy consumption calculation. We measure the overall energy consumption of each ABR by summing up their local playback energy and data acquisition energy. Specifically, for local consumption, we query our local playback *energy profile* to get the actual measurements for the specific chunk. For data acquisition consumption, we use Equation 2 with the throughput and chunk size as the inputs for specific network traces. We excluded mobile devices' base energy consumption within the three-minute streaming unless rebuffering events happen, prolonging the overall streaming time with additional energy consumption. Specifically, without any rebuffering, the overall energy consumption is

$$\sum_{i=1}^{n} (El_i + Ed_i - Eb), \tag{7}$$

where El is the local energy, Ed is the data acquisition energy, Eb is the base consumption, and n is the chunk numbers (45 in our experiments). With additional rebuffering time as Tr, the overall energy consumption is

$$\sum_{i=1}^{n} (El_i + Ed_i - Eb) + Eb * Tr. \tag{8}$$

Energy efficiency calculation. Selecting higher resolutions requires more energy consumption while providing better QoE. We define the QoE per energy consumption as the energy efficiency that represents the QoE gain per unit power consumption. It enables fair comparison of ABR decisions for the energy consumption-QoE trade-off.

Capacity violation calculation. Video frames can get dropped when the video streaming process exceeds the mobile device codec's capacity cap. When all reference frames in one second are dropped, stuttering happens. We define each stuttering event as a *capacity violation*. We log the decoder outputs during the playback and group them into video chunks by taking the number of violations. Then, we create a reference table for capacity violations in the form of <video index, chunk index, representation index, capacity violation> to be used by the evaluations in Section 4.2.3.

4.2 Evaluation Results

Figures 6 and 7 show the normalized results against the highest value in each category, including *energy efficiency*, *average QoE*, *rebuffer time*, *smoothness change*, *energy consumption*, and *data usage*. Overall, GreenABR outperformed all other ABRs in terms of energy efficiency with the lowest energy consumption and the smallest data usage while providing same or better QoE. Since the energy efficiency and energy consumption patterns are similar for Galaxy S4 and XCover Pro, we present our results for average energy efficiency and energy consumption of both devices.

4.2.1 ABRs with Six Representations. GreenABR consumes the least amount of energy while achieving the second-best QoE (only 1% degradation), as shown in Figure 6. This matches our design, where our DQN reward function (Equation 3) considers energy consumption and video quality while our ABR agent learned to make balanced quality-energy trade-offs. Such performance and efficiency are also reflected by GreenABR's 49% to 72% less rebuffering time and 3% to 44% less data usage compared with others. Moreover, GreenABR attains 11% to 28% of energy savings compared to the other ABRs. Even though Throughput-based ABR has the second-lowest energy consumption after GreenABR, it still consumes 11% more energy and sacrifices overall perceptual quality with a 21% reduction compared with GreenABR. Overall, GreenABR achieves 17% to 29% better energy efficiency compared to other ABRs.

GreenABR provides better energy efficiency for XCover than Galaxy S4. The heterogeneity of mobile devices causes the difference. Since each device consumes different base energy, the overall local playback energy can also differ (Equation 8) due to rebuffering events. Generalizing over multiple video contents is essential for learning-based models since they may suffer from overfitting to the training video. To evaluate how our approach generalizes, we use only the "Tears of Steel" video for the training of our model and include the "Big Buck Bunny" and "Nature" videos for testing. Our experiments indicate very similar results for all three videos and we use the average values for our evaluations.

4.2.2 ABRs with Ten Representations. Figure 7 shows that the benefit of GreenABR's energy savings is significantly enlarged in two ten-representation sets. GreenABR saves 15% to 41% of energy consumption on average for all videos with ten representations, as shown in Figure 7a. The main driver for such energy gain is consuming 36% to 59% less data with 53% to 84% lower rebuffering time. Furthermore, GreenABR's achieved QoE is 3% to 22% higher than others. Such achievement is not a compromise for energy savings. In fact, they both benefit from our DQN agent with quality and energy considerations. As a result, GreenABR achieves 27% to 47% better energy efficiency. With the inclusion of 2K and 4K resolutions in Figure 7b, GreenABR achieves greater energy savings of 35% to 57% on average for both devices. In this case, GreenABR's achieved QoE is 1% to 22% higher than others while sustaining 34% to 55% better energy efficiency.

Our experiments show that exceeding the decoding capacity of a device highly impacts energy consumption. Figure 8 presents the total energy consumption of Galaxy S4 and XCover Pro for the two representation sets with ten versions in Table 1. All ABRs except GreenABR consume at least 50% more energy on average for Galaxy S4 due to the incurred capacity violations. On the other hand, the energy consumption of ABRs has a slight increase for XCover Pro since it does not suffer from capacity violations. This increase is mainly caused by the resolution differences in the representation sets. It, in fact, validates our observation—aimlessly increasing resolutions can rapidly deplete the battery with more video processing (Figure 3). More details about capacity violations will be discussed in Section 4.2.3.

Discussion: The performance degradation of Pensieve from Figure 6 to Figure 7 is the highest, because the out-of-the-box hyperparameters in Pensieve are incompatible with new representation sets. We believe Pensieve's performance can be improved with more hyperparameter tuning, different reward functions, or larger training iterations. To validate this, we re-trained Pensieve with the reward function in Equation 3 and named it Pensieve-E. We compared Pensive-E with GreenABR. Even though Pensieve-E achieved slightly better (4%) QoE than GreenABR, it increased the energy consumption significantly (27% to 45%). We never diminish the advantage of Pensieve, but we present the generalization

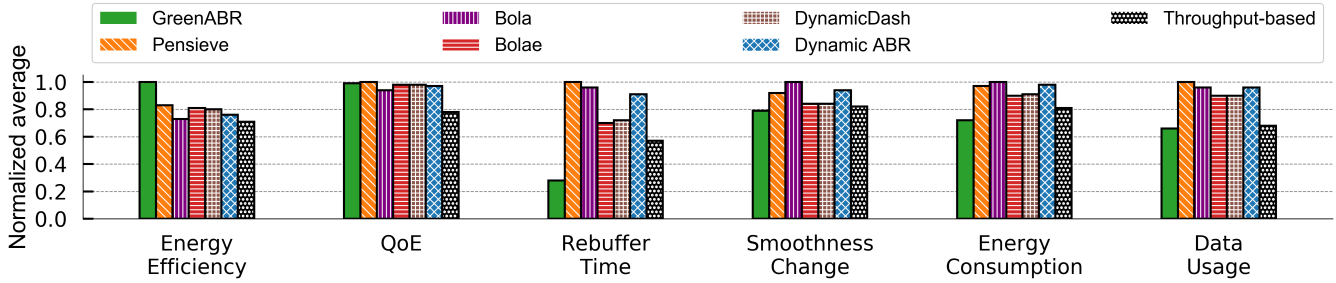
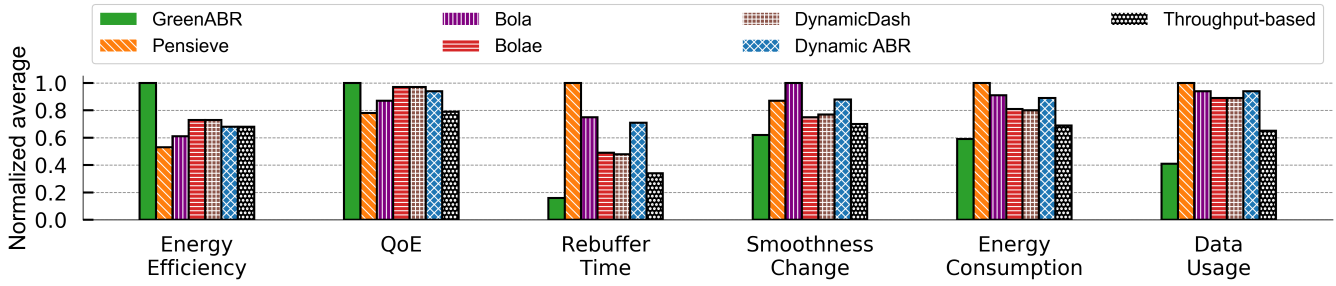
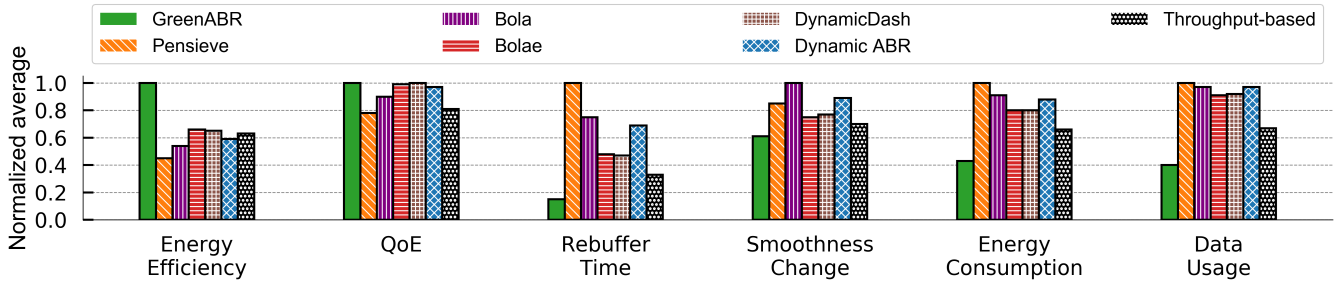


Figure 6: Comparison of GreenABR with other approaches with six representations. For QoE and efficiency, the higher the better. For other metrics, the lower the better. Average results for all videos are used.



(a) Average results for all videos with ten representations.



(b) Average results for all videos with ten representations including 2K and 4K.

Figure 7: Comparison of GreenABR with other approaches with ten representations. For QoE, the higher the better. For other metrics, the lower the better.

performance with different representation sets by using its original training methodology.

We should also note that evaluating algorithms for different QoE models such as encoding bitrate based models can potentially change ABRs' QoE in Figure 7. However, such change may not present the perceptual QoE of real users as explained in Section 4.1 and cannot help with their energy savings. Our experiments support our motivation that sustainable solutions are possible for designing ABRs when perceptual quality and energy savings are considered to form QoE. Furthermore, when considering modern representation sets of streaming applications [5, 7, 9, 25, 27, 40], sustainable solutions become crucial to prevent using extra power and data for no additional QoE gain.

4.2.3 Device Capacity Violations. In this experiment, we use only Galaxy S4 since XCover Pro does not introduce any capacity violations against our representation sets (Table 1). Figure 9 shows that GreenABR outperforms all other ABRs by causing near-zero violations. This is expected because we intentionally trained our ABR agent not to choose the highest configurations for minor quality improvements. The other ABRs greedily pick the highest resolution version with good network throughput but regardless of device capacities. In fact, when stuttering events happen due to capacity violations, users' perceptual quality can be highly impacted. For example, people using Galaxy S4 for video streaming have lower perceptual quality than people with XCover Pro. Including capacity violations into perceptual quality calculation for the Galaxy S4 phone can reduce the achieved QoE in Figure 7b for all ABRs but

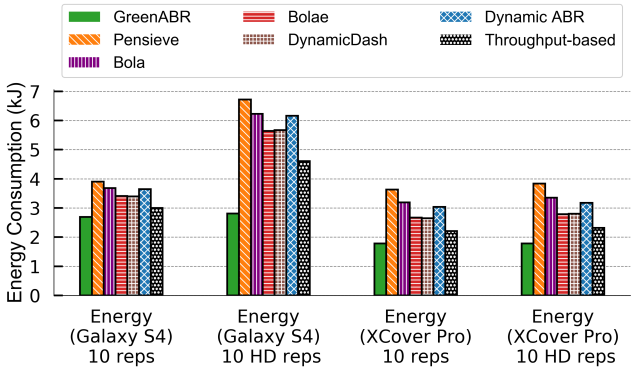


Figure 8: The energy consumption differences of Galaxy S4 and XCover Pro for different representation sets.

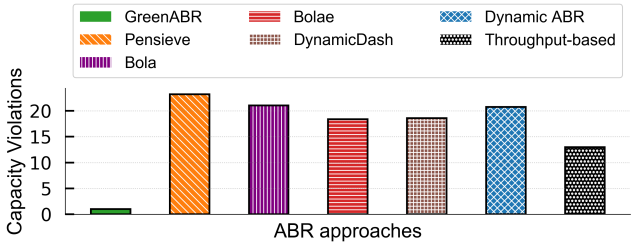


Figure 9: The average number of capacity violations of ABRs for all tested videos and network traces.

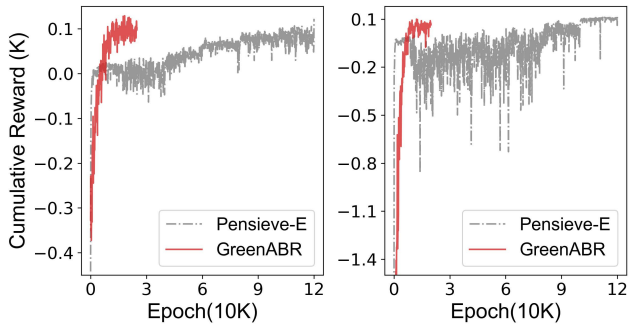


Figure 10: Training iteration comparison of GreenABR and Pensieve-E for 6 and 10 representations respectively.

impacts GreenABR much less than others. We should note that such impact is device-related, which raises the incompatibility issues of all ABRs without concerning device capacities. However, considering device specifications to design an ABR is not the goal of our work. GreenABR is an energy-efficient model which can generally be applied to today's video streaming scenarios in many mobile devices.

4.2.4 Training Performance Evaluation. In this experiment, we evaluate the training performance of Pensieve and GreenABR in terms of the cumulative reward and the required training epochs. To conduct a fair comparison, we use Pensieve-E instead of Pensieve because the former used the same reward function (Equation 3) as GreenABR. We trained Pensieve-E for 120,000 epochs by decreasing the entropy weight at every 20,000 epochs as recommended in the Pensieve [27] paper. Figure 10 shows that Pensieve-E and GreenABR learn similar cumulative rewards for both 6 and 10 representation sets. However, Pensieve-E requires more training epochs for stable performance, which is expected because policy-gradient-based RL approaches have high variance and usually encounter frequent fluctuations during training. After GreenABR converged to a targeted cumulative reward of approximately 100, we finished its training to avoid overfitting to the training video. GreenABR requires less than 25,000 epochs for the training and provides smoother learning than Pensieve-E, which reduces the training burden and eases hyperparameter tuning.

4.2.5 Energy Overhead of GreenABR. In this experiment, we measure the energy overhead of GreenABR on mobile devices. By design, GreenABR is trained on the server side in advance and transferred to the client along with the MPD file. The client is only required to invoke the model to get the selected bitrate at every chunk. Thus, to evaluate the energy consumption of GreenABR on the client, we deployed the trained model to an application and invoked it while measuring the power consumption. Using GreenABR incurred negligible power consumption compared with that for video streaming, which was around 1%.

5 CONCLUSION AND FUTURE WORK

Global mobile Internet traffic is dominated by video streaming, mainly served by the ABR streaming protocols over HTTP. ABR algorithms serve as the primary source of the user's QoE by selecting video representations. The selected ABR algorithm can also lead to significant power consumption differences and drain the mobile device's battery life. In this work, we proposed a deep RL-based ABR algorithm, GreenABR, that maximizes perceived QoE while minimizing the mobile device's consumed energy during video streaming. GreenABR employs a standard video quality metric, VMAF, that aligns with perceptual quality to define user QoE in terms of the video quality, rebuffering events, and smoothness of the streaming. We created a new power model based on our comprehensive measurements and used it for the training of GreenABR. Our experiments show that GreenABR outperforms the tested state-of-the-art ABRs with up to 57% saving in average energy consumption and up to 22% more perceptual QoE improvement. In detail, GreenABR excels competitors with up to 60% saving in data consumption, 84% less rebuffering, and near-zero capacity violations.

In future, we plan to improve the streaming server power efficiency due to encoding, storage, and replication. We will employ different codecs, video characteristics, and perceptual quality models to dynamically set the video representations at the chunk level. In this way, we aim to avoid unnecessary power consumption due to a limited number of representations for every video chunk. We plan to expand our work for a general ABR model that does not require additional training for different representation sets.

ACKNOWLEDGEMENTS

This project is in part sponsored by the National Science Foundation (NSF) under award numbers CCF-2007829, OAC-1842054 and OAC-1724898.

REFERENCES

[1] 2020. Cisco Visual Networking Index: Forecast and Trends, 2017-2022. Retrieved October1, 2022 from https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf

[2] 2020. DASH Industry Forum. Retrieved October 1, 2022 from https://dashif.org/

[3] 2020. DATASET: HSDPA-bandwidth logs for mobile HTTP streaming scenarios. http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs/

[4] 2020. Deep Q Network vs Policy Gradients - An Experiment on VizDoom with Keras. Retrieved October 1, 2022 from https://flyyufelix.github.io/2017/10/12/dqn-vs-pg.html

[5] 2020. Mezzanine requirements. Retrieved December 19, 2020 from https://videodirect.amazon.com/home/help?topicId=G202129880#G202129950

[6] 2020. Monsoon High Voltage Power Monitor. Retrieved August 19, 2020 from https://www.msoon.com/online-store/High-Voltage-Power-Monitor-Part-Number-AAA10F-p90002590

[7] 2020. Per-Title Encode Optimization. Retrieved August 19, 2020 from https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2

[8] 2020. Raw Data - Measuring Broadband America 2016. Retrieved August 19, 2020 from https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016

[9] 2020. Recommended upload encoding settings. Retrieved December 19, 2020 from https://support.google.com/youtube/answer/1722171?hl=en

[10] 2021. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022.* Retrieved October 8, 2021 from https://s3.amazonaws.com/media.mediapost.com/uploads/CiscoForecast.pdf

[11] 2022. GreenABR: Energy Aware Adaptive Video Streaming with Deep Reinforcement Learning. Retrieved March 25, 2022 from https:https://github.com/bekiroguzhan/GreenABR-MMSys22

[12] 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* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 44–58.

[13] Christos G. Bampis, Zhi Li, Ioannis Katsavounidis, Te-Yuan Huang, Chaitanya Ekanadham, and Alan C. Bovik. 2018. Towards Perceptually Optimized End-to-end Adaptive Video Streaming. arXiv:1808.03898 [eess.IV]

[14] Netflix Technology Blog. 2020. *VMAF: The Journey Continues.* Retrieved September 15, 2020 from https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12

[15] T. Breitbach, P. Sanders, and D. Schultes. 2018. Optimizing energy consumption and user experience in a mobile video streaming scenario. In *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*. 1–9.

[16] X. Chen, T. Tan, and G. Cao. 2019. Energy-Aware and Context-Aware Video Streaming on Smartphones. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 861–870.

[17] Zhengfang Duanmu, Abdul Rehman, and Zhou Wang. 2018. A Quality-of-Experience Database for Adaptive Video Streaming. *IEEE Transactions on Broadcasting* 64, 2 (2018), 474–487.

[18] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. 2004. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *J. Mach. Learn. Res.* 5 (Dec. 2004), 1471–1530.

[19] C. Herglotz, S. Coulombe, C. Vazquez, A. Vakili, A. Kaup, and J. Grenier. 2020. Power Modeling for Video Streaming Applications on Mobile Devices. *IEEE Access* 8 (2020), 70234–70244.

[20] Tianchi Huang, Chao Zhou, Rui-Xiao Zhang, Chenglei Wu, Xin Yao, and Lifeng Sun. 2019. Comyco: Quality-Aware Adaptive Video Streaming via Imitation Learning. *CoRR* abs/1908.02270 (2019). arXiv:1908.02270 http://arxiv.org/abs/1908.02270

[21] 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* (Chicago, Illinois, USA). New York, NY, USA, 187–198.

[22] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE *(CoNEXT '12)*. New York, NY, USA, 97–108.

[23] S. Kim, H. Oh, and C. Kim. 2018. eff-HAS: Achieve higher efficiency in data and energy usage on dynamic adaptive streaming. *Journal of Communications and Networks* 20, 3 (2018), 325–342.

[24] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR* abs/1612.00796 (2016). arXiv:1612.00796 http://arxiv.org/abs/1612.00796

[25] Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic Adaptive Streaming over HTTP Dataset. In *Proceedings of the 3rd Multimedia Systems Conference* (Chapel Hill, North Carolina) *(MMSys '12)*. 89–94.

[26] Yao Liu, Sujit Dey, Fatih Ulupinar, Michael Luby, and Yinian Mao. 2015. Deriving and Validating User Experience Model for DASH Video Streaming. *IEEE Transactions on Broadcasting* 61, 4 (2015), 651–665.

[27] 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* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[28] Jiayi Meng, Qiang Xu, and Y. Charlie Hu. 2021. Proactive Energy-Aware Adaptive Video Streaming on Mobile Devices. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 303–316.

[29] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937. https://arxiv.org/abs/1602.01783

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR* abs/1312.5602 (2013). arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (02 2015), 529–33.

[32] A. Mondal, B. Palit, S. Khandelia, N. Pal, J. Jayatheerthan, K. Paul, N. Ganguly, and S. Chakraborty. 2020. EnDASH - A Mobility Adapted Energy Efficient ABR Video Streaming for Cellular Networks. In *2020 IFIP Networking Conference (Networking)*. 127–135.

[33] Yanyuan Qin, Shuai Hao, Krishna R. Pattipati, Feng Qian, Subhabrata Sen, Bing Wang, and Chaoqun Yue. 2019. Quality-Aware Strategies for Optimizing ABR Video Streaming QoE and Reducing Data Usage. In *Proceedings of the 10th ACM Multimedia Systems Conference* (Amherst, Massachusetts) *(MMSys '19)*. Association for Computing Machinery, New York, NY, USA, 189–200. https://doi.org/10.1145/3304109.3306231

[34] Yanyuan Qin, Chinmaey Shende, Cheonjin Park, Subhabrata Sen, and Bing Wang. 2021. *DataPlanner: Data-Budget Driven Approach to Resource-Efficient ABR Streaming*. Association for Computing Machinery, New York, NY, USA, 94–107.

[35] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials* 17, 1 (2015), 469–492.

[36] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 123–137.

[37] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.

[38] Li Sun, Ramanujan K. Sheshadri, Wei Zheng, and Dimitrios Koutsonikolas. 2014. Modeling WiFi Active Power/Energy Consumption in Smartphones. In *2014 IEEE 34th International Conference on Distributed Computing Systems*. 41–51. https://doi.org/10.1109/ICDCS.2014.13

[39] 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* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 272–285. https://doi.org/10.1145/2934872.2934898

[40] Babak Taraghi, Abdelhak Bentaleb, Christian Timmerer, Roger Zimmermann, and Hermann Hellwagner. 2021. Understanding Quality of Experience of Heuristic-based HTTP Adaptive Bitrate Algorithms. (2021).

[41] M. Uitto and M. Forsell. 2018. Towards Energy-Efficient Adaptive Mpeg-Dash Streaming Using Hevc. In *2018 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 1–6.

[42] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.

[43] Hado van Hasselt, Arthur Guez, Matteo Hessel, and David Silver. 2016. Learning functions across many orders of magnitudes. *CoRR* abs/1602.07714 (2016). arXiv:1602.07714 http://arxiv.org/abs/1602.07714

[44] B. Varghese, G. Jourjon, K. Thilakarathne, and A. Seneviratne. 2017. e-DASH: Modelling an energy-aware DASH player. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 1–9.

[45] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham M. Kakade, Igor Mordatch, and Pieter Abbeel. 2018. Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[46] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 495–511. https://www.usenix.org/conference/nsdi20/presentation/yan "https://www.usenix.org/system/files/nsdi20-paper-yan.pdf".

[47] Chaoqun Yue, Subhabrata Sen, Bing Wang, Yanyuan Qin, and Feng Qian. 2020. Energy Considerations for ABR Video Streaming to Smartphones: Measurements, Models and Insights. https://dl.acm.org/doi/10.1145/3339825.3391867

[48] Yasir Zaki, Thomas Pötsch, Jay Chen, Lakshminarayanan Subramanian, and Carmelita Görg. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 509–522. https://doi.org/10.1145/2785956.2787498

[49] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.

A POWER MODEL TRAINING

A.1 Abstract

The proposed power model estimates the power consumption pattern for local playback component by using the normalized streaming attributes of videos. The details of the training methodology and model details are given in Section 3.1.3.

A.2 Artifact Checklist

- **Algorithm:** Linear Regression
- **Dataset:** Collected power measurements during streaming sessions
- **Metrics:** Root mean squared error(rmse)
- **How much time is needed to complete experiments (approximately)?:** Three hours
- **DOI:**10.5281/zenodo.6402904

A.3 Description

A linear regression model to estimate the power consumption between 0 and 1 as 1 is the highest.

A.3.1 How delivered. The dataset for the power consumption measurements and the training code is available under "power_model" folder.

A.3.2 Software dependencies. Below is the list of libraries needed for training and saving the model.

- python version = 3.7.3
- Keras version = 2.3.1
- numpy version = 1.16.4
- pandas version = 0.24.2
- scikit-learn version = 0.21.2
- joblib version = 1.0.1

A.4 Installation

To install the required libraries:

```
python setup.py
```

A.5 Experiment workflow

To train the power model, run the below command under "power_model" folder.

```
python train.py
```

The script trains the model, saves it and prints the evaluation results.

A.6 Evaluation and expected result

For the training dataset, the model performs rmse less than 0.01 and for evaluation dataset it produces rmse 0.036.

B GREENABR

B.1 Abstract

GreenABR proposes an energy-aware ABR model designed by using deep reinforcement learning. The training methodology and details of the model are explained in Section 3.2.

B.2 Artifact Checklist

- **Algorithm:** DQN
- **Dataset:** Power attributes dataset and VMAF measurements
- **Metrics:** QoE based on Equation 3.
- **How much time is needed to complete experiments (approximately)?:** Eight hours
- **Code licenses (if publicly available)?:BSD-2-Clause**
- **DOI:**10.5281/zenodo.6402904

B.3 Description

GreenABR proposes energy aware ABR decisions for HTTP streaming. It requires the power model to be trained in advance. It uses the number of representations as the action space of the RL model, thus requires separate training for six and ten representations case.

B.3.1 How delivered. All training and evaluation files and required measurement data are available under "GreenABR" folder in our public code repository [11]. The training scripts are available for each representation set separately.

B.3.2 Software dependencies. Below is the list of libraries needed for training and saving the model.

- python version = 3.7.3
- Keras version = 2.3.1
- numpy version = 1.16.4
- pandas version = 0.24.2
- scikit-learn version = 0.21.2
- joblib version = 1.0.1
- matplotlib version = 3.1.0

B.4 Installation

To install the required libraries:

```
python setup.py
```

B.5 Experiment workflow

To train GreenABR, run the below command under "rep_6" and "rep_10" folders for the corresponding representation sets.

```
python GreenABR.py
```

The script trains the model, saves it for every 1000 iterations and logs the average reward at each iteration. We found 9000 iterations to be optimal with the hyperparameter values as learning rate(α) = 0.0001, discount factor(γ) = 0.99, initial ϵ = 1.0, and $\epsilon - decay$ = 0.9995 to satisfy enough exploration during training. We set the experience replay memory size to store the most recent 500000 steps while updating the target network at every 100 steps.

B.6 Evaluation and expected result

To evaluate GreenABR for any representation set, copy the pre-trained model of the same representation set along with the power model. All required source files are provided under "evaluation" folder. To generate the streaming logs of the tested videos for GreenABR:

```
python evaluate.py
```

GreenABR is compared with several SOTA models and their streaming logs are generated by using their testing simulators [27, 36]. Results are stored under "test_results" folder for all algorithms. To plot the graphs, run:

```
python create_summary_results.py
python plot_graphs.py
```

for each representation set under the corresponding folder.

C STANDARD QOE MODEL TRAINING

C.1 Abstract

Comparing ABRs designed for different goals are not trivial and may lead to misleading results in terms of real users perception. To enable fair comparisons, we designed a standard QoE model based on SQoE-III [17] dataset which is a large dataset with subjective scores of real users.

C.2 Artifact Checklist

- **Algorithm:** Linear Regression
- **Dataset:** SQoE-III [17]
- **Metrics:** Spearman Correlation Score
- **How much time is needed to complete experiments (approximately)?:** 1 hours
- **DOI:**10.5281/zenodo.6402904

C.3 Description

Our model uses five significant components described in existing quality of experience studies for video streaming. It uses a linear regression model to maximize the Spearman correlation score between the estimated QoE scores and the real users' mean opinion scores.

C.3.1 How delivered. The dataset files and the training code are available under "standard_qoe_model" folder.

C.3.2 Software dependencies. Below is the list of libraries needed for training and saving the model.

- python version = 3.7.3
- numpy version = 1.16.4
- pandas version = 0.24.2
- scikit-learn version = 0.21.2

C.4 Installation

To install the required libraries:

```
python setup.py
```

C.5 Experiment workflow

To train the QoE model, find the coefficients:

```
python train.py
```

The script trains the model and prints the coefficients along with the Spearman score of the model.

C.6 Evaluation and expected result

We compared our model with proposed QoE models in Pensieve, Comyco, and the sample model in the dataset. Our model provides the highest score with 0.7845. To compare with other QoE models:

```
python compare_models.py
```