USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Predict, Prune, Play: Efficient Video Playback Optimization Under Device Diversity and Drift

Harsha Sharma, *Massachusetts Institute of Technology and Amazon;* Pouya Hamadanian and Arash Nasr-Esfahany, *Massachusetts Institute of Technology;* Zahaib Akhtar, *Amazon and North Carolina State University;* Mohammad Alizadeh, *Massachusetts Institute of Technology*

https://www.usenix.org/conference/nsdi26/presentation/sharma

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

Predict, Prune, Play: Efficient Video Playback Optimization Under Device Diversity and Drift

Harsha Sharma[†,‡,*], *Pouya Hamadanian*[†,*], *Arash Nasr-Esfahany*[†], *Zahaib Akhtar*[‡,¶], *Mohammad Alizadeh*[†]
[†]MIT, [‡]Amazon Prime Video, [¶]NCSU

Abstract

Video-streaming platforms tune dozens of playback parameters across thousands of client devices. Our measurements from Amazon Prime Video show that device-specific tuning can enhance stream quality. Yet traditional tuning techniques like Bayesian optimization become prohibitively expensive due to the large configuration space and the constant emergence of new device types.

We introduce AZEEM, a scalable recommendation system leveraging *few-shot prediction* to rapidly identify promising configurations for new devices. The key insight behind AZEEM is that devices exhibit performance similarities that enable predictions from limited observations. Trained on offline data of device-playback configuration interactions, AZEEM efficiently narrows down the search space to a small set of configurations likely to contain optimal or near-optimal candidates. Additionally, AZEEM addresses temporal distribution shift—where the best-performing configurations change over time—by recommending a small, robust set of candidates rather than a single configuration. Evaluations using large-scale real-world datasets show that AZEEM reduces exploration cost by $5.8-13.6\times$ and improves stream quality compared to state-of-the-art Bayesian optimization and multi-armed bandit approaches, enabling effective device-specific optimization at scale. We deploy AZEEM on a subset of Amazon Prime Video's production traffic, where it achieved a relative QoE improvement of 2.7% on average and 10.6% at the 90th percentile over an existing treatment tuning system.

1 Introduction

Video is the dominant form of Internet traffic, accounting for 68% of total traffic in 2023 [58]. To tap into this growing demand, large scale video platforms maximize the availability of their content by supporting a wide range of client devices, such as smart TVs, game consoles, phones/tablets, set-top boxes and more [20]. Unfortunately, the video device landscape is extremely fragmented with devices differing in their (i) hardware (CPU, memory, screen, networking, *etc.*) and (ii) software (operating system, firmware, application stack, *etc.*) and (iii) management (streaming protocol, software release cycles, media compatibility *etc.*). This fragmentation poses stiff challenges [2,4,16], as content platforms must deliver a consistent, high quality experience across all their supported devices.

While a significant body of work has focused on achieving high quality Internet video delivery by improving different components of the video pipeline [3,5,19,27,31–34,39,47,53,61,63,64,69,70], surprisingly, little to no research attention has been devoted towards dealing with this device heterogeneity systematically. We focus on this overlooked part of the video delivery ecosystem.

Through measurements from hundreds of millions of streaming sessions from Amazon Prime Video, we show that playback quality can be improved by tailoring video delivery not just to the network but also to the specific device (§2.1). We analyze different playback configurations (henceforth called *treatments*) which tune a range of components of video playback, including adaptive bitrate algorithms [3,5,19,31,47,61,69,70], video downloading strategies [30,46], and bitrate ladders [32,39,63,64]. Our results show that 90% of devices improve QoE by treatments specifically tailored for them.

However, to maximize quality across all devices, content platforms are currently left with traditional tuning techniques such as manual A/B testing, Bayesian optimization [14, 24, 29, 59, 60] or multi-armed bandits techniques [9,10,34,40]. Given the large and ever expanding configuration space and the constant onboarding of new devices, these techniques become prohibitively expensive for device-specific tuning. Traditional techniques explore the configuration space from scratch for each new device. Due to the high variance of video QoE metrics, it takes thousands of sessions worth of measurements per treatment to make a reliable assessment (§2.2). User-invasive outcomes are inevitable during this exploration. Moreover, since most devices receive a small slice of the overall traffic, it can take days to weeks to accumulate enough data per treatment, during which temporal distribution shifts can render previous measurements obsolete. Thus, compared to optimizing for coarser categories like network and video content, data-driven tuning for each device requires more efficient exploration strategies. Throughout this paper, "device" refers to a device type (e.g. SmartTV model family), not an individual physical unit.

We present AZEEM[1] which learns from past data and then leverages few-shot predictions to intelligently explore the large configuration space. Central to AZEEM is the key

[*]Equal contribution

[1]AZEEM stands for **A**n Optimi**Z**ed **E**xploration **E**xploitation **M**ethod.

insight that devices while diverse, exhibit behaviors that can be decomposed using a handful of *latent factors*, thus enabling predictions of how a device is likely to perform for an *unexplored treatment* through a small set of observations from *explored treatments*. AZEEM occasionally collects a dataset of device-treatment interactions for a small but diverse set of devices to train a prediction model. For a new device, it explores a few bootstrap treatments, predicts the QoE for other treatments, and prunes the search space. Compared to traditional techniques, AZEEM's exploration strategy avoids large regressions and reduces exploration significantly, without relying on crude spatial/temporal clustering or splitting of devices into traffic classes [27, 43].

A well-known challenge for data-driven techniques like AZEEM is the presence of temporal distribution shift which can make predictions stale. Our measurements show that the best-performing treatment for a given device changes over time. Distribution shifts can be attributed to the ever-evolving nature of the video delivery pipeline where iterative improvements get applied to individual components (*e.g.,* an upgraded CDN caching algorithm), making a once-optimal treatment to be superseded by another. AZEEM takes this into stride and instead of recommending a singular best treatment, it provides a small set of candidates (e.g., 5 treatments) that are robust to temporal shifts. By narrowing the search to a few promising candidates, AZEEM enables standard exploration-exploitation techniques to readily draw from a limited pool of strong treatments despite temporal distribution shifts. AZEEM selects its recommendation set by balancing two criteria: (i) high predicted QoE on the target device; (ii) reliability, quantified by a treatment's average performance over all devices. This regularizes the recommendation set and prevents overfitting to ephemeral patterns in measurement data.

Using data from Amazon Prime Video, we rigorously evaluate AZEEM. Our key findings are:

1. Across a wide battery of tests, AZEEM reduces exploration time by $3-6\times$ compared to baselines. In particular, to achieve the same gains as AZEEM, Bayesian optimization causes $13.6\times$ more QoE degradation.
2. AZEEM is robust to temporal distribution shifts and continues to longitudinally outperform baselines, regardless of exploration budget and deployment size.
3. AZEEM is especially effective in improving the tail QoE performance—largely caused by low performing devices or poor networks—delivering up to 41% improvement over Bayesian optimization.
4. In production deployment within Amazon Prime Video, AZEEM achieves a relative QoE improvement of 2.7% on average and 10.6% at the 90th percentile over an existing treatment tuning system. These improvements validate AZEEM efficacy, especially for tail performance.

Although AZEEM's implementation is unique to video delivery, the broader insights are likely to apply to instance-optimization in other large-scale networked systems (e.g., CDN tuning [27, 51, 71]). Our techniques rely on two key properties, (1) structural similarity among instances, and (2) presence of small, robust recommendation sets under temporal drift. We believe both properties are common in network instance-optimization tasks.

2 Motivation

Suppose a video service provider wishes to support a new playback device such as a new set-top box or smart TV—a process called *onboarding*. The provider aims to maximize the Quality of Experience (QoE) for this device's clients.

Delivering video to clients is a multi-step pipeline with many knobs to adjust for better QoE, *e.g.,* modifying the bitrate ladder [32, 39, 63, 64], video fragment downloader [30, 46], the Adaptive BitRate (ABR) algorithm [3, 5, 19, 31, 47, 61, 69, 70], *etc*. There is no single configuration of these knobs (*i.e.,* a treatment) that optimizes the experience for all clients. Rather, treatments must be specialized to the specific characteristics of the client—such as the plyaback device along with the network characteristics.

2.1 Benefits of device-specific tuning

The benefits of optimizing playback settings for different network characteristics are well known [3, 5, 44, 47, 53, 62, 63, 69]. However, most prior work ignores device characteristics and has not considered device-specific tuning.

Streaming devices exhibit widely different hardware (CPU, memory, screen size, *etc*.) and software (operating system, firmware, application stack, *etc*.) capabilities. Playback is affected by factors such as decoding time, available memory, thread parallelism, packet processing rate, *etc*. Even with the same network and video content, two devices can have noticeably different playback experiences and different optimal treatments. For further discussion on device and software peculiarities, see §A.1 in the Appendix.

To understand the interaction between playback settings and device, we collected 5 months of Amazon Prime Video streaming logs across 50 devices in different network conditions. These logs contain more than 750 million sessions which correspond to several hundred millions of streaming hours (number omitted for business concerns). We measured the performance of 84 different treatments across these devices in a randomized control trial (RCT). These treatments span a variety of playback settings, including ABR algorithms, ABR parameters, download parallelism, and customized bitrate ladder.

As one example of device heterogeneity, devices which run single threaded players (e.g. JavaScript Dash.js [26]) cannot easily benefit from techniques which use multi-thread parallelism (*e.g.,* concurrent fragment downloads). Table 1 shows the change in the percentage of sessions that encountered

Table 1: Relative difference in stall rate for various devices at 2 or 3 concurrent download requests.

	# of Concurrent Fragments	
Device	2 Fragments	3 Fragments
A	-0.5%	-0.5%
B	-0.3%	-0.4%
C	-0.2%	-0.2%
D	+0.1%	+0.3%
E	+1.8%	+3.0%
F	+7.0%	+11.4%

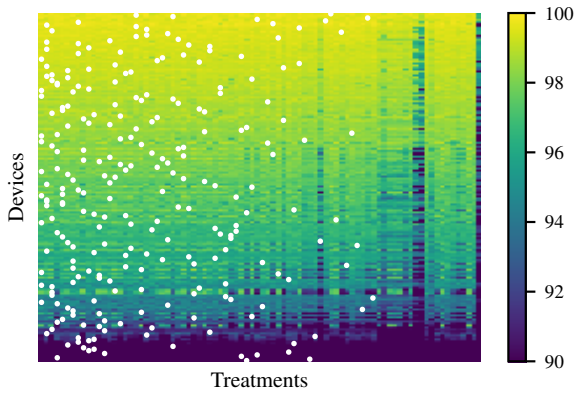


Figure 1: QoE Heatmap per each treatment and device in February 2025. The best treatment is denoted with a white dot.

rebuffers when the number of parallel download requests was increased from 1 to 3 for several device models. Although performance improves for devices A, B, and C with increased parallelism, devices D, E, and F get worse, with E and F experiencing a severe regression. Figure 16 (Appendix §A.1) shows the impact of download parallelism across all devices, with the same takeaway.

To get a more global view, Figure 1 shows a heatmap of the QoE achieved for each combination of treatment and device in the month of February 2025.[2] The QoE score is a composite metric between 0 to 100, with terms for rebuffering, video quality, startup delay, and quality smoothness (see §5.1 for details). For each device, we show the best treatment with a white dot. Treatments are sorted in decreasing order of the number of devices for which they are best. Although some treatments are the 'best' more frequently, the top treatment varies considerably across devices, specifically, 64 treatments (out of 84) are the best for at least one device.

To understand the room for improvement with device-specific tuning, we compare the QoE achieved by (1) the best treatment tuned for each device-network pair, and (2) the best treatment for network [3]. Figure 2 shows the CDF of the QoE improvement attained by the device-network pair specific tuning relative to the overall network based tuning. On more than 90% of devices, a device-specific treatment can be found which performs better than the 'overall-best' treatment in a given network cluster. The benefit varies across devices, with half of the devices gaining at least 25 basis points[3] of QoE and 10% gaining over 75 basis points. At scale, even improvements of a few basis points are significant and translate to better experience for millions of users.

2.2 What makes device-specific tuning hard?

The scale of the ecosystem is a considerable challenge for device-specific tuning. Video streaming services today support tens of device families [2, 20] spread across desktops, smartTVs, set-top boxes, game consoles and mobiles/tablets, *etc*. These individual device families (e.g. smartTVs) in turn contain devices from tens of manufacturers each bearing hundreds of specific models. As such, the number of distinct devices in a production setting surpasses several thousands. Furthermore new devices are being released constantly which need to be promptly onboarded while ensuring that legacy devices continue to be well supported.

With such scale, the only practical solution is *data-driven optimization*, i.e. automated exploration-exploitation techniques to search and find good treatments using measurements in production. Studying each device manually or catering software stacks on a per device level is not scalable, and is only reserved for special circumstances. Simulators do not capture device-specific behaviors while large-scale device farms are expensive to build and operate. Device capabilities and streaming technology evolves continuously, making it difficult to ensure adequate coverage of in-the-wild devices. We discuss alternatives to data-driven optimization and their challenges in more detail in §A.2.

Data-driven specialization of treatments for different network characteristics has been explored [3, 34] and is used in the industry [24, 55]. However, applying the same techniques to device-specific tunings poses two key challenges.

Exploration cost. Exploration-exploitation techniques consist of exploring different treatments followed by exploiting gained knowledge. Upon the arrival of a new device, such techniques deploy a set of treatments, measure the QoE, and build an internal model of treatment → QoE. This model is used to sample more treatments (up to an exploration budget), and eventually recommend a treatment for that device. Two prominent exploration-exploitation strategies studied in prior work (for network-level customization) are Bayesian optimization [24] and multi-armed bandits [34].

To measure the QoE for a treatment, we must accumulate enough data for that treatment. Video streaming metrics have

[2] To prevent confounding effects due to differences in network conditions and content types, we compute QoE for sessions belonging to each (device, network type, and content quality) separately. Thus, each row of the heatmap in Figure 1 corresponds to a (device, network type, content quality) tuple, known as a *cohort*. We omit discussing cohorts further in this section for simplicity; we revisit them in §4.1.1.

[3] In line with common practice for large video service providers, we report QoE improvements in basis points.

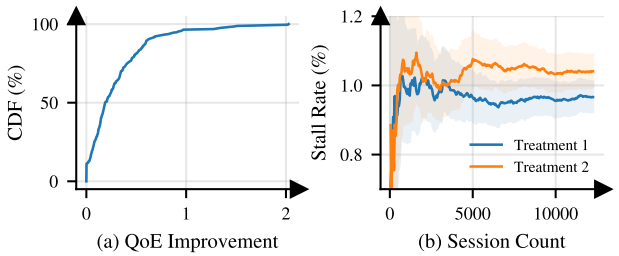


Figure 2: (a) QoE improvement from optimizing treatments per device compared to the 'overall-best' treatment that maximizes the average QoE across all devices. (b) Stall rate with 95% confidence intervals, against number of sessions for two treatments on a set-top-box device. Y-axis is normalized to the final value for Treatment 1.

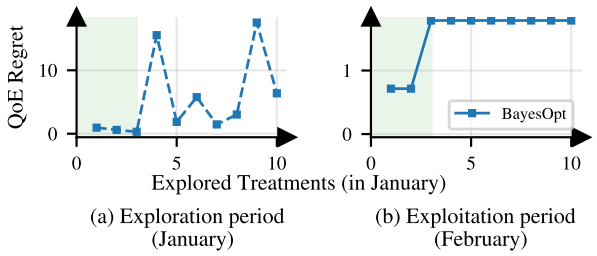


Figure 3: Device-specific tuning for a sample device using Bayesian optimization. (a) QoE regret of sampled treatments with respect to best treatment in Jan 2025 exploration. (b) QoE regret of the best January treatment in Feb 2025. Here, regret is with respect to the best treatment in Feb.

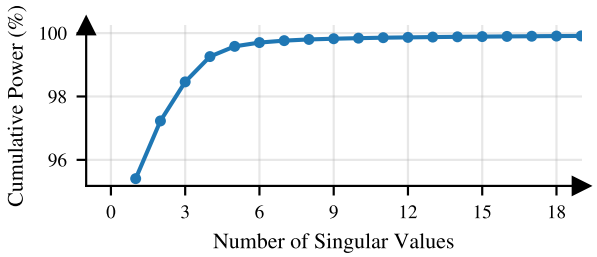


Figure 4: Cumulative energy plot of the QoE matrix.

heavy-tailed distributions [69]. The 'signal' for differences between treatments is typically at the tail—the small percentage of sessions where the issues that impact QoE occur (e.g., network or device anomalies). Therefore, we generally need to accumulate thousands of sessions before the variance in the measurements subsides and treatments separate. As an example, Figure 2 shows the evolution of stall rate measurements for an example device as we acquire more measurements for two treatments. The solid line is the measured (normalized) stall rate (percentage of sessions that experience rebuffering); the shades show the 95% confidence interval. Even after 10,000 sessions, the confidence intervals have a small overlap.

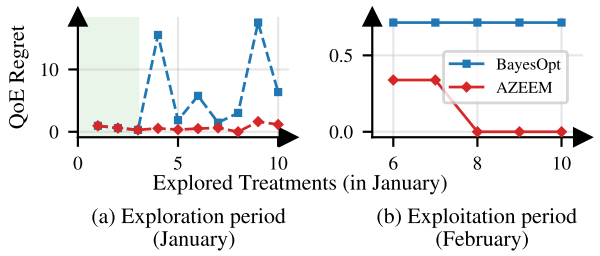
Figure 3 shows an example of this when using Bayesian optimization to tune treatments for a sample device in our dataset. We explore the treatment parameter space (8 parameters), with 84 total choices spaced 'well apart' in the configuration space. The exploration occurs with data collected during January 2025. Figure 3 shows the *QoE regret*—defined as the difference in QoE between the sampled treatment and the optimal treatment for the device in January. To bootstrap the exploration with promising candidates, we select the first 3 treatments (shaded region in the plot) to be top 3 treatments on average across all devices (except the target device) in January. After that, Bayesian optimization takes over and samples treatments. There are significant spikes in QoE regret as it explores the treatment space. This is expected, since the optimizer has no knowledge of expected QoE other than what it has sampled. The only way it can find better treatments is to sample good and bad ones in the process. In this example, it turns out that none of the sampled treatments are better than the third explored treatment, one of initial bootstrap treatments.

QoE drift. Knowledge gained in the exploration phase goes stale with time. Even if exploration cost was minimal and one could try all treatments to pick the best, the question is if the optimal choice during the exploration period will remain optimal in week(s) or month(s) ahead. The answer is no; treatment performance is a complex function of its parameters as well as a range of unobserved time-varying factors, *e.g.,* network/ISP performance, CDN performance, background traffic patterns, device firmware and software version *etc*. As these components evolve over time, so does the QoE and the performance of individual treatments.

These drifts can completely throw off any approach that tries to select a singluar 'best' treatment based on past data. Figure 3 shows an example. The plot shows the QoE regret in February 2025, when deploying the best treatment found after different number of explorations using Bayesian optimization in January 2025. Here, regret is with respect to the optimal treatment that maximizes QoE in February. The best of the first two treatments explored in January performs relatively well in February. However, the third explored treatment, which improved over the first two treatments in January, is much worse in February. Additional samples did not change the best treatment found in January, thus the QoE regret in February does not change after three explorations.

2.3 Key Insights

The main limitation of prior approaches is that they explore the treatment space from scratch for each new device. Even though devices are diverse, intuitively they share some common characteristics (e.g., similar hardware specifications, soft-



(a) Exploration period (January) (b) Exploitation period (February)

Figure 5: Comparison of Bayesian optimization and AZEEM ($k=5$ treatments). (a) QoE regret during Jan 2025 exploration phase. (b) QoE regret in Feb 2025 of the best recommended treatment based on Jan exploration. Here, regret is with respect to the best treatment in Feb

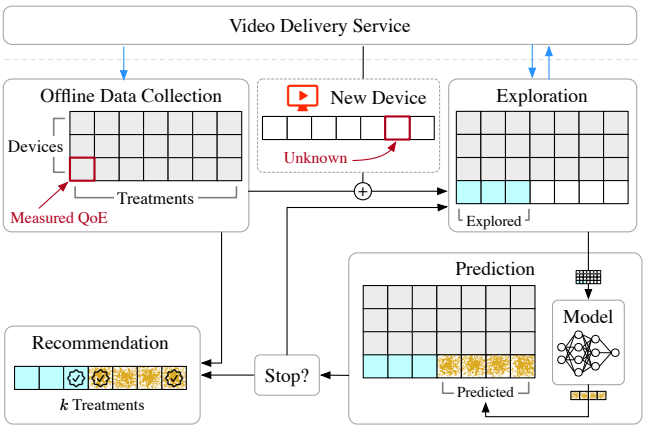


Figure 6: AZEEM's design overview. AZEEM onboards new devices in two phases of exploration and exploitation.

ware architecture, *etc*). They will not be identical; but it is also rare to encounter a device unlike any other on the market. This raises the question: *can we exploit similarities between devices to predict the behavior of unexplored treatments on a device?*

To gain a global view of device-treatment interactions, we can view QoEs in the form of a matrix, as in Figure 1. Each row denotes a device and each column denotes a treatment. Looking at Figure 1, there appear to be some patterns but it is difficult to visualize the similarity between devices. It turns out, however, that there is significant structure to this matrix, and it is "simpler" than it might appear. In particular, the matrix is very well approximated by a low-rank matrix. Figure 4 shows the sum of the top r singular values of the QoE matrix. The top $r=5$ singular values sum up to 99.5% of the cumulative energy of the matrix (sum of all singular values). This shows that the bulk of the "information" in the matrix is captured in the first 5 principal directions. The implication is that, even though the matrix includes 10s of rows (devices) and columns (treatments), each device and treatment can be categorized using a handful of *latent factors* that explain device-treatment interactions and the resulting QoE measurements well. Therefore, in principle, we should be able to categorize a device (i.e., identify its latent factors) based on the QoE of a small number of treatments on that device, and then predict the QoE for all other treatments. This is the goal of AZEEM's prediction model.

As discussed in §2.2, due to distribution shifts, recommending a single treatment per device is a recipe for failure. Our key insight to combat distribution shifts is that although the best per-device treatment changes with time, it is possible to select a small set of k treatments that have a high likelihood of including a strong treatment over time. For a small k (e.g., 5–8), we can hand these treatments off to live adaptation systems that, with continual monitoring, allocate a larger share of traffic to better treatments. For example, Pytheas uses a multi-armed bandit algorithm to hone in on the best treatment from an input set, and combats data drifts by giving more weight to recent measurements [34]. Such techniques struggle to explore the full treatment space, but with a small set of promising candidates, the problem becomes much simpler.

Figure 5 compares AZEEM to Bayesian optimization for the same device shown in §2.2. During the exploration period (January 2025), AZEEM avoids significant regressions. It does so by predicting the QoE of unseen treatments after observing the first three bootstrap treatments (shaded region), and pruning the poor treatments from its exploration. In the exploitation period (February 2025), AZEEM's top $k=5$ recommendations significantly outperform the top $k=5$ recommendations of Bayesian optimization, at any value for the number of explored treatments. Beyond 8 explored treatments (in January), AZEEM begins to recommend the treatment that is optimal in February for this device.

3 Design Overview

AZEEM comprises four main components, illustrated in Figure 6: *i)* offline data collection, *ii)* exploration, *iii)* prediction, and *iv)* recommendation. This section provides a brief overview of each component. Recall from §2.3 that AZEEM's goal is to select a set of k treatments for a new device, such that the set has a high likelihood of containing a strong treatment, all while minimizing exploration time. We describe how these components work together to realize this goal in detail in §4.

Offline data collection Before AZEEM can begin onboarding new devices, we select a fraction of current devices and deploy a RCT (randomized control trial) that evaluates QoE for all viable treatments. This data is later used to build a QoE prediction model for new devices.

Exploration. When a new device arrives for onboarding, AZEEM uses the offline collected data to guide exploration. It selects several treatments at a time and explores them on the

device. After enough measurements, it collects the streaming metrics and reports them to the prediction module.

Prediction. The prediction component trains a lightweight neural network on the offline collected data and the newly explored treatments on the new device. Using this prediction model, AZEEM fills in unexplored treatments with QoE estimates. This data may be handed off to the exploration component again for further exploration, or it will be used for the final recommendation.

Recommendation. This component uses all measured data and predictions to produce a set of k treatments for the new device, among which at least one is desirable. The selection strategy compensates for data drift and model overfitting by balancing model-based suggestions with suggestions solely based on the offline data for other devices.

4 Design Details

In this section, we discuss the inner workings of the components described in §3. Algorithm 1 depicts a pseudo-code of AZEEM, found in §B in the Appendix.

4.1 Formulation

For device i and treatment j, we denote the QoE measured with $qoe_{i,j}$.[4] We denote whether we have a measurement for device i and treatment j (have deployed it and measured it) with a mask variable $m_{i,j}$, which is 1 if measured and 0 otherwise.[5] We'll assume that we have $m-1$ RCT devices, indexed from 1 to $m-1$, and the new device has index m.

4.1.1 Cohorts and Treatments

Beyond device, the user's QoE is strongly correlated to the network conditions and highest content quality available for the video. Therefore, treatments must be tuned for each subgroup defined by device, network, and video quality. As such, we identify a cohort with index i with device d_i, network n_i and content quality q_i. We identify a treatment j with a set of parameters γ_j that uniquely determine what configuration, algorithm and hyperparameters are used.

Network cluster. We follow past work [3] and categorize users to different 'network clusters', defined by past bandwidth and latency measurements. Fundamentally, device and network cluster are separate, and a user of a particular device can be in any network cluster.

[4] We make no assumption on what QoE metric is used, and this approach is compatible with any scalar QoE metric.

[5] This variable is implicitly visualized in Figure 6, where shaded cells have $m_{i,j}=1$, and white cells have $m_{i,j}=0$.

Content type. Not all videos are available at the highest tier of quality, and not all devices are capable of playing highest quality levels (*e.g.,* Ultra-HD not supported on low end devices). Since the bottom-line QoE depends on this availability of both the original video and device, we separate cohorts by content type, which can be any of Ultra High-Definition (UHD), High-Definition (HD) or Standard Definition (SD).

Treatments. Treatments tune components of the playback stack such as fragment download concurrency, bitrate ladder rungs, choice of ABR algorithms and their parameters in line with past work [3, 30, 32, 39, 46, 63, 64]. Within these dimensions, treatments can differ between one or more values, *e.g.,* two treatments A and B could differ in the degree of download concurrency but be similar in all other dimensions. Finally, a total of 84 treatments were identified with the help of domain experts to maximize the chances of achieving optimal performance.

4.2 Offline Data Collection

Since we use data from a RCT to train a prediction model for new devices, the RCT should be representative of the devices faced in onboarding. We take care to select at least one device from each device family, but also ensure one device family does not dominate RCT devices. Since devices are often correlated with network tiers and content types, we also need to ensure selected devices cover the range of these metrics.

This RCT may need updating from time to time, but the cadence is infrequent (e.g., every 6 to 12 months §5.4). Another alternative to infrequent updates is allotting a small share of traffic to the RCT and running it on continual basis, however, such continual broad RCTs are costly. In our results, patterns from one RCT in January 2025 were enough for evaluations up to the end of April, which was the end of our dataset.

Why do we need an RCT? A fair question is why we can't use available measurements from onboarded devices instead of an RCT. Note that after onboarding, only a handful of treatments remain in deployment. While the QoE measurements of these columns (treatments) reveal some information, it can be misleading. As a hypothetical example, the partial outcome matrix in Figure 7 hints that treatments 1, 2 and 3 achieve better QoE compared to treatments 4, 5, and 6. If we viewed the full matrix in Figure 7 we would observe that the poor QoEs we observed for treatments 4, 5 and 6 were due to their devices. Such biased views make it difficult to *disentangle* the causal effects of devices and treatments. In early experiments, we tried to use non-RCT historical data, but predictions were poor and inaccurate. We leave the investigation of approaches that can debias partial matrices to future work.

4.3 Exploration

Bootstrap exploration. To collect some knowledge about a new device, we first explore a set of b treatments chosen

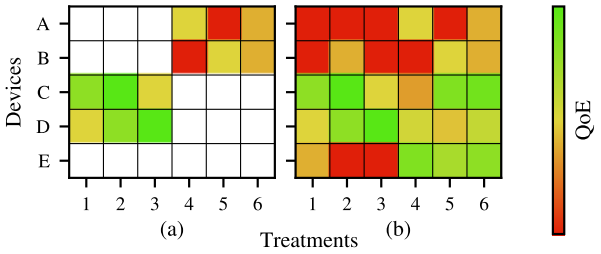


Figure 7: Experiments can be viewed as a matrix. Unobserved treatment is denoted in white. Full matrix **(b)** comparison with the partial matrix **(a)** shows how historical data can cause bias.

based on offline data. To choose this set, we find b treatments in the original RCT that maximize QoE across devices, if we keep only the best of these b treatments for each device:

$$BT_b := \underset{S\in[1..n]^b}{\text{argmax}} \sum_{i=1}^{m-1} max_{j'\in S} qoe_{i,j'}. \quad (1)$$

Where BT_b denotes the set of bootstrap treatments with size b. This is a default selection of treatments, oblivious to the new device.

BT_b is naturally diverse, *i.e.,* these treatments 'cover' the RCT devices well; we are not looking for b treatments that are all good for every device, but a set where at least one of the b treatments performs favorably. This procedure favors selecting unique treatments, each tailored to a subset of the RCT devices. Note that the alternative of just picking the top-b treatments in terms of average QoE over devices would not achieve this; most of the selections would be too similar.

Guided exploration. After the bootstrap exploration, AZEEM uses the bootstrap observations to first retrain its prediction model. In doing so, it becomes 'aware' of the new device that needs to be onboarded. It then kicks off subsequent rounds of exploration, to determine the treatments which are tailored to this new device. It begins by selecting the top $l-b$ treatments with the highest *predicted* QoE, where l is the total exploration budget.

Note that instead of exploring all $l-b$ in one round, AZEEM chooses to explore one predicted treatment at a time and then update its model after each new measurement. The model we discuss in §4.4 is lightweight and takes a few minutes to train on a single CPU thread from scratch. The benefits of this retraining is noticeable in a few instances, as we show in §5.

There is a balance between the bootstrap count b, and the guided exploration count $l-b$. If we set b too low, the prediction model is inaccurate, and if we set it too high, there is little opportunity for specializing the explored treatments. Through empirical evaluation we found $b=3$ to be sufficient for our prediction model to be reliable.

4.4 Prediction

We experimented with several designs for the prediction model, and converged on a lightweight architecture. There are nuances, however, in the representation and loss function that make significant improvements. We also discuss alternative models, some of which we'll evaluate in §5.

Treatment representation. We represent treatments by the hyperparameters that define the corresponding configuration and algorithms. We may not use a single type of algorithm (*e.g.,* ABR) across all treatments. In these cases, we include a one-hot encoding of algorithm choice (all hyperparameters from all algorithms).

In our experiments, we found that supplying these hyperparameters is immensely helpful to predictions. Naïvely representing treatments with a one-hot vector fared worse. This is not surprising; a model that observes these parameters directly can spot patterns between them and QoE that help predictions, *e.g.,* the pattern between stall rate and concurrent fragments seen in Table 1.

Cohort representation. We represent devices, network clusters and content types with one-hot encoded vectors. Beyond device family, other information about the device (*i.e.,* hardware/software specifications) is often not reliably known. Beyond the scope of this work, we hypothesize that a standardized test that profiles devices along important capabilities could help provide more information about devices.

4.4.1 Training Procedure

Score objective. We train a lightweight fully connected neural network to predict a score value $s_{i,j}$, given representations for cohort i and treatment j in input. We found the definition of this score function to be important in the prediction models accuracy; naïvely using the QoE directly $s_{i,j} := qoe_{i,j}$ does poorly as the naïve score function gives equal importance to predicting QoEs of good and poor treatments, however, accurate prediction of good treatments is much more valuable than the prediction accuracy for poor treatments.

Another challenge is the uneven scale of QoEs. Some cohorts have larger average QoEs compared to others, which skews the loss function towards them. Even among treatments of the same cohort, QoE can differ by 30 across treatments. Normalizing across cohorts or treatments does not resolve the issue, as we show in §5.

We can sidestep these challenges if score is set to a *softmax version of the QoE metric*,

$$s_{i,j} = \frac{m_{i,j} \times e^{qoe_{i,j}/\tau}}{\sum_{r=1}^{n} m_{i,r} \times e^{qoe_{i,r}/\tau}}, \quad (2)$$

where τ is a temperature value denoting the significance of QoE differences. For two treatments a and b,

$$\frac{s_{i,a}}{s_{i,b}} = e^{(qoe_{i,a}-qoe_{i,b})/\tau},$$

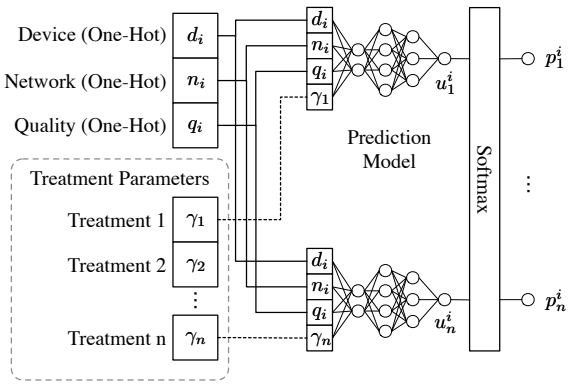


Figure 8: The Prediction Network Architecture which takes cohort identifiers and treatment parameters and outputs a logit. A softmax over logits produces treatment scores.

i.e., the relative score of two treatments depends on how far apart their QoEs are. We can change τ to fit our interpretations of QoE magnitudes. In our evaluation, we found $\tau = 0.1$ works well (tuned on a standard validation set), which is on same order as our QoE improvements.

This score function is nearly zero for poor treatments by design, circumventing the equal importance issue. It also has the same scale for all cohorts, since all these scores are positive and sum to 1 per cohort. For the new cohort where not all treatments have been explored, we used a masked softmax score instead.

4.4.2 Prediction Model

Since the target score sums to 1, we can treat prediction as a classification task and use a cross entropy loss to learn it. We train a fully connected network with parameterized by θ to predict logits $f(d_i,n_i,q_i,\gamma_j;\theta) = u_i^j$ and softmax outputs p_i^j, depicted in Figure 8. The loss function amounts to

$$\mathcal{L}_{CE} = \sum_{i=1}^{m}\sum_{j=1}^{n} -m_i^j \times s_i^j \times \log p_i^j \tag{3}$$

Given the low rank nature of the QoE matrix (§2.3), an alternate technique to predict performance of unobserved treatments is matrix completion [17]. While adequate for the task, matrix completion assumes that unobserved entries in the given matrix are missing at random [42, 48, 57] which is unnecessary in AZEEM's case. This is because AZEEM controls what treatments it wants to bootstrap and explore. As such, fully connected neural networks are much more effective, as we demonstrate later in §5.

4.5 Recommendation

When selecting k treatments for the final recommendation, AZEEM should be wary of aggressively over-tuning on the exploration period. While we cannot account for drifts that happen in the future, we can make more conservative predictions.

To regularize our recommendations against such drifts, we take $x > 0$ from the bootstrap treatments selected in §4.3, and choose the rest of the $k - x > 0$ treatments based on explored and predicted QoEs for the new device. The rationale behind this is that in case data drift causes the model recommendations to become stale, bootstrap treatments are a good next layer of defense. This is especially so, since as covered in §4.3, the choice of bootstrap treatments promoted diversity.

We empirically evaluate the effect of x on the final QoE outcomes in §5. Overall, recommendations policy where $0 < x < k$ are robust and well-performing, with minor gains from tuning x. By default, we suggest $x = 1$.

5 Evaluation

We evaluate AZEEM's ability to optimize large scale video playback with efficient exploration. We compare AZEEM against prior work [24, 34] while mimicking operational settings of a video provider. We study the exploration cost associated with various schemes as well as the QoE improvements from recommendations.

5.1 Setup

Data. We conduct an RCT with over 50 devices across 84 treatments on the large-scale video delivery service. Over a period of 5 months–December 2024 to April 2025–we collect data from over 750 million video-on-demand (VOD) streams across diverse network conditions and device families, including Smart TVs, PCs, set-top boxes, and more. These treatments encompass a range of parameters, including ABR algorithms (deployed in production), fragment downloader settings (*e.g.,* parallel download threads), bandwidth estimation, and custom bitrate ladders.

Metrics. QoE is an inherently subjective metric and despite several efforts, lacks widely accepted consensus [54]. Usually QoE is defined as a mix of stall rate, video quality, bitrate stability and startup delay, *etc*. where video quality is often measured with different metrics, *e.g.,* bitrate, SSIM, PSNR, VMAF [47, 69, 70]. While AZEEM is indifferent to the specifics of the QoE model, we briefly describe what this metric comprises of:

- **Stall Rate:** A small but non-negligible subset of sessions observe stalling. The amount of stalling per session is heavy-tailed and too high-variance to reliably measure. Instead, we use a simpler indicator, *PSB*: *what fraction of sessions experienced any stalling?*. This is a sensible metric as 1) it is low-variance and 2) advancements in network infrastructure have significantly reduced the intensity of stalls, making them a weaker signal to rely on [20].

- **Video Quality:** Per session, we report *quality degradation*, *i.e.,* the fraction of time a session did not stream at maximum available resolution (*e.g.,* SD, HD, UHD). Furthermore, to focus on tail performance, we use the average quality degradation at tails of users–80th to 99th percentile–denoted with *PQD*.
- **Quality Stability:** Beyond average quality, QoE scores [19, 47, 70] typically include quality stability. We measure the fraction of sessions which observed bitrate changes greater than a threshold, denoted with *PLS*.
- **Startup Delay:** A key QoE metric is the time before the video playback begins, also known as the startup delay. We measure the percentage of sessions that had startup delays longer than 10 seconds, denoted with *PSD*.

We define QoE as a linear mix of these metrics,

$$\text{QoE} = 100 - \frac{1}{1.53}\left[PSB + \frac{PLS}{3} + \frac{PSD}{10} + \frac{PQD}{10}\right], \quad (4)$$

where a value of 0 denotes the worst possible quality, and a value of 100 denotes the best. We assign the highest weight to stall rate. Because *PQD* is a tail average, it has a higher magnitude, which we compensate for with a lower weight.

Baselines. We compare AZEEM with a range of schemes, including prior work for optimizing video delivery [24, 34].

1. **BayesOpt** [24]: Bayesian optimization is the typical approach to black-box tuning and is leveraged by well known large scale video providers [24].
2. **Vizier** [29]: While Bayesian optimization can't naturally use QoE data from devices other than the new one, we compare against Vizier which regularizes posteriors for the new device using data from other devices.
3. **Pytheas** [34]: This scheme uses a multi-armed bandit agent to explore and find the best performing treatment. It uses Discounted Upper Confidence Bound (UCB) [28], a variant of the UCB algorithm [8], that accounts for drifts.
4. **AZEEM-Static:** We also evaluate how AZEEM performs if we do not use the prediction model for extra device-specific exploration.
5. **Default-Treatments:** A static selection of treatments will be recommended with no exploration, based on the offline collected dataset.

5.2 Results

To evaluate AZEEM, we emulate the process of onboarding a new device by using the RCT dataset and leave-one-out experimentation. This data contains over 50 devices and 300 cohorts (<device, network cluster, content type> tuples, see §4.1.1). For m iterations, we designate one device as the target device to be onboarded and use the remaining $(m-1)$ devices as the offline collected data. For a given exploration budget l, we explore bootstrap and cohort-guided treatments for the target device. Finally we recommend k treatments per each cohort of this device.

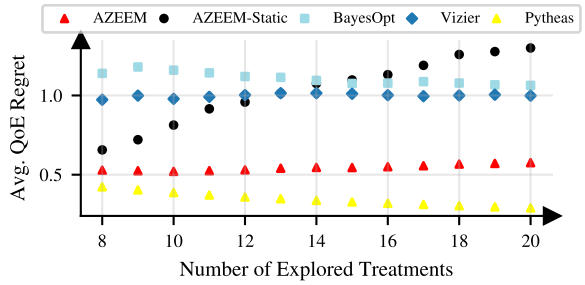
We are interested in 'QoE regret', *i.e.,* the QoE difference between a deployed treatment and the optimal one for that cohort. We look at two instances of QoE regret; **(1)** How much average QoE regret did each scheme cause due to exploration, and **(2)** What is the best QoE regret we observe in an evaluation period after exploration, among the k recommended treatments. The latter is an upper bound on how well a live adaptation system can track the best treatment in the suggested list.

Exploration period. Figure 9a depicts the average QoE regret due to exploration, when the exploration period is January 2025. Black-box exploration exploitation schemes–BayesOpt and Vizier–have no prior knowledge of which treatments are potentially good. These schemes will eventually identify better treatments, but only after exploring dozens of them first. Pytheas uses a multi-armed bandit strategy that makes decisions at a granularity of individual sessions. This allows us to abandon a treatment if its QoE does not look good, which reduces its average QoE regret. AZEEM-Static uses the offline collected data to make exploration recommendations that, initially, are better than black-box approaches; with small l, the bootstrap set is conservative, but as l grows, AZEEM-Static may pick treatments that are great on some devices but quite bad for others. Regardless of this budget, AZEEM outperforms all schemes in exploration QoE regret by $1.2-2.6\times$. AZEEM's prediction model allows it to make targeted exploration choices, rather than shots in the dark like most other schemes.

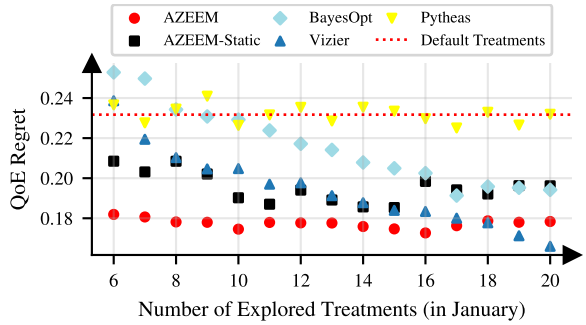
Exploitation period. From each scheme, we take $k = 5$ recommended treatments, and observe their QoE regrets. Figure 9b depicts the best QoE regret from this set. First, note that almost all schemes need exploration to lower this regret. AZEEM, however, has almost gained all information it can with $l = 6$. In fact, there is an instance of AZEEM with $l = 3$ that reaches a best regret QoE of 0.19. We purposefully limit the figures to $l > k$, since 1) at $l = 4, 5$, AZEEM will always make identical recommendations to $l = 3$, and 2) most baselines cannot make recommendations if $l \leq k$.

BayesOpt and Pytheas make worse recommendations with $l = 20$ than what AZEEM does with $l = 3$, *i.e.,* a 6-fold improvement in exploration time (and a $8.3-13.6\times$ reduction in total exploration QoE regret). AZEEM makes the best recommendations up to $l < 18$, after which Vizier takes a small lead. Note that $l = 20$ is 25% of the treatment space explored, and would take 1-2 months of exploration. AZEEM achieves the same level at $l = 6$, for a 3-fold reduction in exploration time (and $5.8\times$ reduction in total exploration QoE regret).

Choosing k and l. To understand the effect of k and l, we vary k, while setting $l = k+1$ (most schemes require $l > k$). We also include AZEEM at $l = 3$, denoted as AZEEM-NoGuided. The outcome is depicted in Figure 10. First, note that AZEEM with and without guided exploration get similar results



(a) Exploration Period (January 2025)



(b) Exploitation Period (February 2025), $k=5$

Figure 9: Exploration cost and exploitation performance of various schemes. **(a)** Average QoE regret during exploration. **(b)** Best QoE regret during the exploitation period of the $k=5$ recommended treatments. AZEEM recommends better treatments with fewer explored treatments.

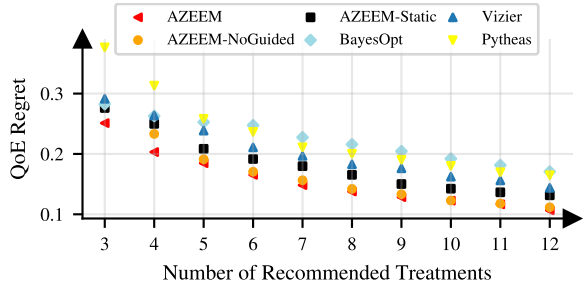


Figure 10: Best QoE regret for various schemes at $l=k+1$ and AZEEM-NoGuided at $l=3$, across k. AZEEM consistently outperforms baselines, improving as k increases. AZEEM-NoGuided closely follows, despite lower exploration.

beyond $k>4$. There is a consistent improvement observed with k, but this is an idealized view of what the live adaptation system will achieve with these k treatments. In practice, as k increases, traffic share is diluted more between treatments and it becomes harder for the live adaptation system to recognize the best treatment out of k. We leave a systematic design and evaluation of this system to future work.

Can $k=1$ work? We observe the QoE regret of the recommended treatment with $k=1$ depicted in Figure 11. The most striking pattern is that nearly all schemes make a

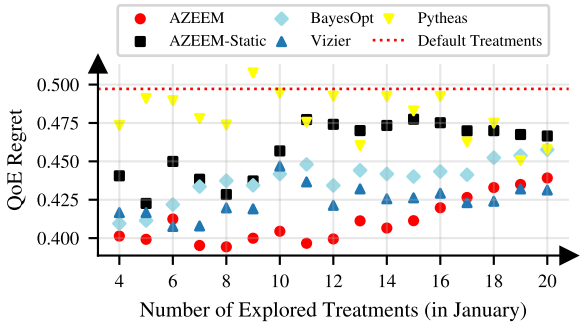


Figure 11: QoE regret during the exploitation period for the $k=1$ recommended treatment. Due to drift, no model makes better recommendations with more exploration.

worse recommendation as they explore more. This is the QoE drift in action; by aggressively exploring the treatment space, these schemes overfit to patterns that dissipate as we move to the exploitation period afterwards. Despite this, AZEEM still makes better recommendations overall.

Overall, AZEEM simultaneously makes better recommendations with a smaller exploration footprint. In the next subsection, we study AZEEM's behavior as we vary hyperparameters, dataset period, prediction model architecture, *etc*.

5.3 Deployment Case Study

To evaluate AZEEM's in-the-wild efficacy, we deployed it for a subset of 300 cohorts of Amazon Prime Video's production traffic. To ensure a fair comparison, we randomly allocated 10% of traffic of these cohorts to AZEEM and the remaining to an existing Production Treatment Tuning system (PTT). As discussed in §4.1, AZEEM was trained to explore treatments which differed in ABR algorithms (and their parameters), download parallelism and bitrate ladders.

Over a two week evaluation period with hundreds of millions of video streams, we measured the per-cohort QoE difference between AZEEM and PTT. Figure 12 shows the CDF of this QoE differences (higher is better). Overall, AZEEM improved QoE for more than 70% of cohorts, achieving an average QoE gain of 2.21% and > 5% for over 17% of cohorts. Figure 12 also shows improvement in individual metrics (§5.1). Most notably, AZEEM significantly reduces startup delay while avoiding major regressions on other metrics. AZEEM also lowers PSB (percentage of sessions with rebuffering) by up to 13% and improves PSB by at least 1% for more than 10% of the cohorts. Finally, we investigated cohorts where AZEEM was outperformed and found that the optimal treatments for these cohorts were not present in the 84 treatments AZEEM was trained for, which limited the extent of improvements it could achieve. Overall the deployment results validate AZEEM efficacy in quickly tuning performance across a large number of varied cohorts.

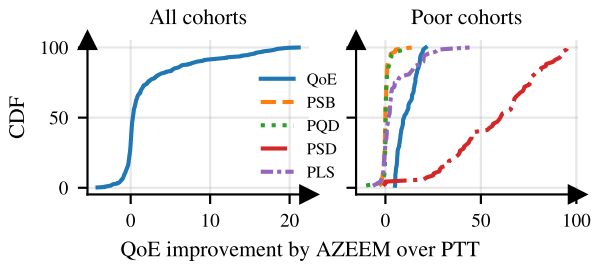


Figure 12: AZEEM's QoE improvements compared to PTT. (a) Across 300 cohorts, QoE improves by an average of 2.21% (221 basis points). (b) For about 17% of cohorts, AZEEM improves QoE by > 5% (500 basis points). Also shown are the improvements in individual QoE metrics for these cohorts.

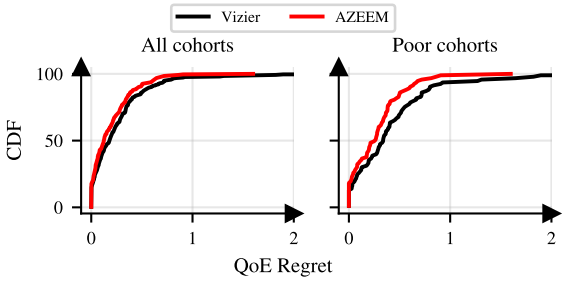


Figure 13: Distribution of best QoE regret across various devices. **(Left)** All clients. **(Right)** Clients with weak network setups. AZEEM brings most notable gains in weaker networks.

AZEEM Overhead. AZEEM is trained on 84×300 data samples (treatments × cohorts). Training finishes in a few minutes on AWS SageMaker ml.g5.48xlarge instance (192 vCPUs, 768 GiB RAM, 8×NVIDIA A10G GPUs, 24 GiB memory).

5.4 Ablation Study

QoE in poor cohorts. High quality video streaming is relatively easier for cohorts with better network conditions [69]. Figure 13 demonstrates the best QoE regret observed before, but focuses on cohorts with weaker network setups. Note the widened gap between AZEEM and baselines, compared to Figure 9b; this is because strong 'cohorts' dilute the QoE improvements made between these schemes. On average across such weak cohorts, AZEEM reduces the QoE regret by 1.7×.

Do these trends hold over time? The experiments in Figure 9b were over an exploration period with data from January 2025, and exploitation period in February 2025. In Figure 14 we can see the same experiment with a wider span for the exploration and evaluation periods. AZEEM maintains a similar QoE regret across time, while maintaining a lead over Vizier, the most competitive baseline in §5.2.

Changing the prediction model. We noted two alternate design choices in §4.4 that we ablate here. First, instead of

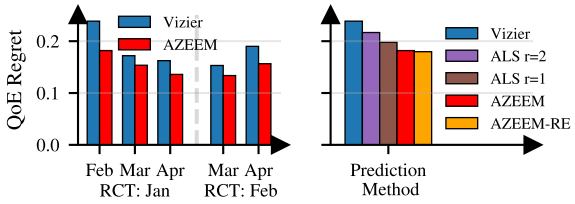


Figure 14: Ablations: (**Left**) Best QoE regret for $k=5$ across different exploration–evaluation periods, shows AZEEM's stability over time. (**Right**) Best QoE regret across prediction models: matrix completion outperform baselines but falls short of neural network model.

training a neural network as the prediction model, we can use matrix completion algorithms. We'll use the Alternating Least Squares (ALS) algorithm to learn the QoE matrix, with rank $r=1$ and $r=2$. Second, we can retrain the prediction model one treatment at a time after each exploration (dubbed AZEEM-RE). We compare these prediction model ablations in Figure 14 at $l=6$ and $l=20$ explored treatments. ALS does outperform Vizier, but does not outperform the neural network prediction model in AZEEM, no matter the exploration regime. Retraining the prediction model also shows little benefit here.

Effect of training loss. In §4.4, we discussed the implications of the loss function and prediction objective. Recall that AZEEM predicts a score value, derived with a softmax over QoE metrics. Figure 15 shows how alternate loss functions fare in our experiment. The "Direct Normalized" loss trains over a normalized version of the QoE, to reduce the effect of outliers in the matrix. As covered in §4.4, this doesn't resolve the issue where the loss function is still assigning equal importance to good and weak treatments, and does not perform as well as the softmax loss. Applying the softmax over a normalized QoE matrix should also not do better; recall that softmax over original QoEs assigns importance to treatments relative to their QoE difference. When we normalized the matrix, these difference can get exaggerated for some devices, fooling the loss function to learn a pattern that doesn't exist in the original QoE matrix.

The "Bootstrap Normalized" learns a direct mapping from QoEs of bootstrap treatments to QoE for a target treatment. In theory this should allow the matrix to learn the scale of the QoE for that particular device directly from its input. Unfortunately, this design does not allow the prediction model to benefit from guided exploration.

Does x matter? Recall that to make robust recommendations, we select x out of k recommended treatments from bootstrap treatments based on offline collected data. To evaluate this, we vary x across several exploration budgets, $l=6,9,12$, and observe QoE regret depicted in Figure 15. While no single value of x is optimal across all values of l, $x=0$ and $x=b$ ($b=3$ in this case) are usually the least performing candidates.

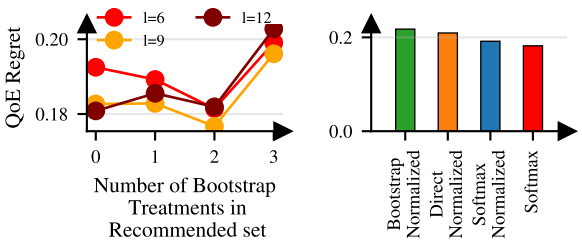


Figure 15: Ablations: (**Left**) Best QoE regret across $k = 5$ recommended treatments across different training loss and objectives, (**Right**) number of bootstrap treatments chosen at recommendation time across x. Across exploration budgets, a consistent trend is to avoid $x = 0$ and $x = b$.

6 Limitations

Long time horizon. Our evaluation is based on four months of data (January–April 2025). Within this period, we find that AZEEM outperforms other state-of-the-art methods such as Bayesian optimization, while also adapting to non-stationarity by recommending a treatment set that remains effective. A clear limitation, however, is that the entire treatment set may either become obsolete or get deprecated over longer time horizons, as underlying network conditions, device capabilities, and playback software features evolve. We therefore note the need for a continuous optimization system that can periodically update the recommended set over extended time frames and leave this for future work.

Dataset limitations. Our offline dataset consists of 84 treatments which were selected with the help of domain experts to maximize the chances of achieving optimal performance. As such, these treatments were already reasonably performant and limit the range of performance differences which would've been observed with more diverse treatments. This can be seen in §5.3, where the performance gains achieved by AZEEM against PTT are significantly higher than those obtained over our offline data. This is due to the greater performance variability between different treatments in production.

7 Related Work

Hyperparameter Tuning Techniques Hyperparameter tuning techniques form a well established line of work to intelligently manage the exploration-exploitation tradeoff. Techniques span a large surface area covering Multi-Armed and Contextual Bandits [9, 10, 40], Reinforcement Learning [66,68], Bayesian Optimization [14,59,60] and black-box hyper-parameter tuning [29] *etc*. In context of video streaming, past works have designed efficient exploration-exploitation techniques for bitrate and CDN selection [34], optimal ABR configurations for different network conditions [3] and bitrate ladder construction [63] *etc*. Our work takes inspiration from these to significantly reduce the exploration cost of finding the optimal settings for video streaming devices.

Optimizing Video QoE Techniques to improve video Quality of Experience are multifaceted with works focused on improving individual components of the video delivery pipeline including adaptive bitrate algorithms [3,5,19,31,47,61,69,70], fragment downloading strategies [30, 46], bitrate ladders [32,39,63,64] and CDN selection techniques [27,33,34] *etc*. Equally notable is line of work on quality of experience models which leverage data-driven techniques to understand how users respond to video stream impairments [12,13,23,36]. Our work complements these prior works by helping large scale video providers efficiently optimize experience at scale.

Low-rank structure Recall from §2.3 that the low-rank structure of the QoE matrix is one of the key insights that AZEEM exploits for video playback optimization. This structure is present in many real-world measurements [65] such as network data [15,37,38,56], and has proven useful for matrix completion [1,7,17,18] in cluster scheduling/manager [21, 22, 52], unbiased trace-driven simulation [5], efficient data analytics job placement [35], elastic training in heterogeneous GPU clusters [50], configuring new channels in cellular networks [45], and 5G device localization [41]. As such, AZEEM's underlying technique can generalize beyond video streaming.

8 Conclusion

Large scale video platforms optimize playback performance by tuning a range of different parameters. However, fragmentation in the device ecosystem makes it prohibitively expensive to rely on traditional tuning techniques such as manual A/B testing or Bayesian optimization to find device specialized settings. Our work introduces AZEEM, which leverages few-shot predictions to prune the search space of configurations. It then recommends a set of configurations (or treatments) which are robust to temporal distribution shifts. Evaluations conducted over real datasets from a large video provider show that AZEEM reduces the treatment exploration costs by $5.8 - 13.6\times$ over Bayesian optimization and multi-armed bandits thus enabling efficient device optimization at scale.

Acknowledgements

We express gratitude to our shepherd, Francis Y. Yan, and the anonymous reviewers for their invaluable feedback which greatly improved the paper. We extend our thanks to Sharath Dharmaji, Luis Naranjo and Nitin Singh for their help in deploying AZEEM in production. Finally, we are also grateful to Azeem Akhtar for inspiring the name of our system in this work. This research was supported in part by the MIT-Amazon Science Hub and by NSF award CNS-2504568.

References

[1] Anish Agarwal, Munther Dahleh, Devavrat Shah, and Dennis Shen. Causal matrix completion. In Gergely Neu and Lorenzo Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 3821–3826. PMLR, 12–15 Jul 2023.

[2] Zahaib Akhtar, Yun Seong Nam, Jessica Chen, Ramesh Govindan, Ethan Katz-Bassett, Sanjay Rao, Jibin Zhan, and Hui Zhang. Understanding video management planes. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18. Association for Computing Machinery, 2018.

[3] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: Auto-Tuning Video ABR Algorithms to Network Conditions. pages 44–58. ACM, 8 2018.

[4] Streaming Video Technology Alliance. The woes of device fragmentation and what to do about it. https://www.svta.org/webinar/the-woes-of-device-fragmentation-and-what-to-do-about-it/, 2021. Streaming Video Technology Alliance.

[5] Abdullah Alomar, Pouya Hamadanian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. CausalSim: A causal framework for unbiased Trace-Driven simulation. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1115–1147, Boston, MA, April 2023. USENIX Association.

[6] Apple. AVFoundation, 2025.

[7] Susan Athey, Mohsen Bayati, Nikolay Doudchenko, Guido Imbens, and Khashayar Khosravi. Matrix completion methods for causal panel data models. *Journal of the American Statistical Association*, 116(536):1716–1730, 2021.

[8] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[9] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[10] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 05 2002.

[11] AWS. AWS Device Farm, 2025.

[12] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. Developing a predictive model of quality of experience for internet video. *SIGCOMM Comput. Commun. Rev.*, 43(4):339–350, August 2013.

[13] Christos George Bampis, Zhi Li, Anush Krishna Moorthy, Ioannis Katsavounidis, Anne Aaron, and Alan Conrad Bovik. Study of temporal effects on subjective video quality of experience. *IEEE Transactions on Image Processing*, 26(11):5217–5231, 2017.

[14] James Bergstra, R. Bardenet, Balázs Kégl, and Y. Bengio. Algorithms for hyper-parameter optimization. 12 2011.

[15] Vineet Bharti, Pankaj Kankar, Lokesh Setia, Gonca Gürsun, Anukool Lakhina, and Mark Crovella. Inferring invisible traffic. In *Proceedings of the 6th International COnference*, Co-NEXT '10, New York, NY, USA, 2010. Association for Computing Machinery.

[16] Streaming Media Blog. The seriousness of device fragmentation and how it impacts streaming services. https://www.streamingmediablog.com/2022/07/device-fragmentation.html, 2022. Streaming Media Blog.

[17] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, June 2012.

[18] Emmanuel J. Candes and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[19] Tianyu Chen, Yiheng Lin, Nicolas Christianson, Zahaib Akhtar, Sharath Dharmaji, Mohammad Hajiesmaili, Adam Wierman, and Ramesh K. Sitaraman. Soda: An adaptive bitrate controller for consistent high-quality video streaming. In *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, page 613–644. Association for Computing Machinery, 2024.

[20] Conviva. The state of streaming in 2022. https://www.conviva.com/wp-content/uploads/2022/09/Q2-SoS.pdf, 2022. Conviva Industry Reports.

[21] Christina Delimitrou and Christos Kozyrakis. Paragon: Qos-aware scheduling for heterogeneous datacenters. *SIGPLAN Not.*, 48(4):77–88, March 2013.

[22] Christina Delimitrou and Christos Kozyrakis. Quasar: resource-efficient and qos-aware cluster management. *SIGPLAN Not.*, 49(4):127–144, February 2014.

[23] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, page 362–373, New York, NY, USA, 2011. Association for Computing Machinery.

[24] Guillaume du Pontavice. Improving streaming experience with bayesian optimization, from ab to az test! https://2021.demuxed.com, 2021. Demuxed 2021 Video Conference.

[25] Firebase. Firebase Test Lab, 2025.

[26] DASH Industry Forum. dash.js: A Reference Client Implementation for the Playback of MPEG DASH via JavaScript and Compliant Browsers, 2023.

[27] Aditya Ganjam, Junchen Jiang, Xi Liu, Vyas Sekar, Faisal Siddiqi, Ion Stoica, Jibin Zhan, and Hui Zhang. C3: internet-scale control plane for video quality optimization. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, page 131–144, USA, 2015. USENIX Association.

[28] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems, 2008.

[29] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1487–1495. ACM, 2017.

[30] Jiawei Huang, Qichen Su, Weihe Li, Zhuoran Liu, Tao Zhang, Sen Liu, Ping Zhong, Wanchun Jiang, and Jianxin Wang. Opportunistic transmission for video streaming over wild internet. *ACM Trans. Multimedia Comput. Commun. Appl.*, 18(3s), February 2023.

[31] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *ACM SIGCOMM Computer Communication Review*, 44:187–198, 2 2015.

[32] Tianchi Huang, Rui-Xiao Zhang, and Lifeng Sun. Deep reinforced bitrate ladders for adaptive video streaming. NOSSDAV '21, page 66–73, New York, NY, USA, 2021. Association for Computing Machinery.

[33] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. Cfa: a practical prediction system for video qoe optimization. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, page 137–150, USA, 2016. USENIX Association.

[34] Junchen Jiang, Shijie Sun, Vyas Sekar, and Hui Zhang. Pytheas: Enabling Data-Driven quality of experience optimization using Group-Based Exploration-Exploitation. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 393–406, Boston, MA, March 2017. USENIX Association.

[35] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. Selecta: Heterogeneous cloud storage configuration for data analytics. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 759–773, 2018.

[36] S. Shunmuga Krishnan and Ramesh K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, page 211–224, New York, NY, USA, 2012. Association for Computing Machinery.

[37] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing network-wide traffic anomalies. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, page 219–230, New York, NY, USA, 2004. Association for Computing Machinery.

[38] Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft. Structural analysis of network traffic flows. *SIGMETRICS Perform. Eval. Rev.*, 32(1):61–72, June 2004.

[39] Pierre Lebreton and Kazuhisa Yamagishi. Quitting ratio-based bitrate ladder selection mechanism for adaptive bitrate video streaming. *IEEE Transactions on Multimedia*, 25:8418–8431, 2023.

[40] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 661–670, New York, NY, USA, 2010. Association for Computing Machinery.

[41] Yu-Tai Lin, N Cameron Matson, and Karthikeyan Sundaresan. Bringing collaborative positioning to native 5g systems for enhanced 2d & 3d location services. *Proceedings of the ACM on Networking*, 2(CoNEXT4):1–21, 2024.

[42] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2019.

[43] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. *SIGCOMM Comput. Commun. Rev.*, 42(4):359–370, August 2012.

[44] Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, and Hui Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370, 2012.

[45] Ajay Mahimkar, Ashiwan Sivakumar, Zihui Ge, Shomik Pathak, and Karunasish Biswas. Auric: using data-driven recommendation to automatically generate cellular configuration. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 807–820, 2021.

[46] Tarun Mangla, Ellen Zegura, Mostafa Ammar, Emir Halepovic, Kyung-Wook Hwang, Rittwik Jana, and Marco Platania. Videonoc: assessing video qoe for network operators using passive measurements. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, page 101–112. Association for Computing Machinery, 2018.

[47] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural Adaptive Video Streaming with Pensieve. pages 197–210. ACM, 8 2017.

[48] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, UAI'07, page 267–275, Arlington, Virginia, USA, 2007. AUAI Press.

[49] Microsoft. App Center Test, 2025.

[50] Zizhao Mo, Huanle Xu, and Chengzhong Xu. Heet: Accelerating elastic training in heterogeneous deep learning clusters. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 499–513, 2024.

[51] Matthew K. Mukerjee, David Naylor, Junchen Jiang, Dongsu Han, Srinivasan Seshan, and Hui Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 311–324, New York, NY, USA, 2015. Association for Computing Machinery.

[52] Weiwu Pang, Sourav Panda, Jehangir Amjad, Christophe Diot, and Ramesh Govindan. {CloudCluster}: Unearthing the functional structure of a cloud service. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1213–1230, 2022.

[53] Sagar Patel, Junyang Zhang, Nina Narodystka, and Sangeetha Abdu Jyothi. Practically high performant neural adaptive video streaming. *Proc. ACM Netw.*, 2(CoNEXT4), November 2024.

[54] Leonardo Peroni and Sergey Gorinsky. Quality of experience in video streaming: Status quo, pitfalls, and guidelines. In *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 558–567, 2024.

[55] Norbert Potocki. Optimizing video playback performance. https://medium.com/pinterest-engineering/optimizing-video-playback-performance-caf55ce310d1, 2017. Pinterest Engineering Blog.

[56] Matthew Roughan, Yin Zhang, Walter Willinger, and Lili Qiu. Spatio-temporal compressive sensing and internet traffic matrices (extended version). *IEEE/ACM Transactions on Networking*, 20(3):662–676, 2012.

[57] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[58] Sandvine. Global internet phenomena report 2024. Technical report, Sandvine Incorporated, 2024. Accessed: 2025-04-22.

[59] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[60] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

[61] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, 8 2020.

[62] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 272–285, 2016.

[63] Farzad Tashtarian, Abdelhak Bentaleb, Hadi Amirpour, Sergey Gorinsky, Junchen Jiang, Hermann Hellwagner,

and Christian Timmerer. ARTEMIS: Adaptive bitrate ladder optimization for live video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 591–611, Santa Clara, CA, April 2024. USENIX Association.

[64] Laura Toni, Ramon Aparicio-Pardo, Karine Pires, Gwendal Simon, Alberto Blanc, and Pascal Frossard. Optimal selection of adaptive streaming representations. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(2s), February 2015.

[65] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[66] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2015.

[67] Talha Waheed, Ihsan Ayyub Qazi, Zahaib Akhtar, and Zafar Ayyub Qazi. Coal not diamonds: How memory pressure falters mobile video qoe. CoNEXT '22, page 307–320, 2022.

[68] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[69] F.Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. Learning in Situ: A Randomized Experiment in Video Streaming. In *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020*, pages 495–511, 2020.

[70] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, New York, NY, USA, 8 2015. ACM.

[71] Minlan Yu, Wenjie Jiang, Haoyuan Li, and Ion Stoica. Tradeoffs in cdn designs for throughput oriented traffic. CoNEXT '12, page 145–156, New York, NY, USA, 2012. Association for Computing Machinery.

Appendices

A Device-level Specialization

A.1 Why Are Devices Different?

In a production user base, typical of large scale providers, thousands of devices are served by the provider. To understand why these different behaviors arise, their root causes can be broadly attributed to the following.

Hardware Heterogeneity. Production devices differ drastically in their hardware capabilities. For instance, in the devices we worked with, we found that a subset of device models in the set-top box family from a single manufacturer suffered excessive frame drops[6] at bitrates above 5 Mb/s due to weak hardware resources[7] and thus needed custom bitrate ladders to prevent playback stalls and stutters despite excellent networking conditions.

Uneven API Support. Different devices exhibit uneven API support and lack feature parity which makes it challenging to take optimal settings from one device and readily deploy to another. For example, devices which run single threaded players (e.g. JavaScript Dash.js [26]) cannot easily benefit from techniques which use multi-thread parallelism (*e.g.,* concurrent fragment downloads). We show this change in stall rate for a large set of devices in Figure 16; for some devices, concurrent fragments are very helpful, while for some others they dramatically increase stall rate.

Limited Control of Software Stack. On most devices, critical components of the software stack–such as the Application SDK, Firmware and Operating System *etc*.–are either provided by the manufacturer or another third-party vendor. Thus video services have limited control in updating or making direct improvements in cases where performance deficiencies are localized to these components. Any changes require coordination with external vendors, leading to prolonged resolution times as fixes must progress through the vendors' release cycles. During this period, users continue to experience suboptimal performance while updates are developed and deployed.

A.2 Alternates to Data-Driven Specialization

Given that the need for cohort specializations arises due to factors originating from device dynamics, it is worth considering three alternates to data-driven specialization. First, one may think of designing components in the playback stack which are device-aware (*e.g.,* a device-aware ABR algorithm that jointly takes into account both network and device conditions). A second effort worthy of consideration is identification of root causes of issues which adversely impact device performance (*e.g.,* the aforementioned frame drops) and then fixing those issues. A third approach is to capture device-specific behaviors in simulations/emulations of such devices. While all are fair considerations, practical constraints render these approaches infeasible if not impossible.

Device Aware Playback Components As an example of a device-aware playback component, consider an ABR which makes bitrate decisions through joint optimization over real time network and device conditions. While *possible* it is not *feasible* to design this algorithm for at least two reasons. First, the full volume of metrics needed to cover the broad range of device behaviors makes designing this ABR algorithm complicated. Such metrics need to include available CPU cycles and architecture, hardware acceleration features, available memory, network processing rate, operating system, decoding hardware *etc*. Note that investigating two metrics—buffer occupancy and network bandwidth—has taken over a decade of research, and investigations have recently begun on a select few device metrics [67]. Second, for many of such metrics there are no standardized low-overhead mechanisms to frequently poll them from the device. Thus, it takes more effort to engineer this pipeline for the existing thousands of devices than to instance-optimize them in a black-box fashion.

Fixing Device Issues While determining root causes of issues which lead to low performing devices and fixing them is also appealing, often times specific behaviors are caused by components outside the control of a video service provider. For example, issues which arise due to deficiencies in the device firmware or runtime application framework *etc* can't be directly fixed by the video service provider. For instance, it is well known that to support Apple devices (iOS, MacOS, tvOS *etc*) providers must use the closed source media Software Development Kit (SDK) provided by Apple [6]. As such, any device characteristics determined by the performance of such third-party libraries must be taken into stride by the video service provider *as-is* and be handled in a black-box manner.

Device Simulation/Emulation Tools for offline exploration-exploitation or performance evaluation, such as simulators [3, 5] and large scale device farms [11, 25, 49] suffer from their own shortcomings. Aforementioned simulators do not consider device specific behaviors and are not easily extendable to cover the large search space of possible configurations (*i.e.,* CPU, memory, thread-parallelism *etc*). On the other hand large scale device farms are prohibitively expensive to build and operationalize. To make matters worse device capabilities and video streaming strategies continually evolve over time, thus require continued longitudinal investments to ensure adequate coverage of in-the-wild devices.

B Algorithm

Algorithm 1 is a pseudo-code for the overall onboarding procedure that AZEEM undergoes.

[6] Frame drop means the video frame was not decoded in time to be viewed on the display.

[7] All models were marked HD capable by the manufacturer.

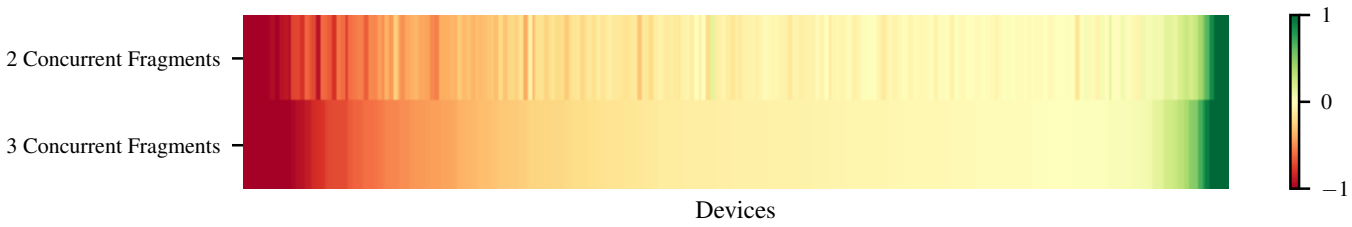


Figure 16: Stall rate improvement compared to one concurrent fragment, across various devices. We cap the colorbar scale at 1% degradation and 1% improvement to avoid outliers.

C Statistical Significance Tests

In Figure 9b, we show the QoE regret achieved by AZEEM and other methods under different exploration budgets. As described in §5, the QoE regret is computed across m iterations, where each device is used once as the test device and the remaining m-1 devices provide training data. While the main evaluation reports the average QoE regret across these m devices, we include the corresponding statistical tests here (for lack of space in the main paper). Specifically, we compare Bayesian Optimization and AZEEM at an exploration budget of six treatments and perform statistical tests to demonstrate that the improvements are statistically significant.

For each cohort, we compute confidence intervals for the QoE regret metrics. The QoE score, defined in §5.1, is modeled using Beta distributions for PSB, PSD, and PLS (percentage of sessions with playback degradation) and a Student's t-distribution for PQD (percentage of time a session did not stream at the maximum available resolution). To estimate the confidence interval of the average QoE score across m devices, we draw samples from these distributions, average them across devices, and obtain the empirical distribution of the mean QoE score for each method.

Comparing Bayesian Optimization and AZEEM at an exploration budget of six treatments, the empirical one-sided p-value for the difference in QoE scores is $p = 0.0033$, indicating that the improvements achieved by AZEEM in Figure 9b are statistically significant.

Algorithm 1 AZEEM: New Device Onboarding

Arguments

l (number of treatments to explore).
b (number of bootstrap treatments).
k (number of treatments to deploy).
x (number of deployed bootstrap treatments).
m cohorts ($m-1$ from offline RCT) and n treatments.
Training Parameters: E (iterations), η (learning rate).

Offline Data Collection

Explore all n treatments for $m-1$ RCT cohorts, and observe QoEs

Exploration (Bootstrap)

$BT \leftarrow []$
for b iterations **do**
    $j \leftarrow \text{argmax}_{j \in [1..n]-BT} \sum_{i=1}^{m-1} qoe_{i,j} - max_{j' \in BT} qoe_{i,j'}$
    $BT.add(j)$
Explore treatments in BT for cohort m, and observe QoEs

Prediction

Initialize neural network parameters θ
for E Iterations **do**
    for $\forall i \in [1..m], j \in [1..n]$ **do**
        $p_{i,j} \leftarrow u_{i,j}/\tau - \log \sum_{r=1}^{n} m_{i,r} \times e^{u_{i,r}/\tau}$
    $\mathcal{L}_{CE} \leftarrow \sum_{i=1}^{m} \sum_{j=1}^{n} -m_{i,j} \times s_{i,j} \times \log p_{i,j}$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{CE}$

Exploration (Guided)

$GT \leftarrow []$
for $l-b$ iterations **do**
    $j \leftarrow \text{argmax}_{j \in [1..n]-BT-GT} f(d_m, n_m, q_m, \gamma_j; \theta)$
    $GT.add(j)$
Explore treatments in GT for cohort m, and observe QoEs

Recommendation

$DT \leftarrow []$
for x iterations **do**
    $j \leftarrow \text{argmax}_{j \in BT-DT} qoe_{m,j}$
    $FT.add(j)$
for $k-x$ iterations **do**
    $j \leftarrow \text{argmax}_{j \in GT \cup BT-DT} qoe_{m,j}$
    $DT.add(j)$
Deploy treatments in FT for cohort m