Appeared in Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)

CFA: A Practical Prediction System for Video QoE Optimization

Junchen Jiang[†], *Vyas Sekar*[†], *Henry Milner*[⋆], *Davis Shepherd*[+], *Ion Stoica*[⋆+°], *Hui Zhang*[†+]
[†]*CMU,* [⋆]*UC Berkeley,* [+]*Conviva,* [°]*Databricks*

Abstract

Many prior efforts have suggested that Internet video Quality of Experience (QoE) could be dramatically improved by using data-driven prediction of video quality for different choices (e.g., CDN or bitrate) to make optimal decisions. However, building such a prediction system is challenging on two fronts. First, the relationships between video quality and observed session features can be quite complex. Second, video quality changes dynamically. Thus, we need a prediction model that is (a) expressive enough to capture these complex relationships and (b) capable of updating quality predictions in near real-time. Unfortunately, several seemingly natural solutions (e.g., simple machine learning approaches and simple network models) fail on one or more fronts. Thus, the potential benefits promised by these prior efforts remain unrealized. We address these challenges and present the design and implementation of Critical Feature Analytics (CFA). The design of CFA is driven by domain-specific insights that video quality is typically determined by a small subset of critical features whose criticality persists over several tens of minutes. This enables a scalable and accurate workflow where we automatically learn critical features for different sessions on coarse-grained timescales, while updating quality predictions in near real-time. Using a combination of a real-world pilot deployment and trace-driven analysis, we demonstrate that CFA leads to significant improvements in video quality; e.g., 32% less buffering time and 12% higher bitrate than a random decision maker.

1 Introduction

Delivering high quality of experience (QoE) is crucial to the success of today's subscription and advertisement-based business models for Internet video. As prior work (e.g., [33, 11]) has shown, achieving good QoE is challenging because of significant spatial and temporal variation in CDNs' performance, client-side network conditions, and user request patterns.

At the same time, these observations also suggest there is a substantial room for improving QoE by dynamically selecting the optimal CDN and bitrate based on a real-time global view of network conditions. Building on this insight, prior work makes the case for a quality optimization system (Figure 1) that uses a *prediction oracle* to suggest the best parameter settings (e.g., bitrate, CDN) to optimize quality (e.g., [33, 11, 35, 32, 20]). Seen in a broader context, this predictive approach can be applied beyond Internet video (e.g., [10, 40, 15, 16, 43]).

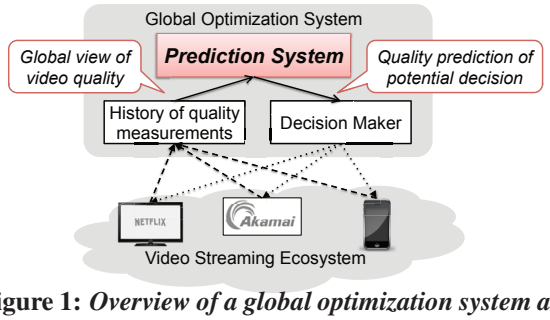


Figure 1: ***Overview of a global optimization system and the crucial role of a prediction system.***

However, these prior efforts fall short of providing a concrete instantiation of such a prediction system. Specifically, we observe that designing such a prediction system is challenging on two key fronts (§2):

- **Capturing complex factors that affect quality:** For instance, an outage may affect only clients of a specific ISP in a specific city when they use a specific CDN. To accurately predict the quality of their sessions, one must consider the combination of all three factors. In addition, the factors that affect video quality vary across different sessions; e.g., wireless hosts may be bottlenecked at the last connection, while other clients may experience loading failures due to unavailability of specific content on some CDNs.
- **Need for fresh updates:** Video quality changes rapidly, on a timescale of several minutes. Ideally, we must make predictions based on recent quality measurements. This is particularly challenging given the volume of measurements (e.g., YouTube had 231 million video sessions and up to 500 thousand concurrent viewers during the Olympics [7]), compounded with the need for expressive and potentially complex prediction models.

Unfortunately, many existing solutions fail on one or

both counts. For instance, solutions that use less complex models (e.g., linear regression, Naive Bayes, or simple models based on last-mile connection) are not expressive enough to capture high dimensional and diverse relationships between video quality and session features. More complex algorithms (e.g., SVM [42]) can take several hours to train a prediction model and will be inaccurate because predictions will rely on stale data.

In this work, we address these challenges and present the design and implementation of a quality prediction system called *Critical Feature Analytics* (CFA). CFA is built on three key domain-specific insights:

1. Video sessions with *same feature values* have *similar quality*. This naturally leads to an expressive model, wherein the video quality of a given session can be accurately predicted based on the quality of sessions that match values on all features (same ASN, CDN, player, geographical region, video content, etc). However, if applied naively, this model can suffer from the curse of dimensionality — as the number of combinations of feature values grows, it becomes hard to find enough matching sessions to make reliable predictions.
2. Each video session has a *subset* of *critical features* that ultimately determines its video quality. Given this insight, we can make more reliable predictions based on similar sessions that only need to match on critical features. For example, in a real event that we observed, congestion of a Level3 CDN led to relatively high loading failure rate for Comcast users in Baltimore. We can accurately predict the quality of the affected sessions using sessions associated with the specific CDN, region and ISP, ignoring other non-critical features (e.g., player, video content). Thus, this tackles the curse of dimensionality, while still retaining sufficient expressiveness for accurate prediction (§3).
3. Critical features tend to be *persistent*. Two remaining concerns are: (a) Can we identify critical features and (b) How expensive is it to do so? The insight on persistence implies that critical features are learnable from recent history and can be cached and reused for fast updates (§4). This insight is derived from recent measurement studies [25, 20] (e.g., the factors that lead to poor video quality persist for hours, and sometimes, even days).

Taken together, these insights enable us to engineer a scalable and accurate video quality prediction system. Specifically, on a coarse timescale of tens of minutes, CFA learns the critical features, and on a fine timescale of minutes, CFA updates quality prediction using recent quality measurements. CFA makes predictions and decisions as new clients arrive.

We implemented a prototype of CFA and integrated it in a video optimization platform that manages many premium video providers. We ran a pilot study on one content provider that has 150,000 sessions each day. Our real-world experiments show that the bitrates and CDNs selected by CFA lead to 32% less buffering time and 12% higher bitrate than a baseline random decision maker. Using real trace-driven evaluation, we also show that CFA outperforms many other simple ML prediction algorithms by up to 30% in prediction accuracy and 5-17% in various video quality metrics.

Contributions and Roadmap:

- Identifying key challenges in building an accurate prediction system for video quality (§2).
- Design and implementation of CFA, built on domain-specific insights to address the challenges (§3-5).
- Real-world and trace-driven evaluation that demonstrates substantial quality improvement by CFA (§6).
- Using critical features learned by CFA to make interesting observations about video quality (§7).

2 Background and Challenges

This section begins with some background on video quality prediction (§2.1). Then, we articulate two key challenges faced by any video quality prediction system: (1) The factors affecting video quality are *complex*, so we need expressive models (§2.2); (2) Quality changes *rapidly*, so models must be updated in near real-time by recent quality measurements (§2.3). We also argue why existing solutions do not address these challenges.

2.1 Background

Most video service providers today allow a video client (player) to switch CDN and bitrate among a set of available choices [33, 20, 32]. These switches have little overhead and can be performed at the beginning of and during a video playback [8]. Our goal then is to choose the best CDN and bitrate for a client by *accurately predicting the video quality of each hypothetical choice of CDN and bitrate*. In theory, if we can accurately predict the quality of each potential decision, then we can identify the optimal decision.

To this end, we envision a prediction system that uses a *global view* of quality measurements to make predictions for a specific video session. It learns a *prediction function* for each quality metric $Pred : 2^{\mathbb{S}} \times \mathbb{S} \mapsto \mathbb{R}$, which takes as input a given set of historical sessions $S \in 2^{\mathbb{S}}$ whose quality is already measured, and a new session $s \in \mathbb{S}$, and outputs a quality prediction $p \in \mathbb{R}$ for s.

Each quality measurement summarizes the quality of a video session for some duration of time (in our case, one minute). It is associated with values of four *quality metrics* [18] and a set of *features*[2] (summarized in Table 1).

[1] For one session, VSF is zero if it starts successfully, one otherwise.

[2] By feature, we refer to the type of attribute (e.g., *CDN*), rather than value of these attributes (e.g., *CDN* = *Akamai*)

Metrics	Description
BufRatio	Fraction of time a session spends in buffering (smooth playback is interrupted by buffering).
AvgBitrate	Time-weighted average of bitrates in a session.
JoinTime	Delay for the video to start playing from the time the user clicks "play".
Video start failure (VSF)	Fraction of sessions that fail to start playing (e.g., unavailable content or overloaded server)[1].
Features	**Description**
ASN	Autonomous System to which client IP belongs.
City	City where the client is located.
ConnectionType	Type of access network; e.g., mobile/fixed wireless, DSL, fiber-to-home [3].
Player	e.g., Flash, iOS, Silverlight, HTML5.
Site	Content provider of requested video contents.
LiveOrVoD	Binary indicator of live vs. VoD content.
ContentName	Name of the requested video object.
CDN	CDN a session started with.
Bitrate	Bitrate value the session started at.

Table 1: ***Quality metrics and session features associated with each session. CDN and Bitrate refer to initial CDN/bitrate values as we focus on initial selections.***

In general, the set of features depends on the degree of instrumentation and what information is visible to a specific provider. For instance, a CDN may know the location of servers, whereas a third-party optimizer [1] may only have information at the CDN granularity. Our focus is not to determine the best set of features that should be recorded for each session, but rather engineer a prediction system that can take an arbitrary set of features as inputs and extract the relationships between these features and video quality. In practice, the above set of features can already provide accurate predictions that help improve quality.

Our dataset consists of 6.6 million quality measurements collected from 2 million clients using 3 large public CDNs distributed across 168 countries and 152 ISPs.

2.2 Challenge 1: Expressive models

We show real examples of the complex factors that impact video quality, and the limitations in capturing these relationships.

High-dimensional relationship between video quality and session features. Video quality could be impacted by *combinations* of multiple components in the network. Such high-dimensional effects make it harder to learn the relationships between video quality and features, in contrast to simpler settings where features affect quality independently (e.g., assumed by Naive Bayes).

In a real-world incident, video sessions of Comcast users in Baltimore who watched videos from Level3 CDN experienced high failure rate (VSF) due to congested edge servers, shown by the blue line in Figure 2. The figure also shows the VSF of sessions sharing the same values on one or two features with the affected sessions; e.g., all Comcast sessions across different cities and CDNs. In the figure, the high VSF of the affected sessions cannot be clearly identified if we look at the sessions that match on only one or two features. Only when three features of *CDN* ("Level3"), *ASN* ("Comcast") and *City* ("Baltimore") are specified (i.e., blue line), can we detect the high VSF and predict the quality of affected sessions accurately.

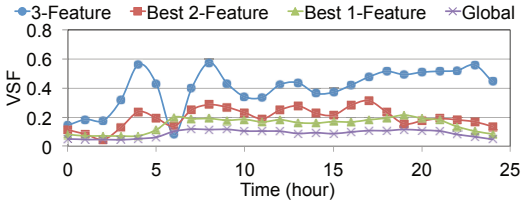


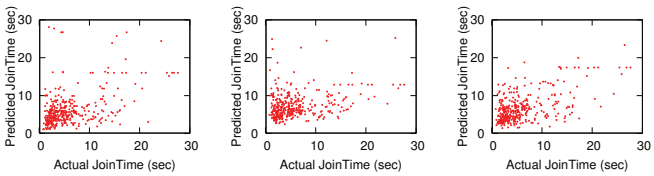
Figure 2: ***The high VSF is only evident when three factors (CDN, ISP and geo-location) are combined.***

In practice, we find that such high-dimensional effects are the common case, rather than an anomalous corner case. For instance, more than 65% of distinct CDN-ISP-City values have VSF that is at least 50% higher or lower than the VSF of sessions matching only one or two features (not shown). In other words, their quality is affected by a combined effect of at least three features.

Limitation of existing solutions: It might be tempting to develop simple predictors; e.g., based on the last-hop connection by using average quality of history sessions with the same *ConnectionType* value. However, they do not take into account the combined impact of features on video quality. Conventional machine learning techniques like Naive Bayes also suffer from the same limitation. In Figures 3(a) and 3(b), we plot the actual JoinTime and the prediction made by the last-hop predictor and Naive Bayes (from `Weka` [6]) for 300 randomly sampled sessions. The figures also show the mean relative error ($\frac{|predicted-actual|}{actual}$). For each session, the prediction algorithms train models using historical sessions within a 10-minute interval prior to the session under prediction. It shows that the prediction error of both solutions is significant and two-sided (i.e., not fixable by normalization).

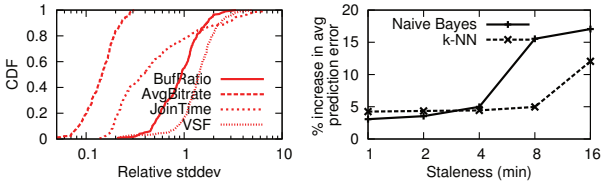
Highly diverse structures of factors. The factors that affect video quality vary across different sessions. This means the prediction algorithm should be expressive enough to predict quality for different sessions using different prediction models. For instance, the fact that many fiber-to-the-home (e.g., FiOS) users have high bitrates and people on cellular connections have lower bitrates is largely due to the speed of their last-mile connection. In contrast, some video clients may experience video loading failures due to unavailability of specific content on some CDNs. A recent measurement study [25] has shown that many heterogeneous factors are correlated with video quality issues. In §7, we show that 15% of video sessions are impacted by more than 30 different combinations of features and give real examples of different factors that affect quality.

Limitation of existing solutions: To see why existing solutions are not sufficient, let us consider the *k*-nearest



(a) Last hop (0.76) (b) Naive Bayes (0.61) (c) k-NN (0.63)

Figure 3: ***Prediction error of some existing solutions is substantial (mean of relative error in parentheses).***



(a) Temporal variability (b) Impact of staleness on accuracy

Figure 4: ***Due to significant temporal variability of video quality (left), prediction error increases dramatically with stale data (right).***

neighbor (k-NN) algorithm. It does not handle diverse relationships between quality and features, because the similarity between sessions is based on the same function of features independent of the specific session under prediction. In Figure 3(c), we plot the actual values of JoinTime and the prediction made by k-NN with the same setup as Figure 3(a)(b). Similar to Naive Bayes and the last-hop predictor, k-NN has substantial prediction error.

2.3 Challenge 2: Fresh updates

Video quality has significant temporal variability. In Figure 4(a), for each quality metric and combination of specific CDN, city and ASN, we compute the mean quality of sessions in each 10-minute interval, and then plot the CDF of the relative standard deviation ($\frac{stddev}{mean}$) of the quality across different intervals. In all four quality metrics of interest, we see significant temporal variability; e.g., for 60% of CDN-city-ASN combinations, the relative standard deviation of JoinTime across different 10-minute intervals is more than 30%. Such quality variability has also been confirmed in other studies (e.g., [33]).

The implication of such temporal variability is that the prediction system must update models in near real-time. In Figure 4(b), we use the same setup as Figure 3, except that the time window used to train prediction models is several minutes prior to the session under prediction. The figure shows the impact of such staleness on the prediction error for JoinTime. For both algorithms, prediction error increases dramatically if the staleness exceeds 10 minutes. As we will see later, this negative impact of staleness on accuracy is not specific to these prediction algorithms (§6.3).

Limitation of existing solutions: The requirement to use the most recent measurements makes it infeasible to use computationally expensive models. For instance, it takes at least one hour to train an SVM-based prediction model from 15K quality measurements in a 10-minute interval for one video site, so the quality predictions will be based on information from more than one hour ago.

3 Intuition behind CFA

This section presents the domain-specific insights we use to help address the expressiveness challenge (§2.2). The first insight is that sessions matching on all features have similar video quality. However, this approach suffers from the curse of dimensionality. Fortunately, we can leverage a second insight that each video session has a subset of *critical features* that ultimately determine its video quality. We conclude this section by highlighting two outstanding issues in translating these insights into a practical prediction system.

Input: Session under prediction s, Previous sessions S
Output: Predicted quality p

```
/* S': identical sessions matching on all
   features with s in recent history(Δ)   */
S' ← SimilarSessionSet(s, S, AllFeatures, Δ);
/* Summarize the quality (e.g., median) of
   the identical sessions in S'.          */
p ← Est(S');
return p;
```

Algorithm 1: ***Baseline prediction that finds sessions matching on all features and uses their observed quality as the basis for prediction.***

3.1 Baseline prediction algorithm

Our first insight is that sessions that have identical feature values will naturally have similar (if not identical) quality. For instance, we expect that all Verizon FiOS users viewing a specific HBO video using Level3 CDN in Pittsburgh at Fri 9 am should have similar quality (modulo very user-specific effects such as local Wi-Fi interference inside the home). We can summarize the intuition as follows:

Insight 1: ***At a given time, video sessions having same value on every feature have similar video quality.***

Inspired by Insight 1, we can consider a baseline algorithm (Algorithm 1). We predict a session's quality based on "identical sessions", i.e., those from recent history that match values on *all* features with the session under prediction. Ideally, given infinite data, this algorithm is accurate, because it can capture all possible combinations of factors affecting video quality.

However, this algorithm is unreliable as it suffers from the classical curse of dimensionality [39]. Specifically, given the number of combinations of feature values (ASN, device, content providers, CDN, just to name a few), it is hard to find enough identical sessions needed

to make a robust prediction. In our dataset, more than 78% of sessions have no identical session (i.e., matching on all features) within the last 5 minutes.

3.2 Critical features

In practice, we expect that some features are more likely to "explain" the observed quality of a specific video session than others. For instance, if a specific peering point between Comcast and Netflix in New York is congested, then we expect most of these users will suffer poor quality, regardless of the speed of their local connection.

Insight 2: ***Each video session has a subset of* critical features *that ultimately determines its video quality.***

We already saw some real examples in §2.2: in the example of high dimensionality, the critical features of the sessions affected by the congested Level3 edge servers are $\{ASN, CDN, City\}$; in the examples of diversity, the critical features are $\{ConnectionType\}$ and $\{CDN, ContentName\}$. Table 2 gives more real examples of critical features that we have observed in operational settings and confirmed with domain experts.

Quality issue	Set of critical features
Issue on one player of Vevo	$\{Player, Site\}$
ESPN flipping between CDNs	$\{CDN, Site, ContentName\}$
Bad Level3 servers for Comcast users in Maryland	$\{CDN, City, ASN\}$

Table 2: ***Real-world examples of critical features confirmed by analysts at a large video optimization vendor.***

A natural implication of this insight is that it can help us tackle the curse of dimensionality. Unlike Algorithm 1, which fails to find a sufficient number of sessions, we can estimate quality more reliably by aggregating observations across a larger amount of "similar sessions" that only need to match on these *critical features*. Thus, critical features can provide expressiveness while avoiding curse of dimensionality.

Algorithm 2 presents a logical view of this idea:

1. **Critical feature learning (line 1):** First, find the critical features of each session s, denoted as $CriticalFeatures(s)$.
2. **Quality estimation (line 2, 3):** Then, find similar sessions that match values with s on critical features $CriticalFeatures(s)$ within a recent history of length Δ (by default, 5 minutes). Finally, return some suitable estimate of the quality of these similar sessions; e.g., the median[3] (for BufRatio, AvgBitrate, JoinTime) or the mean (for VSF).

A practical benefit of Algorithm 2 is that it is interpretable [52], unlike some machine learning algorithms

[3]We use median because it is more robust to outliers.

```
Input: Session under prediction s, Previous sessions S
Output: Predicted quality p
/* CF_s: Set of critical features of s        */
1 CF_s ← CriticalFeatures(s);
/* S': Similar sessions matching values on
   critical features CF_s with s.             */
2 S' ← SimilarSessionSet(s, S, CF_s, Δ);
/* Summarize the quality of the similar
   sessions in S'.                            */
3 p ← Est(S');
4 return p;
```

Algorithm 2: ***CFA prediction algorithm, where prediction is based on similar sessions matching on critical features.***

(e.g., PCA or SVM). This allows domain experts to combine their knowledge with CFA and diagnose prediction errors or resolve incidents, as we explore in §7.2.

At this time, it is useful to clarify what critical features are and what they are not. In essence, critical features provide the explanatory power of how a prediction is made. However, critical features are not a minimal set of factors that determine the quality (i.e., root cause). That is, they can include both features that reflect the root cause as well as additional features. For example, if all HBO sessions use Level3, their critical features may include both *CDN* and *Site*, even if *CDN* is redundant, since including it does not alter predictions. The primary objective of CFA is accurate prediction; root cause diagnosis may be an added benefit.

3.3 Practical challenges

There are two issues in using Algorithm 2.

Can we learn critical features? A key missing piece is how we get the critical features of each session (line 1). This is challenging because critical features vary both across sessions and over time [33, 25], and it is infeasible to manually configure critical features.

How to reduce update delay? Recall from §2.3 that the prediction system should use the most recent quality measurements. This requires a scalable implementation of Algorithm 2, where critical features and quality estimates are updated in a timely manner. However, naively running Algorithm 2 for millions of sessions under prediction is too expensive (§6.3). With a cluster of 32 cores, it takes 30 minutes to learn critical features for 15K sessions within a 10-minutes interval. This means the prediction will be based on stale information from tens of minutes ago.

4 CFA Detailed Design

In this section, we present the detailed design of CFA and discuss how we address the two practical challenges mentioned in the previous section: learning critical features and reducing update delay.

Notations	Domains	Definition
$s, S, \mathbb{S}$		A session, a set of sessions, set of all sessions
$q(s)$	$\mathbb{S} \mapsto \mathbb{R}$	Quality of s
$QualityDist(S)$	$2^{\mathbb{S}} \mapsto 2^{\mathbb{R}}$	$\{q(s) \vert s \in S\}$
$f, F, \mathbb{F}$		A feature, a set of features, set of all features
$CriticalFeatures(s)$	$\mathbb{S} \mapsto 2^{\mathbb{F}}$	Critical features of s
$\mathbb{V}$		Set of all feature values
$FV(f,s)$	$\mathbb{F} \times \mathbb{S} \mapsto \mathbb{V}$	Value on feature f of s
$FSV(F,s)$	$2^{\mathbb{F}} \times \mathbb{S} \mapsto 2^{\mathbb{V}}$	Set of values on features in F of s
$SimilarSessionSet(s,S,F,\Delta)$	$\mathbb{F} \times 2^{\mathbb{F}} \times \mathbb{S} \times \mathbb{R}^+ \mapsto 2^{\mathbb{F}}$	$\{s' \vert s' \in S, t(s) - \Delta < t(s') < t(s), FSV(F,s') = FSV(F,s)\}$

Table 3: ***Notations used in learning of critical features.***

The key to addressing these challenges is our third and final domain-specific insight:

Insight 3: ***Critical features tend to*** **persist** ***on long timescales of tens of minutes.***

This insight is derived from prior measurement studies [25, 20]. For instance, our previous study on shedding light on video quality issues in the wild showed that the factors that lead to poor video quality persist for hours, and sometimes even days [25]. Another recent study from the C3 system suggests that the best CDN tends to be relatively stable on the timescales of few tens of minutes [20]. We independently confirm this observation in §6.3 that using slightly stale critical features (e.g., 30-60 minutes ago) achieves similar prediction accuracy as using the most up-to-date critical features. Though this insight holds for most cases, it is still possible (e.g., on mobile devices) that critical features persist on a relatively shorter timescale (e.g., due to the nature of mobility).

Note that the persistence of critical features does not mean that quality values are equally persistent. In fact, persistence of critical features is on a timescale an order of magnitude longer than the persistence of quality. That is, even if quality fluctuates rapidly, the critical features that determine the quality do not change as often.

As we will see below, this persistence enables (a) automatic learning of critical features from history, and (b) a scalable workflow that provides up-to-date estimates.

4.1 Learning critical features

Recall that the first challenge is obtaining the critical features for each session. The persistence of critical features has a natural corollary that we can use to automatically learn them:

Corollary 3.1: ***Persistence implies that critical features of a session are learnable from history.***

Specifically, we can simply look back over the history and identify the subset of features F such that the quality distribution of sessions matching on F is most similar to that of sessions matching on *all* features. For instance, suppose we have three features $\langle ContentName, ASN, CDN \rangle$ and it turns out that sessions with $ASN = Comcast, CDN = Level3$ consistently have high buffering over the last few hours due to some internal congestion at the corresponding exchange point. Then, if we look back over the last few hours, the data from history will naturally reveal that the distribution of the quality of sessions with the feature values $\langle ContentName = Foo, ASN = Comcast, CDN = Level3 \rangle$ will be similar to $\langle ContentName = *, ASN = Comcast, CDN = Level3 \rangle$, but very different from, say, the quality of sessions in $\langle ContentName = *, ASN = *, CDN = Level3 \rangle$, or $\langle ContentName = *, ASN = Comcast, CDN = * \rangle$. Thus, we can use a data-driven approach to learn that ASN, CDN are the critical features for sessions matching $\langle ContentName = Foo, ASN = Comcast, CDN = Level3 \rangle$.

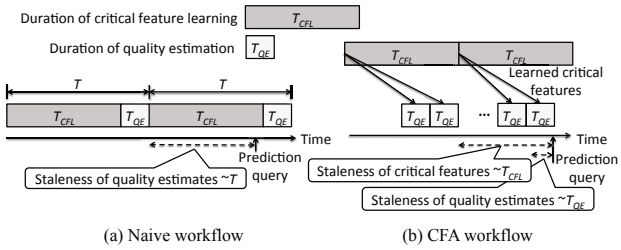
Input: Session under prediction s, Previous sessions S
Output: Critical features for s

```
/* Initialization                                          */
1  MaxSimilarity ← −∞, CriticalFeatures ← NULL;
   /* D_finest : Quality distribution of
      sessions matching on 𝔽 in Δ_learn.                   */
2  D_finest ← QualityDist(SimilarSessionSet(s, S, 𝔽, Δ_learn));
3  for F ⊆ 2^𝔽 do
       /* Exclude F without enough similar
          sessions for prediction.                         */
4      if |SimilarSessionSet(s, S, F, Δ)| < n then
5          continue;
       /* D_F : Quality distribution of
          sessions matching on F in Δ_learn.
          */
6      D_F ← QualityDist(SimilarSessionSet(s, S, F, Δ_learn));
       /* Get similarity of D_finest & D_F.                */
7      Similarity ← Similarity(D_F, D_finest);
8      if Similarity>MaxSimilarity then
9          MaxSimilarity ← Similarity;
10         CriticalFeatures ← F;
11 return CriticalFeature;
```

Algorithm 3: ***Learning of critical features.***

Algorithm 3 formalizes this intuition for learning critical features. Table 3 summarizes the notation used in Algorithm 3. For each subset of features F (line 3), we compute the similarity between the quality distribution (D_F) of sessions matching on F and the quality distribution (D_{finest}) of sessions matching on all features (line 7). Then, we find the F that yields the maximum similarity (line 8-10), under one additional constraint that $SimilarSessionSet(s, S, F, \Delta)$ should include enough (by default, at least 10) sessions to get reliable quality estimation (line 4-5). This check ensures that the algorithm will not simply return the set of all features.

As an approximation of the duration in which criti-



(a) Naive workflow (b) CFA workflow

Figure 5: ***To reduce update delay, we run critical feature learning and quality estimation at different timescales by leveraging persistence of critical features.***

cal features persist, we use $\Delta_{learn} = 60min$. Note that Δ_{learn} is an order of magnitude larger than the time window Δ used in quality estimation, because critical features persist on a much longer timescale than quality values. We use (the negative of) Jensen-Shannon divergence between D_1 and D_2 to quantify their similarity $Similarity(D_1, D_2)$.

Although Algorithm 3 can handle most cases, there are corner cases where $SimilarSessionSet(s, S, \mathbb{F}, \Delta_{learn})$ does not have enough sessions (i.e., more than n) to compute $Similarity(D_F, D_{finest})$ reliably. In these cases, we replace D_{finest} by the set of n sessions that share most features with s over the time window of Δ_{learn}. Formally, we use $\{s' | s'$ matches k_s features with $s\}$, where $k_s = \text{argmin}_k(|\{s' | s' \text{ matches } k \text{ features with } s| \geq n\}|)$.

4.2 Using fresh updates

Next, we focus on reducing the update delay between when a quality measurement is received and used for prediction.

Naively running critical feature learning and quality estimation of Algorithm 2 can be time-consuming, causing the predictions to rely on stale data. In Figure 5(a), T_{CFL} and T_{QE} are the duration of critical feature learning and the duration of quality estimation, respectively. The staleness of quality estimation (depicted in Figure 5) to respond to a prediction query can be as large as the total time of two steps (i.e., $T_{CFL} + T_{QE}$), which typically is tens of minutes (§6.3). Also, simply using more parallel resources is not sufficient. The time to learn critical features using Algorithm 3 grows linearly with the number of sessions under prediction, the number of history sessions, and the number of possible feature combinations. Thus, the complexity of learning critical features T_{CFL} is exponential in the number of features. Given the current set of features, T_{CFL} is on the scale of tens of minutes.

To reduce update delay, we again leverage the persistence of critical features:

Corollary 3.2: ***Persistence implies that critical features can be cached and reused over tens of minutes.***

Building on Corollary 3.2, we decouple the critical feature learning and quality estimation steps, and run them at separate timescales. On the timescale of tens of minutes, we update the results of critical feature learning. Then, on a faster timescale of tens of seconds, we update quality estimation using fresh data and the most recently learned critical features.

This decoupling minimizes the impact of staleness on prediction accuracy. Learning critical features on the timescale of tens of minutes is sufficiently fast as they persist on the same timescale. Meanwhile, quality estimation can be updated every tens of seconds and makes predictions based on quality updates with sufficiently low staleness. Thus, the staleness of quality estimation T_{QE} of the decoupled workflow (Figure 5(b)) is a magnitude lower than $T_{QE} + T_{CFL}$ of the naive workflow (Figure 5(a)). In §6.3, we show that this workflow can retain the freshness of critical features and quality estimates.

In addition, CFA has a natural property that two sessions sharing all feature values and occurring close in time will map to the same critical features. Thus, instead of running the steps per-session, we can reduce the computation to the granularity of *finest partitions*, i.e., distinct values of all features.

4.3 Putting it together

Building on these insights, we create the following practical *three-stage* workflow of CFA.

- **Stage I: Critical feature learning** (line 1 of Algorithm 2) runs offline, say, every tens of minutes to an hour. The output of this stage is a key-value table called *critical feature function* that maps all observed finest partitions to their critical features.
- **Stage II: Quality estimation** (line 2,3 of Algorithm 2) runs every tens of seconds for all observed finest partitions based on the most recent critical features learned in the first stage. This outputs another key-value table called *quality function* that maps a finest partition to the quality estimation, by aggregating the most recent sessions with the corresponding critical features.
- **Stage III: Real-time query/response.** Finally, we provide real-time query/response on the arrival of each client, operating at the millisecond timescale, by simply looking up the most recent precomputed value function from the previous stage. These operations are simple and can be done very fast.

Finally, instead of forcing all finest partition-level computations to run in every batch, we can do triggered recomputations of critical feature learning only when the observed prediction errors are high.

5 Implementation and Deployment

This section presents our implementation of CFA and highlights engineering solutions to address practical

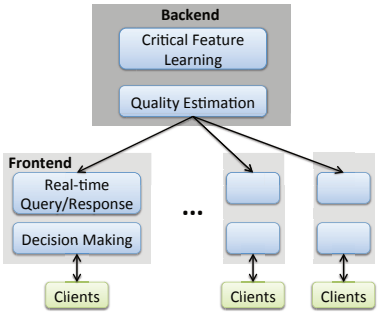


Figure 6: ***Implementation overview of CFA. The three stages of CFA workflow are implemented in a backend cluster and distribute frontend clusters.***

challenges in operational settings (e.g., avoiding bulk data loading and speeding up development iterations).

5.1 Implementation of CFA workflow

CFA's three stages are implemented in two different locations: a centralized backend cluster and geographically distributed frontend clusters as depicted in Figure 6.

Centralized backend: The critical feature learning and quality estimation stages are implemented in a backend cluster as periodic jobs. By default, critical feature learning runs every 30 minutes, and quality estimation runs every minute. A centralized backend is a natural choice because we need a global view of all quality measurements. The quality function, once updated by the estimation step, is disseminated to distributed frontend clusters using Kafka [27].

Note that we can further reduce learning time using simple parallelization strategies. Specifically, the critical features of different finest partitions can be learned independently. Similarly in Algorithm 3, the similarity of quality distributions can be computed in parallel. To exploit this data-level parallelism, we implement them as Spark jobs [4].

Distributed frontend: Real-time query/response and decision makers of CDN/bitrate are co-located in distributed frontend clusters that are closer to clients than the backend. Each frontend cluster receives the quality function from the backend and caches it locally for fast prediction. This reduces the latency of making decisions for clients.

5.2 Challenges in an operational setting

Mitigating impact of bulk data loading: The backend cluster is shared and runs other delay-sensitive jobs; e.g., analytics queries from production teams. Since the critical feature learning runs periodically and loads a large amount of data (≈30 GB), it creates spikes in the delays of other jobs (Figure 7). To address this concern, we engineered a simple heuristic to evenly spread the data retrieval where we load a small piece of data every few minutes. As Figure 7 shows, this reduces the spikes

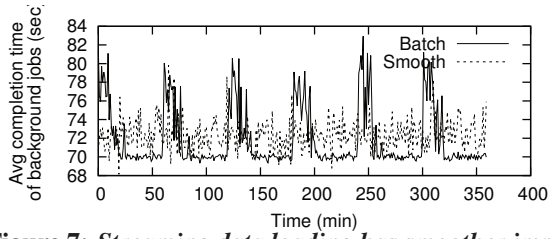


Figure 7: ***Streaming data loading has smoother impact on completion delay than batch data loading.***

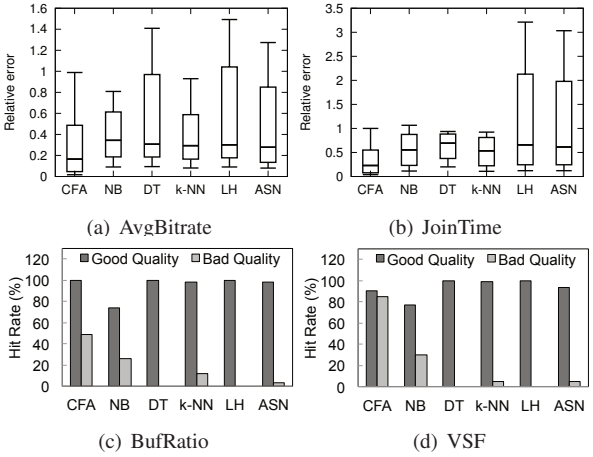


Figure 8: ***Distributions of relative prediction error ($\{5,10,50,90,95\}$%iles) on AvgBitrate and JoinTime and hit rates on BufRatio and VSF. They show that CFA outperforms other algorithms.***

caused by bulk data loading in batch mode. Note that this does not affect critical feature learning.

Iterative algorithm refinement: Some parameters (e.g., learning window size Δ_{learn}) of CFA require iterative tuning in a production environment. However, one practical challenge is that the frontend-facing part of the backend can only be updated once every couple of weeks due to code release cycles. Thus, rolling out new prediction algorithms may take several days and is a practical concern. Fortunately, the decoupling between critical feature learning and quality estimation (§4.2) means that changes to critical feature learning are confined to the backend cluster. This enables us to rapidly refine and customize the CFA algorithm.

6 Evaluation

In this section, we show that:

- CFA predicts video quality with 30% less error than competing machine learning algorithms (§6.1).
- Using CFA-based prediction, we can improve video quality significantly; e.g., 32% less BufRatio, 12% higher AvgBitrate in a pilot deployment (§6.2).
- CFA is responsive to client queries and makes predictions based on the most recent critical features and quality measurements (§6.3).

6.1 Prediction accuracy

We compare CFA with five alternative algorithms: three simple ML algorithms, Naive Bayes (NB), Decision Tree (DT), k-Nearest Neighbor (k-NN)[4], and two heuristics which predict a session's quality by the average quality of other sessions from the same ASN (ASN) or matching the last-mile connection type (LH). All algorithms use the same set of features listed in Table 1.

Ideally, we want to evaluate how accurately an algorithm can predict the quality of a given client on every choice of CDN and bitrate. However, this is infeasible since each video client is assigned to only one CDN and bitrate at any time. Thus, we can only evaluate the prediction accuracy over the observed CDN-bitrate choices, and we use the quality measured on these choices as the ground truth. That said, this approach is still useful for doing a relative comparison across different algorithms.

For AvgBitrate and JoinTime, we report *relative error*: $\frac{|p-q|}{q}$, where the q is the ground truth and p is the prediction. For BufRatio and JoinTime, which have more "step function" like effects [18], we report a slightly different measure called *hit rate*: how likely a session with good quality (i.e., BufRatio $< 5\%$, VSF=0) or bad quality is correctly identified. Figure 8 shows that for AvgBitrate and JoinTime, CFA has the lowest $\{5, 10, 50, 90\}$%th percentiles of prediction error and lower 95%th percentiles than most algorithms. In particular, median error of CFA is about 30% lower than the best competing algorithm. In terms of BufRatio and VSF, CFA significantly outperforms other algorithms in the hit rate of bad quality sessions. The reason for hit rate of bad quality to be lower than that of good quality is that bad quality sessions are almost always less than good quality, which makes them hard to predict. Note that accurately identifying sessions that have bad quality is crucial as they have the most room for improvement.

6.2 Quality improvement

Pilot deployment: As a pilot deployment, we integrated CFA in a production system that provides a global video optimization service [20]. We deployed CFA on one major content provider and used it to optimize 150,000 sessions each day. We ran an A/B test (where each algorithm was used on a random subset of clients) to evaluate the improvement of CFA over a baseline random decision maker, which many video optimization services use by default (modulo business arrangement like price) [9].

Table 4 compares CFA with the baseline random decision maker in terms of the mean BufRatio, AvgBitrate and a simple QoE model ($QoE = -370 * BufRatio + AvgBitrate/20$), which was suggested by [33, 18]. Over all sessions in the A/B testing, CFA shows an improvement in both BufRatio (32% reduction) and AvgBitrate (12.3% increase) compared to the baseline. This shows that CFA is able to simultaneously optimize multiple (possibly conflicting) metrics. To put these numbers in context, our conversation with domain experts confirmed that these improvements are significant for content providers and can potentially translate into substantial benefits in engagement and revenues [2]. CFA's superior performance and that CFA is more automated than the custom algorithm indicate that domain experts were willing to invest time running longer pilot. Figure 9 provides more comparison and shows that CFA consistently outperforms the baseline over time and across different major cities in the US, connection types and CDNs.

	CFA	Baseline	Improvement
QoE	155.43	138.27	12.4%
BufRatio	0.0123	0.0182	32%
AvgBitrate	3200	2849	12.31%

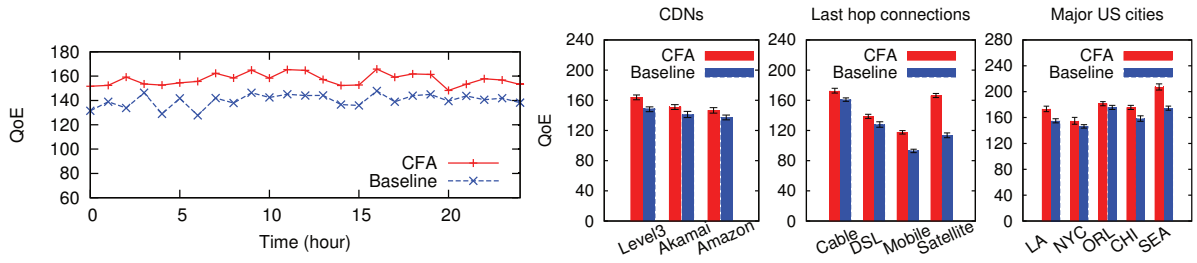
Table 4: ***Random A/B testing results of CFA vs. baseline in real-world deployment.***

Trace-driven simulation: We complement this real-world deployment with a trace-driven simulation to simultaneously compare more algorithms over more quality metrics. However, one key challenge is that it is hard to estimate the quality of a decision that was not used by a specific client in the trace.

To address this problem, we use the *counterfactual methodology* from prior work in online recommendation systems [30, 31]. Suppose we have quality measurements from a set of clients, where client c is assigned to a decision $d_{rand}(c)$ of CDN and bitrate at random. Now, we have a new hypothetical algorithm that maps client c to $d_{alg}(c)$. Then, we can evaluate the average quality of clients assigned to each decision d, $\{c|d_{alg}(c) = d\}$, by the average quality of $\{c|d_{alg}(c) = d, d_{rand}(c) = d\}$. Finally, the average quality of the new algorithm is the weighted sum of average quality of all decisions, where the weight of each decision is the fraction of sessions assigned to it. This can be proved to be an unbiased (offline) estimate of d_{alg}'s (online) performance [5].[5] For instance, if out of 1000 clients assigned to use Akamai and 500Kbps, 200 clients are assigned to this decision in the random assignment, then we can use the average quality of these 200 sessions as an unbiased estimate of the average quality of these 1000 sessions. Fortunately, our dataset includes a (randomly chosen) portion of clients with randomized decision assignments (i.e.,

[4] NB, DT, and k-NN are mplemented using a popular ML library `weka`[6].

[5] One known limitation of this analysis is that it assumes the new assignments do not affect each decision's overall performance. For instance, if we assign all sessions to one CDN, they may overload the CDN and so this CDN's quality in the random assignments is no longer useful. Since this work only focuses on controlling traffic at a small scale relative to the total load on the CDN (and our experiments are in fact performed at such a scale), this methodology is still unbiased.



(a) CFA vs. baseline by time

(b) CFA vs. baseline by spatial partitions

Figure 9: ***Results of real-world deployment. CFA outperforms the baseline random decision maker (over time and across different large cities, connection types and CDNs).***

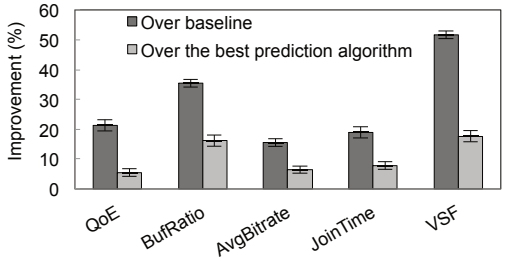
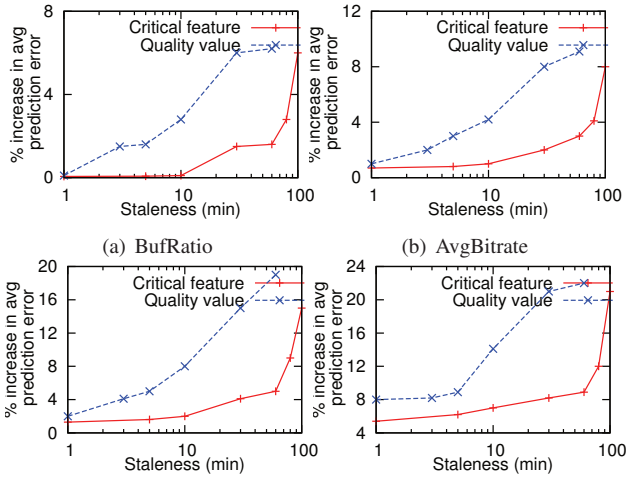


Figure 10: ***Comparison of quality improvement between CFA and strawmen.***

Stage	Run time (mean / median)	Required freshness
Critical feature learning	30.1/29.5 min	30-60 min
Quality estimation	30.7/28.5 sec	1-5 min
Query response	0.66/0.62 ms	1 ms

Table 5: ***Each stage of CFA is refreshed to meet the required freshness of its results.***



(a) BufRatio

(b) AvgBitrate

(c) JoinTime

(d) VSF

Figure 11: ***Latency of critical features and quality values (x-axis) on increase in accuracy (y-axis).***

CDN and bitrate). Thus, we only report improvements for these clients.

Figure 10 uses this counterfactual methodology and compares CFA with the best alternative from §6.1 for each quality metric and the baseline random decision maker (e.g., the best alternative of AvgBitrate is k-NN). For each quality metric and prediction algorithm, the decision maker selects the CDN and bitrate that has the best predicted quality for each client. For instance, the improvement of CFA over the baseline on VSF is 52% – this means the number of sessions with start failures is 52% less than when the baseline algorithm is used. The figures show that CFA outperforms the baseline algorithm by 15%-52%. They also show that CFA outperforms the best prediction algorithms by 5%-17%.

6.3 Timeliness of prediction

Our implementation of CFA should (1) retain freshness to minimize the impact of staleness on prediction accuracy, and (2) be responsive to each prediction query.

We begin by showing how fast each stage described in §4.2 needs to be refreshed. Figure 11 shows the impact of staleness of critical features and quality values on the prediction accuracy of CFA. First, critical features learned 30-60 minutes before prediction can still achieve similar accuracy as those learned 1 minute before prediction. In contrast, quality estimation cannot be more than 10 minutes prior to when prediction is made (which corroborates the results of Figure 4(b)). Thus, critical feature learning needs to be refreshed every 30-60 minutes and quality estimation should be refreshed at least every several minutes. Finally, prediction queries need to be responded to within several milliseconds [20] (ignoring network delay between clients and servers).

Next, we benchmark the time to run each logical stage described in §4.2. Real-time query/response runs in 4 geographically distributed data centers. Critical feature learning and quality estimation run on two clusters of 32 cores. Table 5 shows the time for running each stage and the timescale required to ensure freshness. It confirms that the implementation of CFA is sufficient to ensure the freshness of results in each stage.

7 Insights from Critical Features

In addition to the predictive power, CFA also offers insights into the "structure" of video quality in the wild. In this section, we focus on two questions: (1) What types

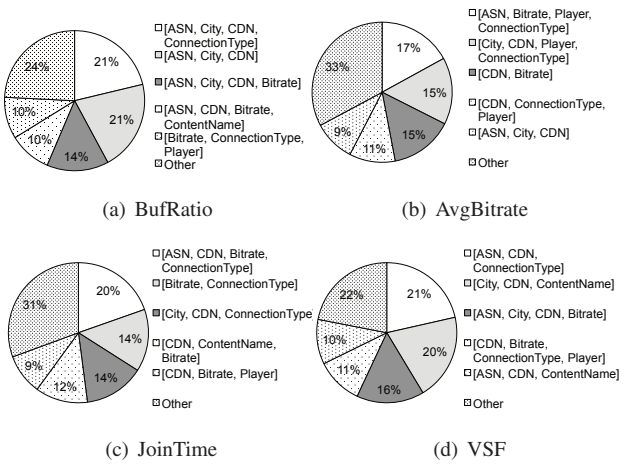


Figure 12: ***Analyzing the types of critical features: This shows a breakdown of the total number of sessions assigned to a specific type of critical features.***

of critical features are most common? (2) What factors have significant impact on video quality?

7.1 Types of critical features

Popular types of critical features: Figure 12 shows a breakdown of the fraction of sessions that are assigned to a specific type of critical feature set. We show this for different quality metrics. (Since we focus on a specific VoD provider, we do not consider the *Site* or *LiveOrVoD* for this analysis.) Across all quality metrics, the most popular critical features are *CDN*, *ASN* and *ConnectionType*, which means video quality is greatly impacted by network conditions at the server (*CDN*), transit network (*ASN*), and last-mile connection (*ConnectionType*).

We also see interesting patterns unique to individual metrics. *City* is among the top critical features of BufRatio. This is perhaps because network congestion usually depends on the volume of concurrent viewers in a specific region. *Bitrate* (initial bitrate) has a larger impact on AvgBitrate than on other metrics, since the videos in the dataset are mostly short content (2-5 minutes) and AvgBitrate is correlated with initial bitrate. Finally, *ContentName* has a relatively large impact on failures (VSF) but not other metrics, because VSF is sometimes due to the requested content not being ready.

Distribution of types of critical features: While the quality of about 50% of sessions is impacted by 3-4 popular types of critical features, 15% of sessions are impacted by a diverse set of more than 30 types of critical feature (not shown). This corroborates the need for expressive prediction models that handle the diverse factors affecting quality (§2.2).

7.2 Values of critical features

Next, we focus on the most prevalent *feature values* (e.g., a specific ASN or player). To this end, we define *prevalence* of a feature value by the fraction of video sessions matching this feature value that have this feature as one of their critical features; e.g., the fraction of video sessions from Boston that have *City* as one of their critical features. If a feature value has a large prevalence, then the quality of many sessions that have this feature value can be explained by this feature.

	City	*ASN*	*Player*	*ConnectionType*
BufRatio	Some major east-coast cities			Satellite, Mobile, Cable
AvgBitrate		Cellular carriers	Players with different encodings	
JoinTime		Cellular carrier		Satellite, DSL
VSF		Small ISPs		Satellite, Mobile

Table 6: ***Analysis of the most prevalent values of critical features. A empty cell implies that we found no interesting values in this combination.***

We present the values of critical features with a prevalence higher than 50% for each quality metric and only consider a subset of the features (*ASN*, *City*, *ContentName*, *ConnectionType*) that appear prominently in Figure 12. We present this analysis with two caveats. First, due to proprietary concerns, we do not present the names of the entities, but focus on their characteristics. Second, we cannot confirm some of our hypothesis as it involves other providers; as such, we intend this result to be illustrative rather than conclusive.

Table 6 presents some anecdotal examples we observed. In terms of BufRatio, we see some of the major east coast cities (e.g., Boston, Baltimore) are more likely to be critical feature values than other smaller cities. We also see both poor (Satellite, Mobile) and broadband (Cable) connection types have high prevalence on BufRatio and JoinTime. This is because poor quality sessions are bottlenecked by poor connections, while some good quality sessions are explained by their broadband connections. "Player" has a relatively large prevalence on AvgBitrate, because the content provider uses different bitrate levels for different players (Flash or iOS). Finally, in terms of VSF, some small ISPs have large prevalence. We speculate that this is because their peering relationships with major CDNs are not provisioned, so their video sessions have relatively high failure rates.

8 Related Work

Internet video optimization: There is a large literature on measuring video quality in the wild (e.g., content popularity [38, 55], quality issues [25] and server selection [51, 47]) and techniques to improve user experience (e.g., bitrate adaptation algorithms [56, 26, 23], CDN optimization and federation [32, 37, 11, 35] and cross-provider cooperation [57, 19, 24]). Our work builds on

insight from this prior work. While a case for a similar vision is made in [33], our work gives a systematic and practical prediction system.

Global coordination platform: Decision making based on a global view is similar to other logically centralized control systems (e.g., [14, 33, 20, 48]). They examined the architectural issues of decoupling the control plane from the data plane, including scalability (e.g., [50, 17]), fault tolerance (e.g., [36, 54]), and use of big data systems (e.g., [20, 4]). In contrast, our work offers concrete algorithmic techniques over such a control platform [20] for video quality optimization.

Large-scale data analytics in system design: Many studies have applied data-driven techniques for performance diagnosis (e.g., [46, 15, 40]), revenue debugging (e.g., [13]), TCP throughput prediction (e.g., [22, 34]), and tuning TCP parameters (e.g., [43, 41]). Recent studies try to operate these techniques at scale [16]. While CFA shares this data-driven approach, we exploit video-specific insights to achieve scalable and accurate prediction based on a global view of quality measurements.

QoE models: Prior work has shown correlations between various video quality metrics and user engagement (e.g., users are sensitive to BufRatio [18]), and built various QoE models (e.g., [28, 45, 12, 10]. Our work focuses on improving QoE by predicting individual quality metrics, and can be combined with these QoE models.

9 Discussion

Relationship to existing ML techniques: CFA is a domain-specific prediction system that outperforms some canonical ML algorithms (§6.1). We put CFA in the context of three types of ML algorithms.

- *Multi-armed bandit* algorithms [53] find the decision with the highest reward (i.e., best CDN and bitrate) from multiple choices. They assume each decision has a fixed distribution of rewards, but the video quality of a CDN also depends on client-side features. In contrast, contextual multi-armed bandit algorithms [44] assume the best decision depends on contextual information, but they require appropriate modeling between the context and decision space, to which critical features provide one viable approach.
- *The feature selection* problem [21] seems similar to critical feature learning, but with a key difference: critical features vary across video sessions. Thus, techniques looking for features that are most important for all sessions are not directly applicable.
- *Advanced ML* algorithms today can handle highly complex models [29, 42] efficiently, so in theory the critical features could be automatically identified, albeit in an implicit manner. CFA uses existing ML models (specifically, the "variable kernel conditional density estimation" method [49]) and may be less accurate than advanced ML techniques, but CFA can predict with more recent data since it tolerates stale update on the critical features. Furthermore, CFA is less opaque since it is based on domain-specific insights about critical features (§7).

Finer grain information and selection: Currently, CFA makes predictions based on client side information only. While clients provide accurate information regarding QoE, prediction can be much more accurate if CFA were to leverage finer-grained information from other entities in the ecosystem, including servers, caches and network paths. Furthermore, CFA currently selects resources at the CDN granularity. This means CFA cannot do much if the CDN redirects the client based on its location and the servers the CDN redirects the client to are congested. However, if the client were able to specify the server to stream from, we could avoid the overloaded servers and improve quality.

10 Conclusions

Many prior research efforts posited that quality prediction could lead to improved QoE (e.g., [33, 11, 35, 10]). However, these efforts failed to provide a prescriptive solution that (a) is expressive enough to tackle the complex feature-quality relationships observed in the wild and (b) can provide near real-time quality estimates. To this end, we developed CFA, a solution based on domain-specific insights that video quality is typically determined by a subset of critical features which tend to be persistent. CFA leverages these insights to engineer an accurate algorithm that outperforms off-the-shelf machine learning approaches and lends itself to a scalable implementation that retains model freshness. Using real deployments and trace-driven analyses, we showed that CFA achieves up to 30% improvement in prediction accuracy and 12-32% improvement in QoE over alternative approaches.

Acknowledgments

This paper would not be possible without the contribution of Conviva Stuff, especially Jibin Zhan, Faisal Zakaria Siddiqi and Rui Zhang. The authors thank Siddhartha Sen for shepherding the paper and the NSDI reviewers for their feedback. This research is supported in part by NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple Inc., Blue Goji, Bosch, Cisco, Cray, Cloudera, Ericsson, Facebook, Fujitsu, Guavus, HP, Huawei, Intel, Microsoft, Pivotal, Samsung, Schlumberger, Splunk, State Farm, Virdata and VMware. Junchen Jiang was supported in part by NSF award CNS-1345305 and Juniper Networks Fellowship.

References

[1] Conviva inc. http://www.conviva.com/.

[2] Personal communication with aditya ganjam from conviva, who is an expert on video qoe.

[3] Quova. http://developer.quova.com/.

[4] Spark. http://spark.incubator.apache.org/.

[5] Technical note on counterfactual evaluation. https://www.cs.cmu.edu/cfe_technote.pdf.

[6] The weka manual 3.6.10. http://goo.gl/ISSY3c.

[7] Youtube and the olympics. http://goo.gl/4hgL4q.

[8] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia*, 2011.

[9] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, , and Z.-L. Zhang. A Tale of Three CDNs: An Active Measurement Study of Hulu and Its CDNs. In *Proc. IEEE Global Internet Symposium*, 2012.

[10] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan. Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, 2014.

[11] A. Balachandran, V. Sekar, A. Akella, and S. Seshan. Analyzing the potential benefits of cdn augmentation strategies for internet video workloads. In *ACM IMC*, pages 43–56, 2013.

[12] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a predictive model of quality of experience for internet video. In *ACM SIGCOMM '13*.

[13] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah. Adtributor: revenue debugging in advertising systems. In *USENIX NSDI*, 2014.

[14] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *USENIX NSDI 2005*.

[15] D. R. Choffnes, F. E. Bustamante, and Z. Ge. Crowdsourcing service-level network event monitoring. In *ACM SIGCOMM CCR*, volume 40, pages 387–398. ACM, 2010.

[16] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In *Conference on Innovative Data Systems Research (CIDR)*, 2015.

[17] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella. Towards an elastic distributed sdn controller. In *ACM HotSDN*, 2013.

[18] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. A. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. In *Proc. SIGCOMM*, 2011.

[19] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber. Pushing cdn-isp collaboration to the limit. *ACM SIGCOMM CCR*, 43(3), 2013.

[20] A. Ganjam, F. Siddiqi, J. Zhan, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-scale control plane for video quality optimization. In *NSDI*. USENIX, 2015.

[21] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[22] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer tcp throughput. *ACM SIGCOMM CCR*, 35(4):145–156, 2005.

[23] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: evidence from a large video streaming service. In *ACM SIGCOMM 2014*.

[24] J. Jiang, X. Liu, V. Sekar, I. Stoica, and H. Zhang. Eona: Experience-oriented network architecture. In *ACM HotNets*, 2014.

[25] J. Jiang, V. Sekar, I. Stoica, and H. Zhang. Shedding light on the structure of internet video quality problems in the wild. In *CoNEXT*. ACM, 2013.

[26] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Streaming with Festive . In *ACM CoNEXT 2012*.

[27] J. Kreps, N. Narkhede, and J. Rao. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, 2011.

[28] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. In *IMC*, 2012.

[29] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[30] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[31] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011.

[32] H. Liu, Y. Wang, Y. R. Yang, A. Tian, and H. Wang. Optimizing Cost and Performance for Content Multihoming. In *Proc. SIGCOMM*, 2012.

[33] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A case for a coordinated internet video control plane. In *ACM SIGCOMM*, pages 359–370. ACM, 2012.

[34] M. Mirza, J. Sommers, P. Barford, and X. Zhu. A machine learning approach to tcp throughput prediction. In *ACM SIGMETRICS Performance Evaluation Review*, volume 35, pages 97–108. ACM, 2007.

[35] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. Practical, real-time centralized control for cdn-based live video delivery. In *ACM SIGCOMM*, pages 311–324. ACM, 2015.

[36] A. Panda, C. Scott, A. Ghodsi, T. Koponen, and S. Shenker. Cap for networks. In *ACM HotSDN*, 2013.

[37] L. Peterson and B. Davie. Framework for cdn interconnection. 2013.

[38] L. Plissonneau and E. Biersack. A longitudinal view of http video streaming performance. In *Proc. MMSys*, 2012.

[39] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[40] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger. Diagnosing performance changes by comparing request flows. In *USENIX NSDI*, 2011.

[41] S. Savage, N. Cardwell, and T. Anderson. The case for informed transport protocols. In *HotOS*, pages 58–63. IEEE, 1999.

[42] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[43] S. Seshan, M. Stemm, and R. H. Katz. Spand: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, pages 1–13, 1997.

[44] A. Slivkins. Contextual bandits with similarity information. *The Journal of Machine Learning Research*, 15(1):2533–2568, 2014.

[45] H. H. Song, Z. Ge, A. Mahimkar, J. Wang, J. Yates, Y. Zhang, A. Basso, and M. Chen. Q-score: Proactive Service Quality Assessment in a Large IPTV System. In *Proc. IMC*, 2011.

[46] M. Stemm, R. Katz, and S. Seshan. A network measurement architecture for adaptive applications. In *INFOCOM*, volume 1, pages 285–294. IEEE, 2000.

[47] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting behind akamai (travelocity-based detouring). *ACM SIGCOMM CCR*, 2006.

[48] L. Suresh, M. Canini, S. Schmid, and A. Feldmann. C3: Cutting tail latency in cloud data stores via adaptive replica selection. In *USENIX NSDI*, 2015.

[49] G. R. Terrell and D. W. Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[50] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On controller performance in software-defined networks. In *USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, 2012.

[51] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafo, and S. Rao. Dissecting Video Server Selection Strategies in the YouTube CDN. In *ICDCS*, 2011.

[52] A. Vellido, J. Martin-Guerroro, and P. Lisboa. Making machine learning models interpretable. In *Proceedings of the 20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN). Bruges, Belgium*, pages 163–172, 2012.

[53] R. Weber et al. On the gittins index for multiarmed bandits. *The Annals of Applied Probability*, 2(4):1024–1033, 1992.

[54] H. Yan, D. A. Maltz, T. E. Ng, H. Gogineni, H. Zhang, and Z. Cai. Tesseract: A 4d network control plane. In *NSDI*, volume 7, pages 27–27, 2007.

[55] H. Yin et al. Inside the Bird's Nest: Measurements of Large-Scale Live VoD from the 2008 Olympics. In *Proc. IMC*, 2009.

[56] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *SIGCOMM*, pages 325–338. ACM, 2015.

[57] M. Yu, W. Jiang, H. Li, and I. Stoica. Tradeoffs in cdn designs for throughput oriented traffic. In *CoNEXT*, pages 145–156. ACM, 2012.