GreenABR+: Generalized Energy-Aware Adaptive Bitrate Streaming

BEKIR OGUZHAN TURKKAN, IBM Research, Yorktown Heights, USA University at Buffalo, Buffalo, USA
TING DAI, IBM Research, Yorktown Heights, USA
ADITHYA RAMAN, University at Buffalo, Buffalo, USA
TEVFIK KOSAR, University at Buffalo, Buffalo, USA
CHANGYOU CHEN, University at Buffalo, Buffalo, USA
MUHAMMED FATIH BULUT, Microsoft, Redmond, USA
JAROSLAW ZOLA, University at Buffalo, Buffalo, USA
DABY SOW, IBM Research, Yorktown Heights, USA

Adaptive bitrate (ABR) algorithms play a critical role in video streaming by making optimal bitrate decisions in dynamically changing network conditions to provide a high quality of experience (QoE) for users. However, most existing ABRs suffer from limitations such as predefined rules and incorrect assumptions about streaming parameters. They often prioritize higher bitrates and ignore the corresponding energy footprint, resulting in increased energy consumption, especially for mobile device users. Additionally, most ABR algorithms do not consider perceived quality, leading to suboptimal user experience. This article proposes a novel ABR scheme called GreenABR+, which utilizes deep reinforcement learning to optimize energy consumption during video streaming while maintaining high user QoE. Unlike existing rule-based ABR algorithms, GreenABR+ makes no assumptions about video settings or the streaming environment. GreenABR+ model works on different video representation sets and can adapt to dynamically changing conditions in a wide range of network scenarios. Our experiments demonstrate that GreenABR+ outperforms state-of-the-art ABR algorithms by saving up to 57% in streaming energy consumption and 57% in data consumption while providing up to 25% more perceptual QoE due to up to 87% less rebuffering time and near-zero capacity violations. The generalization and dynamic adaptability make GreenABR+ a flexible solution for energy-efficient ABR optimization.

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

Additional Key Words and Phrases: Video streaming, energy efficiency, deep reinforcement learning

This project is in part sponsored by the National Science Foundation (NSF) under award numbers CCF-2007829 and OAC-2313061.

Authors' addresses: B. O. Turkkan, IBM Research, 1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA, University at Buffalo, 338 Davis Hall, Buffalo, NY 14260, USA; e-mail: bekirogu@buffalo.edu; T. Dai and D. Sow, IBM Research, 1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA; e-mails: ting.dai@ibm.com, sowdaby@us.ibm.com; A. Raman, T. Kosar, C. Chen, and J. Zola, University at Buffalo, 338 Davis Hall, Buffalo, NY 14260, USA; e-mails: araman5@buffalo.edu, tkosar@buffalo.edu, changyou@buffalo.edu, jzola@buffalo.edu; M. Fatih Bulut, Microsoft, 14820 NE 36th St, Redmond, WA 98052, USA; e-mail: mfbulut@us.ibm.com.

ACM 1551-6865/2024/08-ART269
https://doi.org/10.1145/3649898

ACM Reference Format:
Bekir Oguzhan Turkkan, Ting Dai, Adithya Raman, Tevfik Kosar, Changyou Chen, Muhammed Fatih Bulut, Jaroslaw Zola, and Daby Sow. 2024. GreenABR+: Generalized Energy-Aware Adaptive Bitrate Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 9, Article 269 (August 2024), 24 pages. https://doi.org/10.1145/3649898

1 INTRODUCTION

Global internet traffic continues to experience tremendous growth and has reached a 23% compound annual growth rate in 2022. Video streaming, in particular, followed a growing trend with a 24% increase in 2022 and comprised 65% of global internet traffic [11].

Dynamic Adaptive Streaming over HTTP (**DASH**) [3] is one of the leading technologies responsible for this growing trend. DASH encodes each video at different bitrates, quality levels, and resolutions and stores these versions in equal-sized segments. Depending on the dynamically changing network conditions, the DASH client relies on an **adaptive bitrate** (**ABR**) algorithm to select the most suitable representation for the next segment.

Most of the existing ABR algorithms [12, 24, 30, 39, 41, 55] share three main objectives: (1) selecting the highest bitrate available, (2) minimizing the number of stalling events due to depleted client buffers, and (3) minimizing the oscillations in bitrate selections. Although these objectives are essential for providing a satisfactory user experience, the **quality of experience** (**QoE**) models employed by these algorithms do not take into account *perceptual quality*, which is crucial for accurately representing the streaming quality perceived by users. This is particularly significant for mobile users since the increase in bitrate becomes indistinguishable after a certain threshold due to the limited screen size. Considering perceptual quality enables more energy-efficient decisions by avoiding higher bitrates that do not contribute to an increase in perceived quality.

Heuristic-based ABR algorithms (e.g., Sabre [39], BOLA [41], MPC [55], and Taraghi et al. [44]) use predefined rules based on specific streaming parameters (e.g., buffer level, throughput information, and recent video representation) and are limited in accurately predicting future behavior. To address this issue, learning-based algorithms have been developed. For instance, CS2P [43] employs a Hidden-Markov Model to learn network characteristics from historical data, while Pensieve [30] uses a policy-gradient Reinforcement Learning method to optimize ABR decisions under various network conditions. Comyco [22], on the other hand, proposes an Imitation Learning model that selects video representations with expert policies. However, these approaches overlook either video perceptual qualities or energy consumption and often require more training iterations or computation. Furthermore, they commonly fail to generalize across different video representation sets, requiring additional training and hyperparameter tuning.

While much research has focused on improving the QoE of users in video streaming, relatively little attention has been given to improving the energy efficiency of ABR algorithms. Prior work has proposed hardware-related solutions, such as adjusting screen brightness [50] or turning off the NIC during idle time [25]. However, these approaches operate on top of ABR algorithms and do not directly influence video representation selection during streaming. Recently, researchers have explored context-aware and machine learning-based approaches to address this challenge. For example, a context-aware model was proposed to adjust bitrate selections based on device vibration levels [16], and EnDASH employs machine learning techniques to dynamically estimate optimal buffer levels and bitrate selections [34]. Additionally, an energy-aware adaptation scheme has been proposed to maximize QoE under a predefined power budget for 360° videos [32]. However, these approaches have limitations, which we discuss further in Section 2.

We present GreenABR+, a power-efficient ABR algorithm that maintains high perceptual quality while minimizing power consumption. GreenABR+ extends GreenABR [45] to work with

different representation sets, eliminating the need for additional training. Our contributions are as follows:

— We introduce a generalized energy-efficient ABR solution that utilizes **Deep Deterministic Policy Gradient** (**DDPG**) learning to make intelligent decisions to deliver high perceptual quality, reduce rebuffering events and quality oscillations, and conserve energy. Our model sustains stable performance across different video representation sets without additional training.
— We present a power model that captures the power consumption patterns of real video streaming sessions with varying encoding parameters and video characteristics.
— We develop a novel QoE calculation method based on the SQoE-III dataset [18]. Our approach considers five key components of QoE: video quality, rebuffering duration, stalling events, quality oscillation, and quality level changes. We use VMAF as the quality metric, which closely aligns with users' perceptions. Our model surpasses existing approaches in approximating real users' subjective scores.
— We compare GreenABR+ with state-of-the-art video streaming approaches. Our results demonstrate that GreenABR+ surpasses all other ABRs by saving up to 57% in streaming energy consumption and 57% in data consumption while achieving up to 25% more perceptual QoE. This is due to up to 87% less rebuffering time and near-zero capacity violations. In terms of energy efficiency (achieved QoE per unit energy consumption), GreenABR+ outperforms all other methods by up to 56%.

The rest of the article 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, compares GreenABR+ to other state-of-the-art ABR algorithms and generalization performance of the GreenABR+; and Section 5 concludes the article.

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 [3]. DASH enables streaming applications to design customized ABR algorithms for their specific needs. Early ABR algorithms, such as **Buffer-Based** (**BB**) [23], 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 [41], Festive [24], MPC [55], BOLAE [40], Throughput-based [40], DynamicDash [40], and Dynamic ABR [40].

BOLA [41] prioritizes the buffer occupancy level to avoid stalling events and target a minimum buffer level throughout the streaming. In this regard, it starts with the lowest available bitrate and quickly fills up the client buffer. After the buffer threshold is satisfied, it uses the available bandwidth to select the highest available bitrate. BOLAE [40] improves BOLA [41] with additional rules on the available bandwidth. Festive [24], MPC [55], and CS2P [43] employ throughput estimation techniques to decide the highest bitrate that would not cause stalling. Dynamic ABR [40] and DynamicDash [40] 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 [30] proposes an RL-based approach to target the overall QoE. Oboe [12] employs self-tuning algorithms to adapt runtime parameters for different network conditions dynamically. However, those approaches model the video quality linearly with the encoding bitrate. Similarly, Fugu [52] 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 [56], which falls short of modeling real user perceptions [2, 18]. Lately, Comyco [22] 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 applications have a significant impact on mobile devices' battery life. Previous work employed hardware-related solutions to save streaming power consumption. For example, e-DASH [50] dynamically adjusts screen brightness with different video contents to save battery life, while eff-HAS [25] turns off network connections when a predefined buffer level is reached. Uitto et al. [47] use HEVC codec to stream videos, which can achieve similar qualities compared with other codecs but with less data transfer and less power consumption.

Chen et al. [16] consider both video bitrate and vibration levels to optimize their ABR algorithm. He proposes to avoid higher bitrate decisions under high vibration levels since vibrations may affect the QoE of users. eff-HAS [25] studies user-preference history, predicts user retention time based on the video content, and uses a lower bitrate to stream the first few chunks of those videos with lower user retention. The above algorithms are tailored to specific use-cases [16], provided as an additional feature [25, 50], require hardware support [47], or they do not consider videos' perceptual quality in daily use, thus not achieving power savings for general purposes.

Recent work EnDASH [34] 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 available bandwidth may be significantly smaller during rush hours than at other times of the day. Likewise, Meng et al. [32] 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 play only part of the video. QUAD [35] and DataPlanner [36] propose models to maximize the QoE for a targetted quality level and data consumption budget, respectively. 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] present a quadratic function that estimates average power consumption during streaming with an active network channel by using the encoding bitrate of the video and the network signal strength. Additionally, they propose a linear function for local playback power consumption without network communication. Herglotz et al. [20] 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. [53] categorize streaming power consumption in CPU, display, network, and residual power. They propose a separate model for each component for regular and 360° videos. They state that video consumption is highly affected by device differences. However, all the above approaches target estimating the power consumption value, including non-ABR-related components, instead of capturing the energy pattern for ABR streaming. Thus, these tailored models may perform poorly for unseen devices and mislead the learning agents for training models.

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. Furthermore, ABR algorithms developed with prior training, like Pensieve [30], Comyco [22], and Oboe [12], assume only a limited number of video representations. However, real streaming applications like Netflix use a wider range of representations for the served videos [8]. Similarly, they use different representation sets based on their videos' popularity and genre. Therefore, learning models adapting to different representation sets are required in real-world video streaming scenarios. However, models trained for a fixed set of representations require retraining to adapt to these changes. In addition, policy-based models like **Asynchronous Advantage Actor-Critic** (**A3C**) [33] used in Pensieve are known to be sensitive to the training parameters and require tuning hyperparameter for each training as evaluated in our training performance experiments in Section 4.2.5. Existing studies use DDPG for networking optimization[21, 54] and the energy efficiency of the base stations during video streaming [28]. However, they do not consider adaptive HTTP streaming scenarios with flexible representation sets.

High bitrate may lead to high video quality. However, this relation is not linear when the perceptual quality is considered with different video contents, mobile devices, and distances between users and devices [2]. 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 into 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 energy savings motivate our work on GreenABR+.

3 GREENABR+ DESIGN

GreenABR+ extends the GreenABR model to ensure stable performance over different representation sets without requiring additional training. Both models target achieving high perceptual quality while saving energy consumption. In this regard, they consider video streaming as an RL problem and train their models with no assumptions about the network and streaming conditions. They use the same energy model and perceptual quality metric for their training. However, they differ in training models, requirements, and generalization performance. Specifically, GreenABR+ is trained for a broad bitrate range rather than discrete bitrate levels that enable supporting multiple encoding ladders without additional training. In contrast, GreenABR requires a distinct model for each representation set and incurs more computing costs. Considering hyperparameter tuning needs for each training, GreenABR may lead to more computing costs to ensure stable performance.

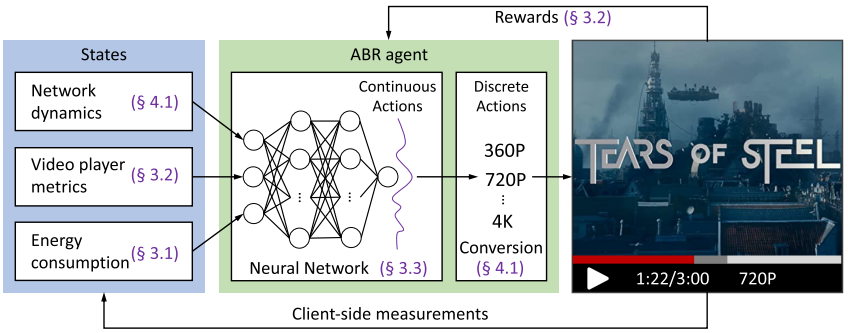


Fig. 1. The overview of GreenABR+.

In this section, we first describe the components of modeling sustainable video streaming. Subsequently, we explain how we consider video streaming as a reinforcement learning problem. Finally, we discuss the design and training details of GreenABR+.

3.1 Modeling Sustainable Video Streaming

Designing sustainable video streaming solutions has two essential goals: maximizing users' QoE and minimizing energy consumption. Existing approaches commonly prioritize either energy or quality at the cost of the other due to their conflicting nature. However, an efficient solution requires optimizing for both goals by finding the optimal decisions for the energy-quality trade-offs. Due to the complexity of this optimization problem, static rule-based models commonly fall short. Learning-based models, which consider energy consumption metrics, network dynamics, and video player metrics, can efficiently find optimal decisions.

Similarly, using perceptual quality helps to estimate users' QoE more accurately to avoid unnecessary data and energy consumption. Figure 1 shows the overall architecture of the energy-aware solution that we employ in GreenABR and GreenABR+ models. We formulate video streaming as a reinforcement learning problem, as explained in Section 3.2, to find bitrate decisions achieving substantial energy savings while not sacrificing any QoE. We include energy, streaming, and network parameters in our state space and use a standard perceptual quality metric, VMAF, in our reward signal calculations.

Energy consumption information is vital to guide the learning agent for efficient decisions. Collecting this information manually for each video and different representation sets is not scalable. Similarly, it involves device-specific parameters that can vary significantly among different devices. We present an energy model to estimate energy consumption patterns to overcome this limited approach.

Energy Model. Previous work [20] has shown five essential components contributing to the overall energy consumption for video streaming: *data acquisition*, *video processing*, *displaying video files*, *audio processing*, and *speaker operations*. Our model includes the first three components' consumption as they are the most relevant to ABR decisions. We model them in two parts: *data acquisition* and *local playback*, which includes video processing and display. 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 the video player's operation power without streaming. The resultant model accurately estimates energy consumption behavior for different video representations to guide the learning agent. For the

local playback component, we build a regression model to capture the energy consumption by training a feed-forward neural network with one input layer, two hidden layers, and one output layer. We use related encoding parameters and video characteristics, including *encoding bitrate*, *resolution*, *frame rate*, *file size*, *motion rate*, and *quality measurements* (i.e., VMAF) as our inputs and estimate the energy consumption as the output. For our dataset, we collected hours of energy consumption measurements of real streaming sessions. For training, we randomly split the dataset as 80% training and 20% testing and trained the model until it converged with an RMSE less than 0.01, corresponding 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, the Samsung XCover Pro. Our model is accurate, with RMSE less than 0.036 (about 12% estimation error) for the new device. For data acquisition, we adopt the existing throughput-based model [42] to calculate the data acquisition energy:

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

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. Our experiments use the recommended values [42] for TCP transfer as $\alpha = 210$ and $\beta = 28$. We utilize this energy model to train both the GreenABR and GreenABR+ models. Further details of our energy model are explained in our earlier work, GreenABR [45].

3.2 Solving Video Streaming with Reinforcement Learning

Why RL for ABR? ABR algorithms target maximizing the overall video streaming QoE by selecting the appropriate version of video chunks to adapt streaming environments dynamically, e.g., network throughput, client buffer size, and 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.

RL-based approaches aim to learn an optimal policy by maximizing the expected cumulative reward for an agent interacting with an unknown environment. Designing an ABR algorithm with RL techniques 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 energy 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.

States. For each chunk of video, the 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 energy 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 into chunks, each with multiple representations for adaptive selections. All different representations form the action space. A video representation contains the following parameters: *encoding bitrate*, *resolution*, *frame rate*, and *codec*. Typically, all representations share the same frame rate and codec for the same video. 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. Specifically, we fix the resolution for certain bitrate levels and use the same resolution for bitrate levels in between. Models using discrete actions require a fixed set of video representations, as in GreenABR. On the other hand, models with flexible representation sets, such as GreenABR+, can use a bitrate range and consider any value from the given range as a different representation.

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

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

where $\alpha * \mathtt{VMAF}_i$ is the video quality, $2^{\alpha * \mathtt{VMAF}_i - \beta}$ is the quality amplifier, $r_p * rt_i$ is the rebuffering penalty, $\alpha * |\mathtt{VMAF}_i - \mathtt{VMAF}_{i-1}|$ 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, $\mathtt{VMAF}_i$ is the video multimethod assessment fusion metric [13] for 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[1]. We set $\alpha = 0.05$, $\beta = 3$, $\gamma = 0.001$, and $\zeta = 2$. We use two sign functions $\lfloor \frac{\mathrm{sgn}\,(\alpha * \mathtt{VMAF}_i - \beta) + 1}{2} \rfloor$ and $\lfloor \frac{\mathrm{sgn}\,(\gamma * E_i - \zeta) + 1}{2} \rfloor$ to make sure that the quality and energy penalty amplifiers work only when the corresponding quality and penalty metrics $\alpha * \mathtt{VMAF}_i$ and $\gamma * E_i$ exceed their thresholds β and ζ. Otherwise, the amplifiers have no effects with a value of 0.

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 hyperparameter tuning [49], 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 $\alpha * \mathtt{VMAF}_i$ range from 0 to 5 since VMAF's range is [0, 100] [13]. Likewise, α is applied to the 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 1,000 to 4,500 millijoules.

We set the quality threshold $\beta = 3$. Videos with quality levels from 2 to 3 are considered to have fair qualities [2]. The quality amplifier $2^{\alpha * \mathtt{VMAF}_i - \beta}$ rewards exponentially for fair, good, and excellent qualities with quality levels as [2, 3), [3, 4), and [4, 5), respectively. High-quality videos always come with a high encoding bitrate, as shown in Figure 2. However, such a relationship is not linear. Once encoding bitrate reaches a threshold, increasing bitrate no longer increases perceptual quality [2]. For example, in Figure 2, after the encoding bitrate of 2.4 Mbps, the perceptual video quality ranges from 4.5 to 5, which is usually indistinguishable by

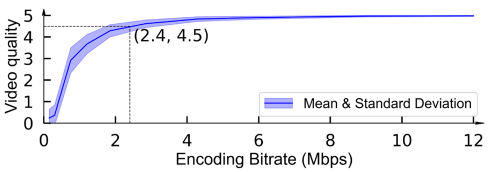


Fig. 2. Video perceptual quality ($\alpha * \mathtt{VMAF}$) to encoding bitrate in phone model measurements.

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

human eyes. However, the increment of encoding bitrate might increase other video parameters, such as resolution, which can significantly boost energy consumption.

To avoid our models 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 have observed that videos with a quality level of 4.5 consume 2 joules on average. Pursuing higher quality than 4.5 with an even higher bitrate or resolution does not increase overall perceptual quality but instead consumes more unnecessary energy. For example, as shown by the top right shaded area in Figure 2, aggressively choosing a higher bitrate than 2.5 Mbps with already high video quality is undesired. Our energy penalty amplifier punishes such behavior exponentially, which statistically guides our models to make energy-efficient choices.

We should note that the above reward function is used for our training to guide our agent for more energy-efficient decisions. Still, we define a more standard QoE calculation model in Section 4 to estimate users' QoE regardless of energy consumption and to compare with other ABRs fairly.

Modeling video streaming for fixed video representation sets. HTTP streaming incorporates the media representation file that contains a pre-defined set of video representations. This forms a static set of actions for the RL-based model. Two popular RL techniques, DQN-based methods and policy-gradient-based methods such as A2C, can learn the ABR decisions, achieving high rewards. Although policy-gradient-based methods are more general than DQN-based models, they are typically more challenging to train due to the high variance in gradient estimation [5, 19, 51]. By contrast, DQN-based approaches do not suffer from such a problem. Due to this merit, their training is typically more stable and requires 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 easily adapt to different client environments, i.e., mobile devices. In our earlier model, GreenABR, we employ the DQN-based method and show its benefits in our experiments in Section 4.2.4.

3.3 GreenABR+ Model

Modeling video streaming for flexible representation sets. Video streaming applications commonly use different encoding ladders [1] for different videos based on their popularity and genres. Therefore, the ABR algorithm should support different video representation sets. Although using discrete action spaces in DQN-based and policy-gradient-based approaches [30, 45] is efficient in learning ABR decisions, their retraining overhead is significantly large due to the changes in action space. Furthermore, changing action space can inevitably impact the model's performance since the model's hyperparameters were tuned for one action space but used for another. To mitigate such performance impact, hyperparameters must be re-tuned for the corresponding action spaces. It means more training time and power consumption for the new model training.

Our earlier work, GreenABR, is an energy-efficient video streaming model designed with the above-defined states, fixed actions, energy model, and reward calculation. Although it provides substantial energy savings while achieving high QoE, it requires new training for each representation set and may require tuning hyperparameters for the new set. GreenABR+, on the other hand, extends the GreenABR approach using DDPG learning that helps overcome the burden of retraining models. It proposes training a single model for a bitrate range and using the same model with a simple *continuous-to-discrete conversion* for different video representation sets. Therefore, it avoids the expensive re-training caused by hyperparameter tuning. It shares the same state space, including the energy component and the reward function in Equation (2) with the GreenABR, and uses DDPG learning for the agent.

The advantage of our DDPG-based model over DQN-based and policy-gradient-based methods will be demonstrated in the experiments.

DDPG is designed for continuous action spaces, which adapts the policy-gradient method from Actor-Critic models and adapts the target network and the replay buffer structures from Q-Learning models [27]. It learns a Q-function using off-policy data and the Bellman equation while learning a policy with the deterministic policy gradient algorithm and the learned Q-function. For the above reasons, we use DDPG to build our ABR agent, shown in Figure 3. We use linear approximation over a large representation set to make our streaming environment work for continuous action space. Specifically, we approximated the values of chunk data size and VMAF linearly for uncovered bitrates. The linear approximation works for chunk data size calculation since the encoding bitrate already decides it. Considering the non-linear relation of bitrate and VMAF values in Figure 2, we use more bitrate levels for the range of 0-3 Mbps to make VMAF estimation more accurate. The higher encoding bitrates become relatively insignificant for VMAF values, and linear approximation accurately calculates the VMAF values for them. The other state variables are calculated in the same way for both GreenABR and GreenABR+ models.

ALGORITHM 1: GreenABR+ training algorithm: DDPG algorithm with $\epsilon - greedy$ approach [27].

1 Initialize replay buffer $\mathcal{D}$;
2 Initialize critic network $Q(s, a|\theta^Q)$ and actor policy $\mu(s|\theta^\mu)$ with random weights θ^Q and θ^μ ;
3 Initialize target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$;
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 $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise ;
11 **end**
12 Take action a_t and observe QoE reward r_t (Equation 2) and state s_{t+1} ;
13 Store transition (s_t, a_t, r_t, s_{t+1}) to buffer $\mathcal{D}$;
14 Sample a random batch of transitions $(s_j, a_j, r_j, s_{j+1})_{j=1}^n$ from $\mathcal{D}$;
15 Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1, \\ r_j + \gamma Q'(s_{j+1}, \mu'(s_{j+1}|\theta^{\mu'})|\theta^{Q'}); & \text{otherwise.} \end{cases}$
16 Update critic by minimizing the loss: $\frac{1}{n}\sum_{j=1}^n (y_j - Q(s_j, a_j; \theta^Q))^2$;
17 Update the actor policy using the sampled policy gradient:
$$\nabla_{\theta^\mu} J \approx \frac{1}{n}\sum_{j=1}^{n} \nabla_a Q(s, a|\theta^Q)|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|s_j$$
18 Decay ϵ and update the target networks:
$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}, \quad \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
19 **end**
20 **end**

Training. Our ABR agent uses the recommended structure for DDPG [17, 27] with three-layer feed-forward neural networks for both actor and critic networks. Similarly, our target networks have the same structure as actor and critic networks, and we used the replay buffer for stable learning. We also used the $\epsilon - greedy$ approach to maintain efficient exploration of the action space.

Our training process is described in Algorithm 1. We randomly initialize our actor, critic, target networks, and replay buffer. We use the $\epsilon - greedy$ strategy and the Ornstein-Uhlenbeck process [46] to balance the *exploration-exploitation* tradeoffs. Specifically, starting with a high ϵ value, we obtain random actions (Line #6-8) to accelerate the exploration for early iterations. We decay the ϵ at the end of each iteration (Line #18) and exploit the actor-network for choosing actions (Line #10).

We use the Ornstein-Uhlenbeck process [46] to add noise to the selected actions (Line #10). It creates continuous action spaces [2] for the agent to explore for later iterations. We calculate the new

[2]Although we use continuous bitrate values during training, we use *continuous-to-discrete-conversion* for our evaluations with other ABRs.

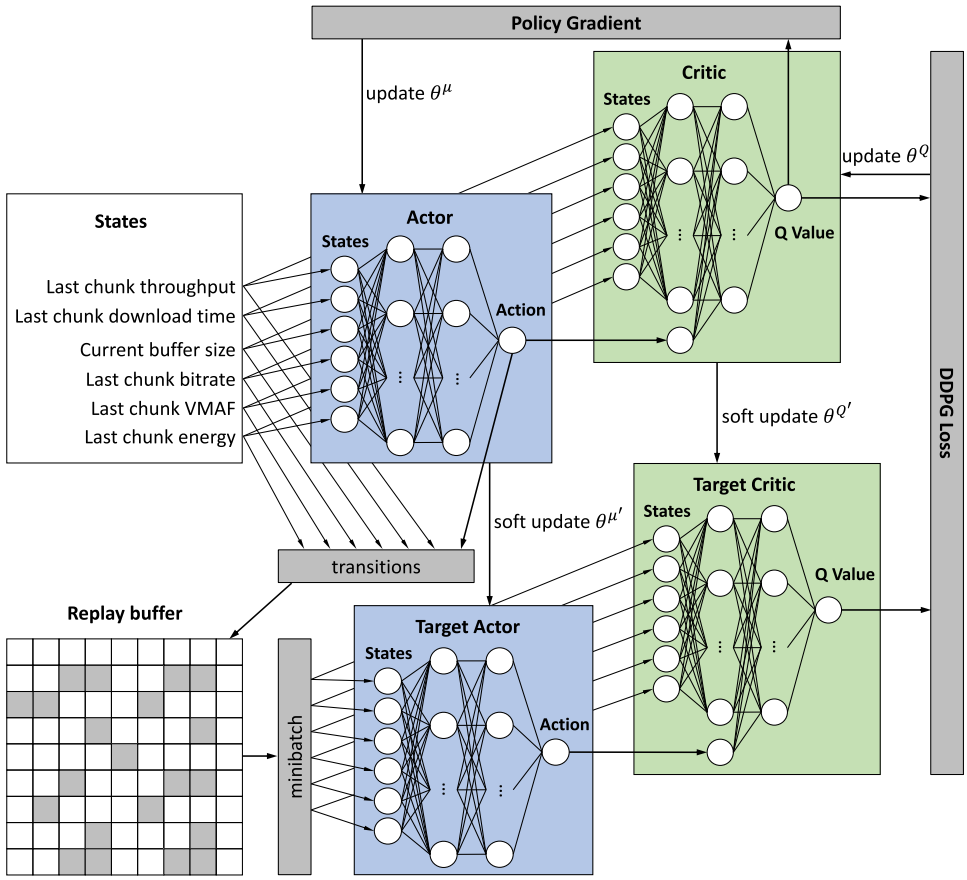


Fig. 3. The DDPG architecture used by GreenABR+.

state and the expected reward, then store the transition into the replay buffer $\mathcal{D}$ (Line #12-13). It sustains the stability of the training process, similar to the DQN algorithm [17].

To update the critic network's weight, we randomly sample a minibatch of transitions from the replay buffer (Line #14). The critic network is updated by minimizing the loss function:

$$\frac{1}{n}\sum_{j=1}^{n}(y_j - Q(s_j, a_j; \theta^Q))^2, \tag{3}$$

where y_j is the target value calculated by adding immediate reward r_j and expected discounted reward for the subsequent step $\gamma Q'(s_{j+1}, \mu'(s_{j+1}|\theta^{\mu'})|\theta^{Q'})$ with the discount factor γ.

To update the actor network, we optimize the policy parameter θ^{μ} with respect to the expected return (long-term cumulative reward) by gradient descent:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{n}\sum_{j=1}^{n} \nabla_a Q(s, a|\theta^Q)|_{s=s_j, a=\mu(s_j)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|s_j \tag{4}$$

Using the chain rule, we first take the gradient of Q with respect to the action a and then take the gradient of the deterministic policy function μ with respect to θ^{μ} (Line #17).

We use the target networks to avoid catastrophic forgetting [31] during training. In contrast to the DQN model, DDPG's target networks are slowly updated rather than copying weights at some frequencies (every C steps). The weights of these target networks are then updated by having them slowly track the learned networks:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \qquad \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{5}$$

where the change factor $\tau \ll 1$, meaning that the target values are constrained to change slowly, greatly improving the learning stability (Line #18).

4 EVALUATION

This section describes the evaluation methodology and experimental results. GreenABR+, including the energy 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 our power model data and compare the performances of different ABR algorithms. The three videos are "Big Buck Bunny", "Tears of Steel", and "Nature" videos in cartoon, sci-fi, and documentary genres, respectively. For all tested ABRs, we use the "Tears of Steel" video for training and all three videos for testing.

Action spaces. We encoded all the videos in our benchmark 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 [26, 30, 44] and industrial [6, 8, 10] recommendations. Since GreenABR+ is trained for continuous encoding bitrate levels, we use *continuous-to-discrete conversion* to decide the corresponding bitrate level in the available representation set. Specifically, we map the selected bitrate level to the nearest bitrate level in the representation set.

Table 1. Representation Sets for the Experiments

Action spaces			Resolution	Bitrate
6 reps	10 reps	10 HD reps		(Kbps)
	✓	✓	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

Energy data collection. In our experiments, we powered our phones with a Monsoon power meter [7] 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. 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 the energy consumption in each second by taking the mean value of the measured 5,000 (1second/200microsecond) power meter readings. 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, we include 3G traces from HSDPA [4], 4G/LTE traces from Belgium [48], and the FCC broadband traces [9] used in Pensieve and Sabre [39] simulators[3]. We have produced synthetic traces to complete our benchmarks, including non-stationary network behaviors such as temporal disconnection and network mode switch, which are not covered by the aforementioned network traces. 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. We selected a random throughput within the corresponding bandwidth range every second during each time interval. With random exploration, we captured fluctuated network conditions and changing network types. The distribution of the network traces and their bandwidth levels are shown in Table 2.

Table 2. Distribution of Network Traces

Trace Type	Bandwidth (Mbps)				Distr. Percent
	Avg	Min	Max	>12	
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%

We used fewer 4G traces since they have less diverse throughput levels with very high bandwidth for more than 10% of the time. Traces with more than 12 Mbps average bandwidth can be considered high bandwidth traces in our testbed since our highest encoding bitrate is 12 Mbps. We randomly selected 70% of traces of all network trace groups for training and used the rest for evaluations.

Compare with other ABR approaches. We compared GreenABR+ with our previous work, GreenABR, and state-of-the-art ABR schemes[4] including (1) *buffer* based approaches, i.e., BOLA and BOLAE, (2) *bandwidth* based approach, i.e., Throughput-based, (3) *buffer and bandwidth* based approaches, i.e., Dynamic ABR, DynamicDash, and (4) *reinforcement learning* based approach Pensieve. Although QUAD and Comyco are closely related to our work, we could not include them in our evaluations due to the lack of source code and very long and unstable learning for our ten-representation sets, respectively. We believe it is caused by the dynamic programming component used for expert behavior.

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 10 representations to show the high dynamics and broader range of real-world scenarios. The third set also contains 10 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 10 representations set to evaluate ABRs in broader action ranges without any hardware limit while using the second 10 representations set to evaluate ABRs' practicality and generality against capacity violations. The range of the used encoding bitrates is compatible with industry and academic standards [6, 8, 10, 18, 26, 44].

Perceptual quality calculation. Learning-based ABR models commonly include essential design components in their reward function, such as the energy penalty and the quality amplifier of GreenABR+, and train their models to maximize the achieved reward. They usually use the same reward function to calculate the QoE in their evaluations. However, included components such as

[3]Pensieve considers package headers while Sabre does not. We also modified the Sabre code to consider package headers to make a fair comparison.

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

energy consumption may not directly affect users' QoE. Similarly, ABRs trained with another reward function or ABRs with predefined rules may not consider these components 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 the number of quality switches to be the dominant components of perceptual QoE in the existing studies [18, 29, 38]. 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] [18]:

$$\begin{aligned} \texttt{QoE} := \alpha * \sum_{i=1}^{n} \texttt{VMAF}_\texttt{i} - \beta * \sum_{i=1}^{n} rt_i - \gamma * r_c - \sigma * \sum_{i=2}^{n} (|\texttt{VMAF}_\texttt{i} - \texttt{VMAF}_{\texttt{i}-1}|) \\ - \mu * \sum_{i=2}^{n} \lfloor (|\texttt{VMAF}_\texttt{i} - \texttt{VMAF}_{\texttt{i}-1}|)/20 \rfloor, \end{aligned} \tag{6}$$

where QoE is the quality of experience of a video streaming session, $\sum_{i=1}^{n} \texttt{VMAF}_\texttt{i}$ is the accumulated video quality, $\sum_{i=1}^{n} rt_i$ is the total rebuffering duration, r_c is the total number of rebuffering events, $\sum_{i=2}^{n} (|\texttt{VMAF}_\texttt{i} - \texttt{VMAF}_{\texttt{i}-1}|)$ is cumulative smoothness change, $\sum_{i=2}^{n} \lfloor (|\texttt{VMAF}_\texttt{i} - \texttt{VMAF}_{\texttt{i}-1}|)/20 \rfloor$ is the total number of quality switches, n is the total number of video chunks and $\alpha, \beta, \gamma, \sigma, and \mu$ are the coefficients. As suggested in [2, 35], 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 into 70% training and 30% testing. As suggested in [18], we repeated the training 1,000 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 [18]. 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 and 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 (2), 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.

Energy consumption calculation. We measure the overall energy consumption of each ABR by summing up their local playback energy and data acquisition energy. Specifically, we query our local playback *energy profile* for local consumption to get the actual measurements for the specific chunk. Our network traces set includes diverse network types and patterns, and measuring data energy consumption under these scenarios is not trivial. For instance, having the exact same throughput pattern for a 4G connection without interventions to the device's networking is impractical. Thus, we use Equation (1) for data acquisition consumption with the throughput

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

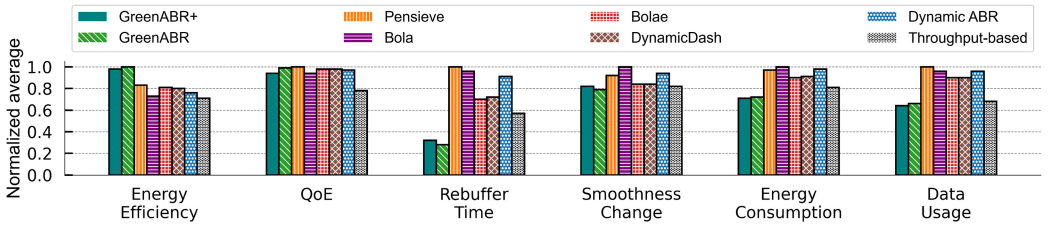


Fig. 4. Comparison of GreenABR+ with other approaches with six representations. For QoE and efficiency, the higher, the better. For other metrics, the lower, the better. Average results for all videos are used.

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

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

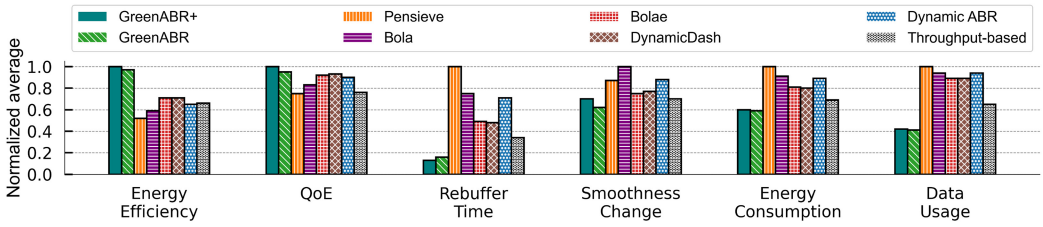
Energy efficiency calculation. Selecting higher resolutions requires more energy consumption while providing better QoE. We define the QoE per energy consumption as the energy efficiency that represents the QoE gain per unit energy consumption. It enables a fair comparison of ABR decisions for the energy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

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 followed the same methodology in our energy data collection and played the videos with our application. We logged the decoder outputs at each second during the playback and grouped them into video chunks by taking the total number of violations. Finally, we created 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.

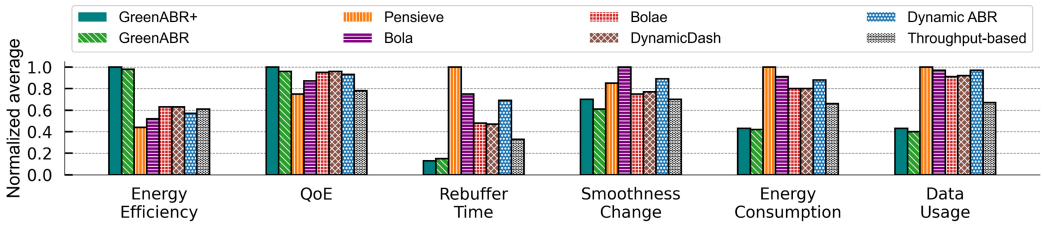
4.2 Evaluation Results

Figures 4 and 5 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 and GreenABR+ outperform all other ABRs regarding energy efficiency, with the lowest energy consumption and the smallest data usage while providing similar QoE. Since energy efficiency and energy consumption are the only device-specific categories and Galaxy S4 and XCover Pro have similar consumption patterns, we present combined results of both devices for these categories. We will describe the detailed comparison of different ABRs in the following subsections.

4.2.1 ABRs with Six Representations. GreenABR+ consumes the least energy and data among all ABRs while maintaining a comparable QoE to GreenABR (with only a 6% degradation), as



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



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

Fig. 5. Comparison of GreenABR with other approaches with 10 representations. For QoE, the higher, the better. For other metrics, the lower, the better.

shown in Figure 4. This matches our design, where our reward function (Equation 2) considers energy consumption and video quality while our ABR agent learns to make balanced quality-energy tradeoffs. GreenABR is the only model that performs slightly better than GreenABR+ in terms of energy efficiency while maintaining better QoE performance. This is attributed to the fine-tuning of GreenABR for the fixed six-representation set, whereas GreenABR+ is trained for the larger bitrate range. GreenABR+ consumes 44% to 68% less rebuffering time and 6% to 36% less data usage than other ABRs and performs similarly to GreenABR. Moreover, GreenABR+ attains 11% to 28% of energy savings compared to the other ABRs. Even though Throughput-based ABR has the third-lowest energy consumption after GreenABR+ and GreenABR, it still consumes 13% and 11% more energy and sacrifices overall perceptual quality with a 17% and 21% reduction compared with them, respectively. Overall, GreenABR+ achieves 16% to 28% better energy efficiency than other ABRs and demonstrates similar performance to GreenABR (only a 2% difference).

GreenABR+ and GreenABR provide slightly more 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 videos. To evaluate the generalization of our approach, we trained our model solely on the "Tears of Steel" video and included the "Big Buck Bunny" and "Nature" videos for testing. Our experiments indicate very similar results for all three videos, and we utilize the average values for all our evaluations.

4.2.2 ABRs with 10 Representations. Figure 5 shows that the benefit of our sustainable approach is significantly enlarged in two 10-representation sets. Specifically, GreenABR+ achieves up to 48% better energy efficiency, 13% to 40% energy savings, and up to 25% better QoE.

GreenABR achieves the second-best energy efficiency (3% less than GreenABR+) with 15% to 41% energy savings on average and up to 21% better QoE. The primary factor contributing to the energy efficiency improvement of both GreenABR+ and GreenABR is their ability to consume up

to 58% and 59% less data, leading to respective reductions of 87% to 84% in rebuffering time. Furthermore, improving the QoE gain of our models is not a compromise for energy savings. In fact, they both benefit from our sustainable approach with quality and energy considerations.

With the inclusion of 2K and 4K resolutions in Figure 5(b), GreenABR+ achieves greater energy savings of 34% to 57% on average for both devices compared to other ABRs. In the meantime, it achieves 4% to 25% higher QoE than others, and sustained 37% to 56% better energy efficiency. GreenABR achieved similar performance with GreenABR+ with only 2% less energy efficiency and 4% less QoE degradation.

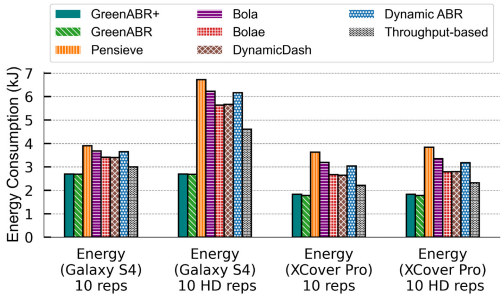


Fig. 6. The energy consumption differences of Galaxy S4 and XCover Pro for different representation sets.

In our experiments, we found that exceeding the decoding capacity of a device highly impacts energy consumption. Figure 6 presents the total energy consumption of Galaxy S4 and XCover Pro for both the 10 representation set and the 10-HD representation set in Table 1. Our results indicate that all ABRs except GreenABR+ and GreenABR consume at least 50% more energy on average for Galaxy S4 due to the incurred capacity violations. Meanwhile, the energy consumption of the XCover Pro for the 10-HD representation set is slightly higher than that for the 10-representation set, attributed to the inclusion of 2K and 4K resolutions, as it does not experience capacity violations. The resolution differences in the two representation sets mainly cause this slight increase. It validates our observation—aimlessly increasing resolutions can rapidly deplete the battery with more video processing.

Discussion: The performance degradation of Pensieve from Figure 4 to Figure 5 is the highest, mainly because the out-of-the-box hyperparameters in Pensieve are incompatible with new representation sets. We believe Pensieve's performance can be improved with hyperparameter tuning, different reward functions, or larger training iterations. To validate the above statement, we re-trained Pensieve with the reward function in Equation (2) and named it Pensieve-E. We compared Pensive-E with GreenABR. Even though Pensieve-E achieved slightly better (4%) QoE than GreenABR, it increased energy consumption significantly (27% to 45%). We never diminish the advantage of Pensieve, but we present the generalization performance with its original training methodology. We should also note that evaluating algorithms for different QoE models, such as encoding bitrate-based models, can change ABRs' QoE in Figure 5. 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. GreenABR+ and GreenABR have similar performances for both experiments. GreenABR achieves slightly better performance for the six-representations set, while GreenABR+ provides better efficiency and QoE for the 10-representations sets. We should mention that GreenABR+ is trained only once for the bitrate range from 0.15 to 12 Mbps, and we used the same model for all evaluations. In contrast, GreenABR is trained and tuned individually for each representation set. We further show the benefit of GreenABR+ in our generalization evaluations in Section 4.2.5.

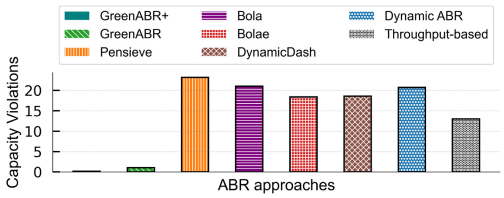


Fig. 7. The average number of capacity violations of ABRs for all tested videos and network traces.

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 [6, 8, 10, 26, 30, 44], 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 used only Galaxy S4 since XCover Pro does not introduce any capacity violations against our representation sets (Table 1). Figure 7 shows that GreenABR+ and GreenABR outperform all other ABRs by causing near-zero violations. This is expected because we intentionally trained our ABR agents not to choose the highest configurations for minor quality improvements. The other ABRs greedily pick the highest resolution version with good network throughput regardless of device capacities. Users' perceptual quality can be highly impacted when stuttering events happen due to capacity violations. 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 5(b) for all ABRs but impacts GreenABR+ and GreenABR much less than others. We should note that such an impact is device-related. However, considering device specifications to design an ABR is not the objective of this work.

4.2.4 Training Performance Evaluation. In this experiment, we evaluate the training performance of Pensieve, GreenABR, 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 2) as GreenABR and GreenABR+. Moreover, we compare them using the 10-reps set since GreenABR+ is trained on continuous encoding bitrates ranging from 150 Kbps to 12,000 Kbps, matching the same bitrate bounds of the 10-reps set.

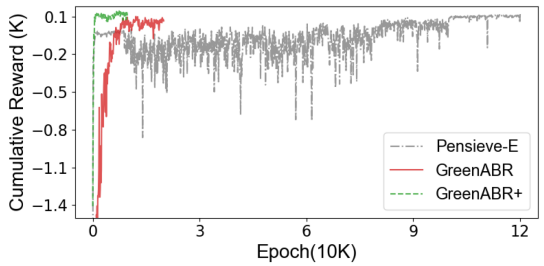


Fig. 8. Training performance comparison of GreenABR+, GreenABR, and Pensieve-E.

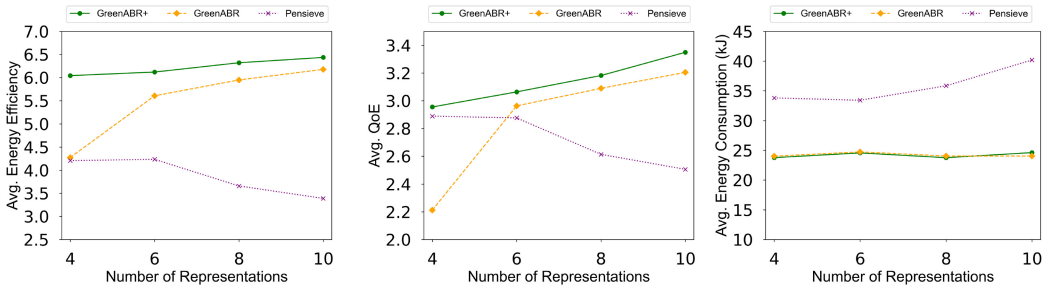
We trained Pensieve-E for 120,000 epochs by decreasing the entropy weight at every 20,000 epochs as recommended in Pensieve paper [30]. We finished the training of models when they converged to a targeted cumulative reward of approximately 100 to avoid overfitting to the training video.

Figure 8 shows that all models learn similar cumulative rewards. 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.

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 for retraining needs due to representation set changes. On the other hand, GreenABR+

Table 3. Generalization Evaluations Representations

Action spaces				Resolution	Bitrate
4 reps	6 reps	8 reps	10 reps		(Kbps)
		✓	✓	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



(a) Average energy efficiency. (b) Average QoE. (c) Average energy consumption.

Fig. 9. Comparison of GreenABR+, GreenABR, and Pensieve with the same trained models tested over different representation sets.

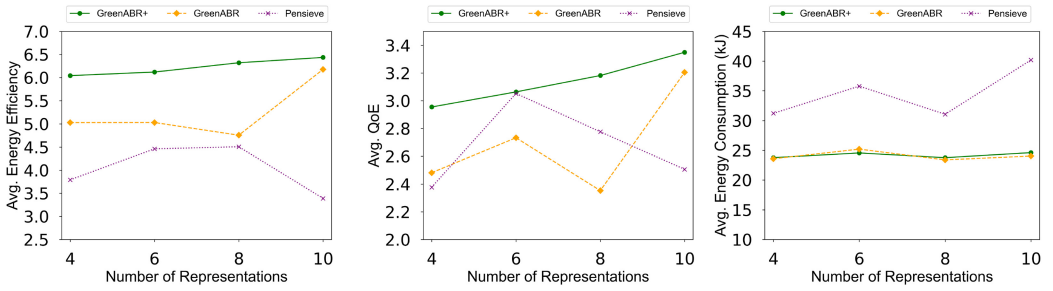
achieves the same performance with less than 10,000 iterations and less fluctuation during training. Using target networks and replay buffer helps stabilize the training. Including the $\epsilon - greedy$ strategy in our training accelerates the exploration and enables faster model convergence.

4.2.5 Generalization Evaluations. In this set of experiments, we compare the generalization of learning-based ABRs (i.e., GreenABR+, GreenABR, and Pensieve) over different representation sets, shown in Table 3. Since it is not applicable to the rule-based ABRs, we omit the others. To choose different representation sets, we first include the 10-reps set from Table 1 into our testbed. We then randomly choose 4, 6, and 8 bitrate levels in the 10-reps range to complete our testbed.

Testing one model over different representation sets. We trained GreenABR and Pensieve on the 10-reps in Table 3 and we trained GreenABR+ on a bitrate range of [150 Kbps, 12000 Kbps]. We test their trained models on all the representation sets in Table 3 by mapping the selected bitrates to the closest tested bitrate level. For instance, if a model selects a bitrate level of 1850 Kbps, we map it to the 1200 Kbps version for the 4-reps experiment since the selected version is not included.

Figure 9 shows that GreenABR+ achieves 4% to 47% higher energy efficiency for the 10-reps set compared to GreenABR and Pensieve, respectively. Both GreenABR and Pensieve show unstable results when evaluated with other sets. GreenABR degrades its performance up to 31% when the model is tested with the 4-reps set. On the other hand, Pensieve improves its efficiency up to 23% for the 4-reps. It shows that the recommended hyperparameters in the original Pensieve model are more suitable for the low number of representation sets. In contrast, the GreenABR+ model keeps up its performance with only 7% degradation towards the 4-reps set as shown in Figure 9(a). Missing some low bitrate representations causes this deficit with increased rebuffering time. Figure 9(c) demonstrates a similar pattern for the achieved QoE results. GreenABR+ preserves its performance by up to 7% difference among all representation sets, while GreenABR and Pensieve shift by up to 31% and 15%, respectively. GreenABR+ and GreenABR consume a similar amount of energy for all representation sets with around a 4% difference, while Pensieve's energy consumption shifts up to 17%. It shows QoE mainly decides the performance of GreenABR+ and GreenABR, while QoE and energy consumption both impact Pensieve's performance. We should note that we provide the next chunk size information for all training representations to Pensieve. If the provided chunk size information is limited to the testing set, its performance degrades drastically and causes negative QoE performance, especially for 4-reps and 6-reps sets. In practice, this information is available for only the representation sets in the encoding ladder.

Impact of Retraining Models. In the second set of evaluations, we used the same GreenABR+ model since it does not need re-training for the same bitrate range level. On the other hand, we



(a) Average energy efficiency. (b) Average QoE. (c) Average energy consumption.

Fig. 10. Comparison of GreenABR+, with retrained GreenABR and Pensieve for each representation set.

trained GreenABR and Pensieve for each representation set separately to understand the benefit of retraining the models. We used the same hyperparameters that we used in 10-reps training. Figure 10(a) shows that separate training improves the performance of GreenABR for 4-reps and Pensieve for 6-reps and 8-reps, but it also degrades drastically for 8-reps for GreenABR and for 4-reps for Pensieve in terms of energy efficiency. Likewise, GreenABR achieves better QoE for 4-reps, and Pensieve does for 6-reps compared to the same model evaluations shown in Figure 10(b). However, they do not achieve stable performance for all representation sets. GreenABR+ outperforms GreenABR and Pensieve for all representation sets and provides stable performance. The energy consumption results are similar to the first set of experiments.

Discussion: Models trained with discrete action spaces, like GreenABR and Pensieve, may have degraded performance when used with representations other than the ones in the training set. Thus, model retraining is required whenever the action space is changed. Retraining models is costly due to the consumption of time, computational resources, energy, and the need for hyperparameter tuning. GreenABR+ saves significant training energy consumption, considering the retraining costs, since it does not require additional training for any representation sets that stay in the same bitrate range as the training bitrate range. The energy consumption details of training these models are discussed in Section 4.2.6.

4.2.6 Energy Consumption Evaluations. In this set of experiments, first, we evaluate the cost of training RL-based ABRs. Then, we measure the energy consumption of invoking our models on mobile devices. **Energy Overhead of Training ABR Models.** We measured the energy consumption of training RL-based models, GreenABR+ (DDPG), GreenABR (DQN), and Pensieve (A3C), using the same training environment and libraries. We used pyRAPL [37], developed as part of PowerAPI project [14], to measure the CPU power consumption during training. We repeat the experiments for 100, 200, 300, 400, 500, 1000, and 2000 iterations to understand the energy consumption behavior. Figure 11 indicates that training energy consumption increases linearly with the number of iterations for all models. It also shows GreenABR and GreenABR+ models have similar energy consumption for the same number

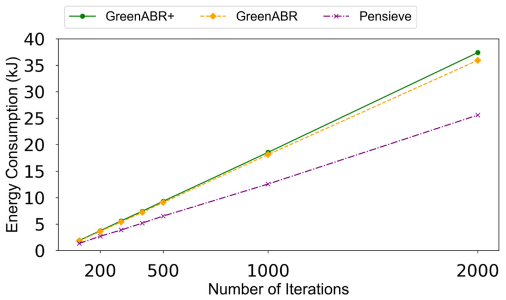


Fig. 11. Training energy consumption comparison of GreenABR+, GreenABR, and Pensieve.

of iterations. Pensieve consumes less power than other models since it does not require batch normalization during training compared to GreenABR and GreenABR+ models. However, it requires more iterations to converge and sustain stability, as Figure 8 demonstrates. Considering the number of iterations needed for convergence of these models, Pensieve requires more power consumption with ≈1500 kJ for 120,000 iterations, compared to GreenABR with ≈540 kJ with 30,000 iterations and GreenABR+ with ≈187 kJ for 10,000 iterations.

Energy Overhead of Using GreenABR Models. In this experiment, we measure the energy overhead of GreenABR+ and GreenABR on mobile devices. We deployed our models to an application and inferred the models at each second. We compared our measurements during inferring models with the base power consumption of the application. Our results from five runs on two devices indicate that our models do not introduce additional power consumption.

5 CONCLUSION

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 developed a new RL-based ABR model, GreenABR+, which inherits the GreenABR model to maximize perceived QoE while minimizing the mobile device's consumed energy during video streaming by using the power model component and perceptual quality metric, VMAF. GreenABR+ preserves stable performance in terms of energy efficiency and QoE over different representation sets without requiring additional training. It avoids the training cost due to action space changes and hyperparameter tuning and saves a significant amount of training energy consumption.

To compare our models with other SOTA ABR studies, we developed a new QoE calculation method with the components as video quality, rebuffering duration, number of rebuffering events, number of quality oscillations, and amount of smoothness changes by using a large dataset with real users' opinion scores. 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 25% more perceptual QoE improvement. In detail, GreenABR+ excels competitors with up to 59% saving in data consumption, 87% less rebuffering and near-zero capacity violations. GreenABR+ achieves more stable training performance and converges the same level of QoE while saving a significant amount of training time and cost.

In conclusion, sustainable ABR solutions enable saving a large amount of energy consumption for mobile streaming scenarios while maintaining high QoE. The generalization of these models is important for training efficiency and can be achieved by novel approaches using deep reinforcement learning techniques.

REFERENCES

[1] Netflix Technology Blog. 2016. Toward A Practical Perceptual Video Quality Metric. Retrieved August 19, 2020 from https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[2] Christos Bampis, Zhi Li, Ioannis Katsavounidis, Te-Yuan Huang, Chaitanya Ekanadham, and Alan Bovik. 2021. Towards perceptually optimized adaptive video streaming-A realistic quality of experience database. *IEEE Transactionson Image Processing*. (2021), 1–1. https://doi.org/10.1109/TIP.2021.3073294

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

[4] Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2012. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Trans. Multimedia Comput. Commun. Appl.* 8, 3, Article 24 (July 2012), 19 pages. https://doi.org/10.1145/2240136.2240137

[5] Felix Yu. 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

[6] Prime Video Direct. 2020. Mezzanine requirements. Retrieved December 19, 2020 from https://videodirect.amazon.com/home/help?topicId=G202129880#G202129950
[7] Monsoon Solutions Inc. 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
[8] Netflix Technology Blog. 2020. Per-Title Encode Optimization. Retrieved August 19, 2020 from https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2
[9] Federal Communications Commission. 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
[10] YouTube Help. 2020. Recommended upload encoding settings. Retrieved December 19, 2020 from https://support.google.com/youtube/answer/1722171?hl=en
[11] Sandvine. 2023. *Global Internet Phenomena*. Retrieved November 7, 2023 from https://www.sandvine.com/phenomena
[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 (SIGCOMM'18)* . New York, NY, USA, 44–58.
[13] Netflix Technology Blog. 2020. *VMAF: The Journey Continues*. Retrieved September 15, 2020 from https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12
[14] Aurelien Bourdon, Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. 2013. PowerAPI: A software library to monitor the energy consumed at the process-level. *ERCIM News* 2013 (2013). https://api.semanticscholar.org/CorpusID:1162790
[15] T. Breitbach, P. Sanders, and D. Schultes. 2018. Optimizing energy consumption and user experience in a mobile video streaming scenario. In *Proceedings of the 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 *Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 861–870.
[17] OpenAI Dcoumentation. 2018. Deep Deterministic Policy Gradient. Retrieved February 1, 2023 from https://spinningup.openai.com/en/latest/algorithms/ddpg.html
[18] 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.
[19] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5 (Dec. 2004), 1471–1530. ISSN: 1532-4435.
[20] 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. https://doi.org/10.1109/ACCESS.2020.2986580
[21] Ruying Hong, Qiwei Shen, Lei Zhang, and Jing Wang. 2019. Continuous bitrate & latency control with deep reinforcement learning for live video streaming. In *Proceedings of the 27th ACM International Conference on Multimedia (ACM MM'19*). ACM, New York, NY, USA, 2637–2641.
[22] 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 (Nice, France) (MM'19)*. Association for Computing Machinery, New York, NY, USA, 429–437. https://doi.org/10.1145/3343031.3351014
[23] 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 (SIGCOMM'14)* . New York, NY, USA, 187–198.
[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 (CoNEXT'12)*. New York, NY, USA, 97–108.
[25] 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.
[26] Stefan Lederer, Christopher Müller, and Christian Timmerer. 2012. Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd Multimedia Systems Conference (MMSys'12)*. 89–94.
[27] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR'16)*, San Juan, Puerto Rico, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1509.02971
[28] Dong Liu, Jianyu Zhao, Chenyang Yang, and Lajos Hanzo. 2021. Accelerating deep reinforcement learning with the aid of partial model: Energy-efficient predictive video streaming. *IEEE Transactions on Wireless Communications* 20, 6 (2021), 3734–3748.

[29] 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.

[30] 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 (SIGCOMM'17)* . ACM, New York, NY, USA, 197–210.

[31] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, Gordon H. Bower (Ed.). Vol. 24, Academic Press, 109–165. DOI: https://doi.org/10.1016/S0079-7421(08)60536-8

[32] Jiayi Meng, Qiang Xu, and Y. Charlie Hu. 2021. Proactive energy-aware adaptive video streaming on mobile devices. In *Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC'21)*. USENIX Association, 303–316.

[33] 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 *Proceedings of the International Conference on Machine Learning*. 1928–1937. Retrieved from https://arxiv.org/abs/1602.01783

[34] 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 *Proceedings of the 2020 IFIP Networking Conference (Networking)*. 127–135.

[35] 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 (ACM MMSys'19)* . New York, NY, USA, 189–200.

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

[37] Spirals research group. 2019. pyRAPL. Retrieved February, 2023 from https://github.com/powerapi-ng/pyRAPL

[38] 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.

[39] 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 (MMSys'18)*. ACM, New York, NY, USA, 123–137.

[40] 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 (MMSys'18)*. ACM, New York, NY, USA, 123–137.

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

[42] Li Sun, Ramanujan K. Sheshadri, Wei Zheng, and Dimitrios Koutsonikolas. 2014. Modeling WiFi active power/energy consumption in smartphones. In *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems*. 41–51. DOI: https://doi.org/10.1109/ICDCS.2014.13

[43] 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 (SIGCOMM'16)*. ACM, New York, NY, USA, 272–285. DOI: https://doi.org/10.1145/2934872.2934898

[44] Babak Taraghi, Abdelhak Bentaleb, Christian Timmerer, Roger Zimmermann, and Hermann Hellwagner. 2021. Understanding quality of experience of heuristic-based HTTP adaptive bitrate algorithms. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (Istanbul, Turkey) (NOSSDAV'21)*. Association for Computing Machinery, New York, NY, USA, 82–89. https://doi.org/10.1145/3458306.3458875

[45] 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 *Proceedings of the 13th ACM Multimedia Systems Conference (MMSys'22)*. New York, NY, USA. DOI: https://doi.org/10.1145/3524273.3528188

[46] G. E. Uhlenbeck and L. S. Ornstein. 1930. On the theory of the Brownian motion. *Physical Review* 36, 5 (Sep 1930), 823–841. DOI: https://doi.org/10.1103/PhysRev.36.823

[47] M. Uitto and M. Forsell. 2018. Towards energy-efficient adaptive Mpeg-Dash streaming using Hevc. In *Proceedings of the 2018 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 1–6.

[48] 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.

[49] Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (Barcelona, Spain) (NIPS'16)*. Curran Associates Inc., Red Hook, NY, USA, 4294–4302.

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

[51] 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 *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.

[52] 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 *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation (NSDI'20)*. 495–511.

[53] Chaoqun Yue, Subhabrata Sen, Bing Wang, Yanyuan Qin, and Feng Qian. 2020. Energy considerations for ABR video streaming to smartphones: measurements, models and insights. In *Proceedings of the 11th ACM Multimedia Systems Conference (Istanbul, Turkey) (MMSys'20)*. Association for Computing Machinery, New York, NY, USA, 153–165. https://doi.org/10.1145/3339825.3391867

[54] Won Joon Yun, Dohyun Kwon, Minseok Choi, Joongheon Kim, Giuseppe Caire, and Andreas F. Molisch. 2022. Quality-aware deep reinforcement learning for streaming in infrastructure-assisted connected vehicles. *IEEE Transactions on Vehicular Technology* 71, 2 (2022), 2002–2017. DOI: https://doi.org/10.1109/TVT.2021.3134457

[55] 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 (SIGCOMM'15)* . ACM, New York, NY, USA, 509–522. DOI: https://doi.org/10.1145/2785956.2787498

[56] 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.

Received 22 April 2023; revised 10 November 2023; accepted 9 February 2024