CFA: A Practical Prediction System for Video QoE Optimization

Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, Hui Zhang

What is QoE ?

Quality of Experience (**QoE**) is a measure of the delight or annoyance of a customer's experiences with a service (e.g., web browsing, phone call, TV broadcast).

QoE factors

Human Influence Factors

Context Influence Factors

System Influence Factors

- Content-related
- Media-related (encoding, resolution, sample rate, …)
- Network-related (bandwidth, delay, jitter, …)
- Device-related (screen resolution, display size, …)

Why bother?

Delivering high quality of experience (QoE) is crucial to the success of today’s subscription and advertisement-based business models for Internet video.

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 realtime global view of network conditions.

Why bother?

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.

Issues:

-First, the relationships between video quality and observed session features can be quite complex

-Second, video quality changes dynamically.

Why bother?

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

Challenges

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.)

Three 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).

Challenges

Need for fresh updates:

Video quality changes rapidly, on a timescale of several minutes. Ideally, we must make predictions based on recent quality measurements. Volume of measurements is particularly challenging - e.g. YouTube had 231 million video sessions and up to 500 thousand concurrent viewers during the Olympics.

Challenges

Unfortunately, many existing solutions fail on one or one both counts.

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) can take several hours to train a prediction model and will be inaccurate because predictions will rely on stale data.

Critical Feature Analytics (CFA)

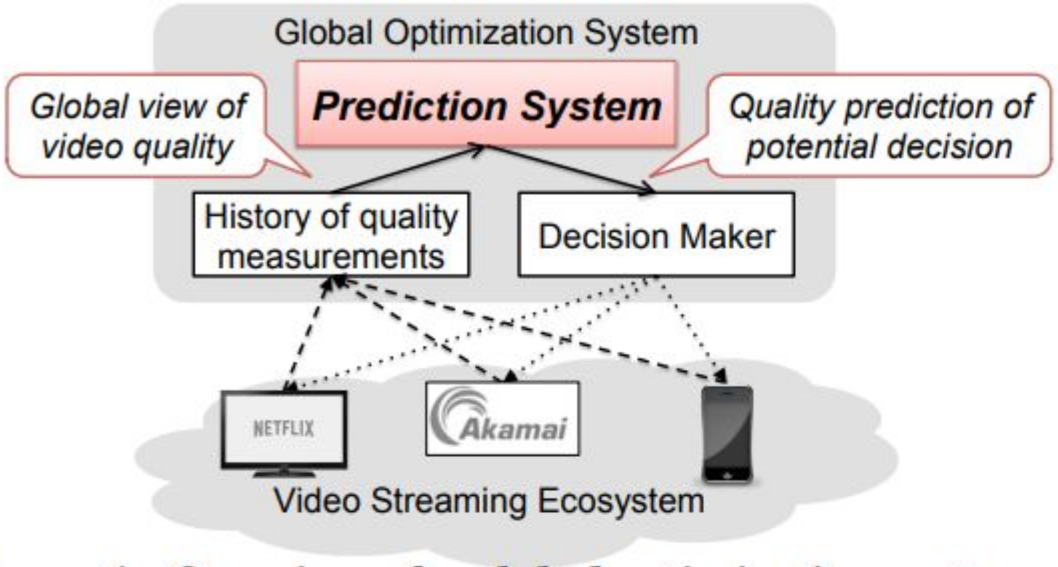

Global Optimization System
Prediction System
Global view of video quality
Quality prediction of potential decision
History of quality measurements
Decision Maker
NETFLIX
Akamai
Video Streaming Ecosystem

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

Critical Feature Analytics (CFA)

CFA is built on three key domain-specific insights:

> 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.

Critical Feature Analytics (CFA)

CFA is built on three key domain-specific insights:

> 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).

Critical Feature Analytics (CFA)

CFA is built on three key domain-specific insights:

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

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.***

Real life example

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.

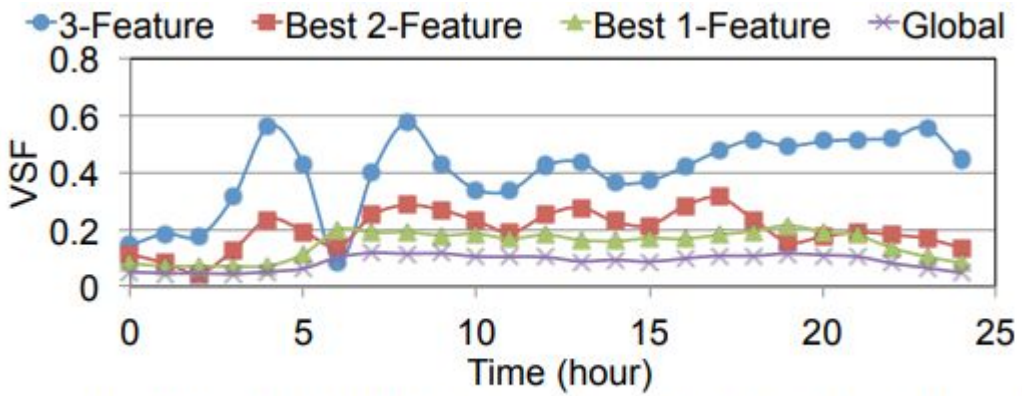


Figure 2: ***The high VSF is only evident when three factors (CDN, ISP and geo-location) are combined.***

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.

In practice, we find that such high-dimensional effects are the common case, rather than an anomalous corner case.

Challenge 1: Expressive models

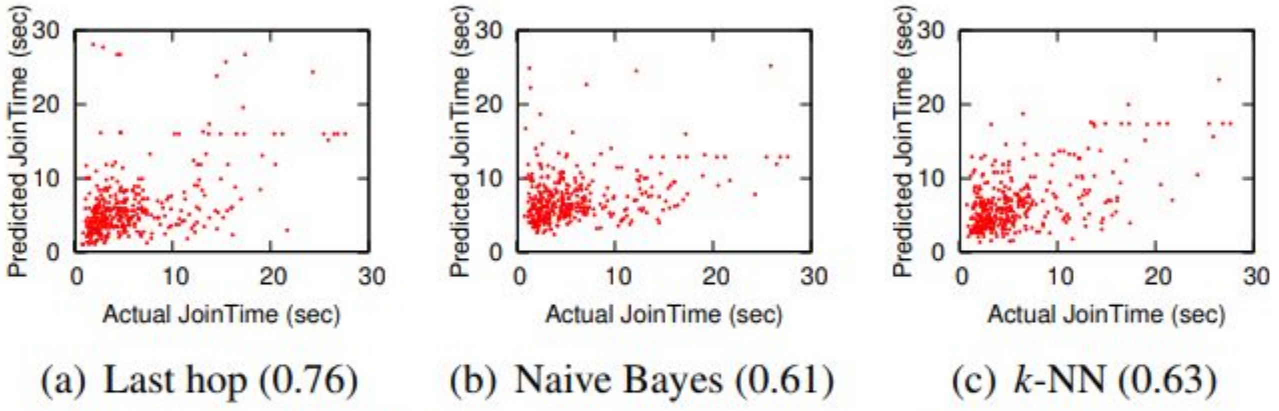
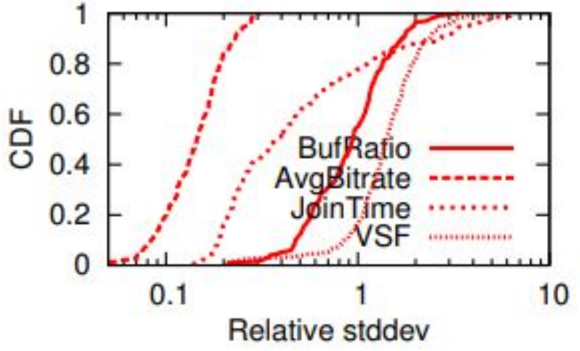


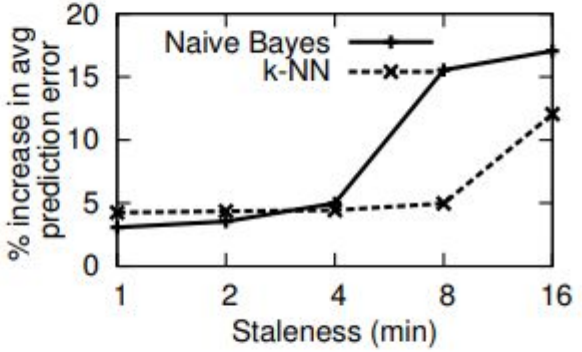
Figure 3: ***Prediction error of some existing solutions is substantial (mean of relative error in parentheses).***

To see why existing solutions are not sufficient, let us consider the k-NN. 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.

Challenge 2: Fresh updates



(a) Temporal variability



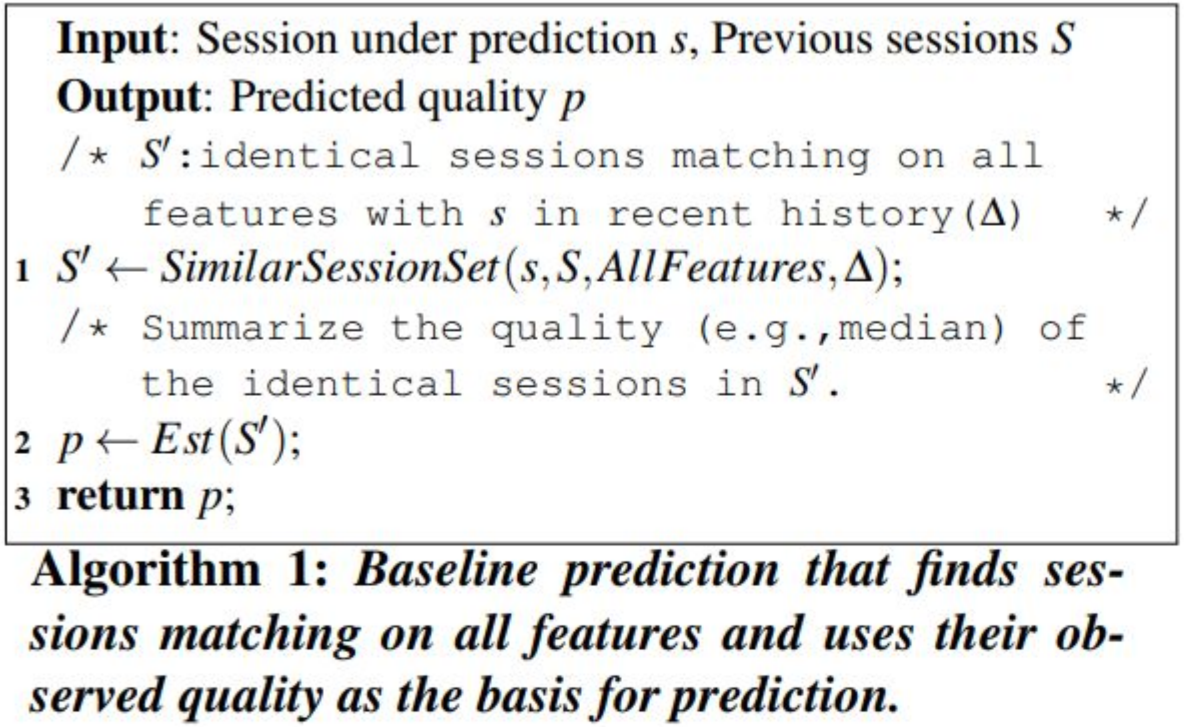
(b) Impact of staleness on accuracy

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

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.

CFA



```
Input: Session under prediction $s$, Previous sessions $S$
Output: Predicted quality $p$
/* $S'$:identical sessions matching on all
   features with $s$ in recent history($\Delta$)   */
1 $S' \leftarrow SimilarSessionSet(s, S, AllFeatures, \Delta)$;
/* Summarize the quality (e.g.,median) of
   the identical sessions in $S'$.                  */
2 $p \leftarrow Est(S')$;
3 return $p$;
```

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

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

CFA

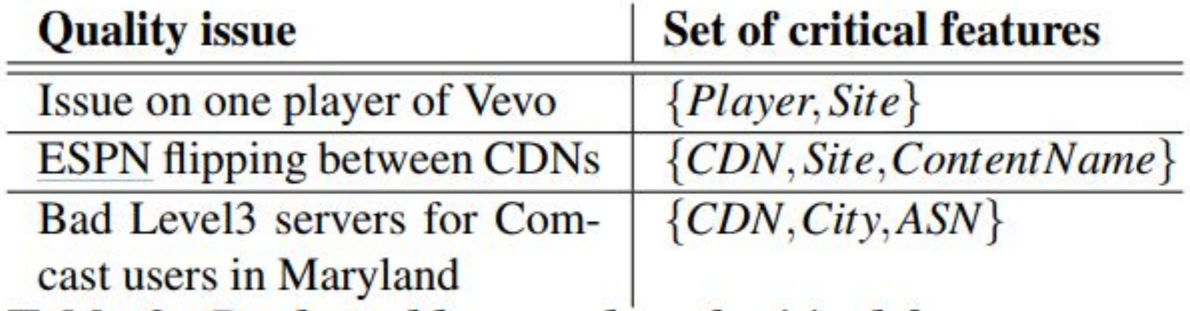
This algorithm is unreliable as it suffers from the classical curse of dimensionality.

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

CFA

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



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.***

Algorithm 2 presents a logical view of this idea:

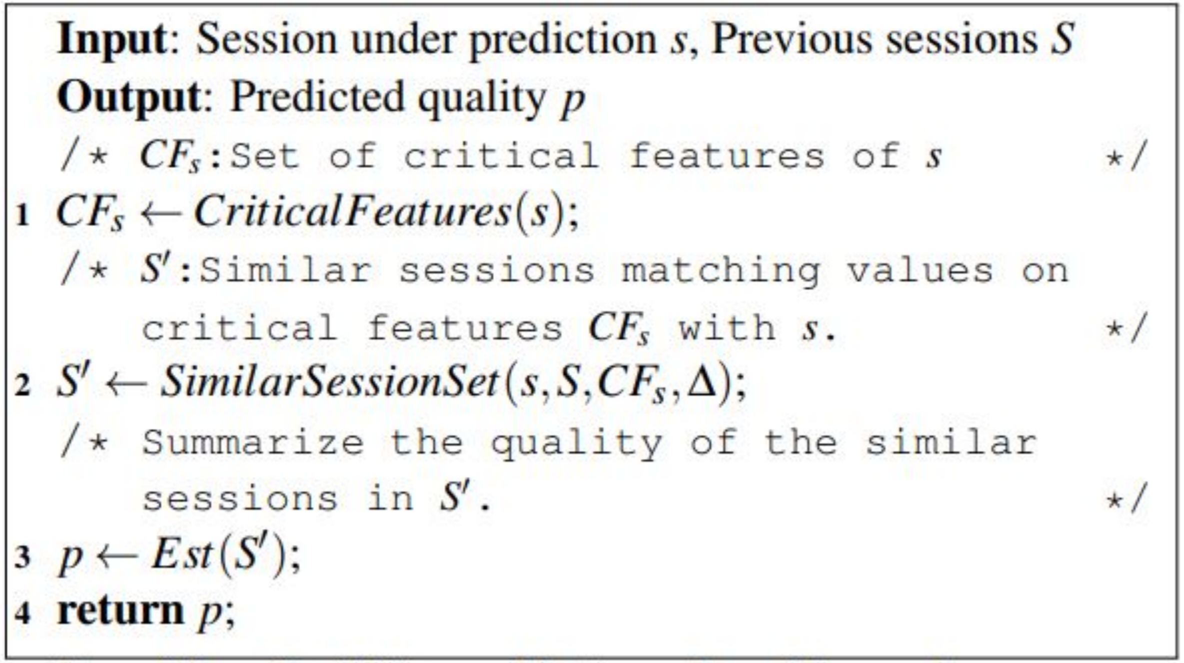
1. Critical feature learning (line 1):

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 (for BufRatio, AvgBitrate, JoinTime) or the mean (for VSF).



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

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

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, and it is infeasible to manually configure critical features.

Practical challenges

There are two issues in using Algorithm 2.

How to reduce update delay?

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. 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.

Addressing those issues

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.

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

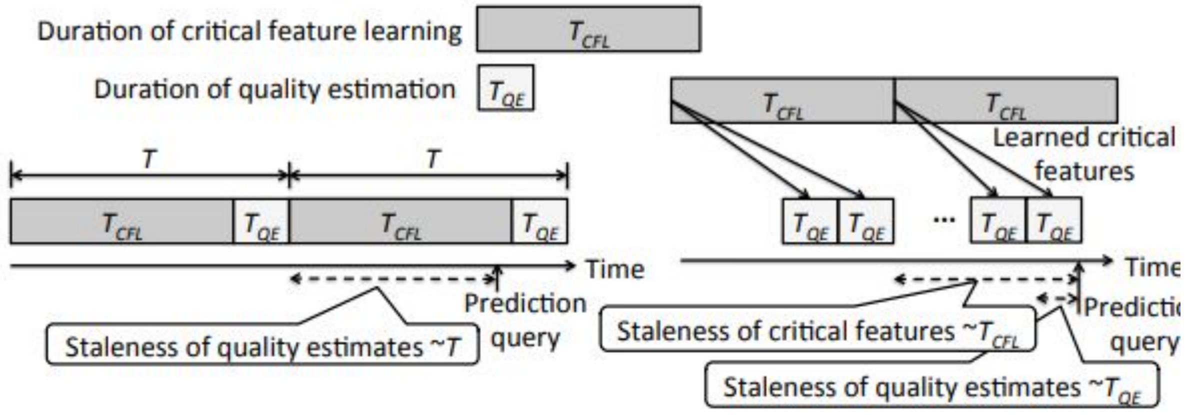
Notations	Domains	Definition
$s, S, \mathbb{S}$		A session, a set of sessions, set of all sessions
$q(s)$	$\mathbb{S} \mapsto \mathbb{R}$	Quality of s
$QualityDist(S)$	$2^{\mathbb{S}} \mapsto 2^{\mathbb{R}}$	$\{q(s) \mid s \in S\}$
$f, F, \mathbb{F}$		A feature, a set of features, set of all features
$CriticalFeatures(s)$	$\mathbb{S} \mapsto 2^{\mathbb{F}}$	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,Δ)	$\mathbb{F} \times 2^{\mathbb{F}} \times \mathbb{S} \times \mathbb{R}^{+} \mapsto 2^{\mathbb{F}}$	$\{s' \mid s' \in S, t(s) - \Delta < t(s') < t(s), FSV(F,s') = FSV(F,s)\}$

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

```
Input: Session under prediction $s$, Previous sessions $S$
Output: Critical features for $s$
/* Initialization */
1  $MaxSimilarity \leftarrow -\infty, CriticalFeatures \leftarrow NULL$;
/* $D_{finest}$:Quality distribution of
   sessions matching on $\mathbb{F}$ in $\Delta_{learn}$. */
2  $D_{finest} \leftarrow QualityDist(SimilarSessionSet(s,S,\mathbb{F},\Delta_{learn}))$;
3  for $F \subseteq 2^{\mathbb{F}}$ do
       /* Exclude $F$ without enough similar
          sessions for prediction. */
4      if $|SimilarSessionSet(s,S,F,\Delta)| < n$ then
5          continue;
       /* $D_F$:Quality distribution of
          sessions matching on $F$ in $\Delta_{learn}$.
          */
6      $D_F \leftarrow QualityDist(SimilarSessionSet(s,S,F,\Delta_{learn}))$;
       /* Get similarity of $D_{finest}$ & $D_F$. */
7      $Similarity \leftarrow Similarity(D_F, D_{finest})$;
8      if $Similarity > MaxSimilarity$ then
9          $MaxSimilarity \leftarrow Similarity$;
10         $CriticalFeatures \leftarrow F$;
11 return $CriticalFeature$;
```

Algorithm 3: *Learning of critical features.*

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


Duration of critical feature learning
T_{CFL}
Duration of quality estimation
T_{QE}
T
T
T_{CFL}
T_{QE}
T_{CFL}
T_{QE}
Time
Prediction query
Staleness of quality estimates ~T
T_{CFL}
T_{CFL}
Learned critical features
T_{QE}
T_{QE}
...
T_{QE}
T_{QE}
Time
Prediction query
Staleness of critical features ~T_{CFL}
Staleness of quality estimates ~T_{QE}

(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.***

Putting it together

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.

D_finest by the set of n sessions that share most features with s over the time window of ∆learn.

Formally, we use {s’ |s’ matches k_s features with s}, where k_s = argmin_k(|{s’ |s’ matches k features with s| ≥ n}|).

Putting it together

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

Putting it together

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.

Implementation and Deployment

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

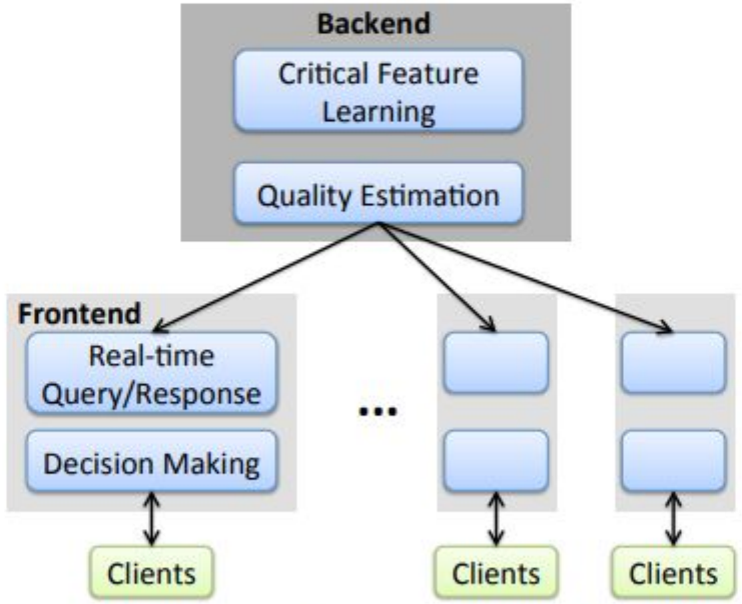


Figure 6: ***Implementation overview of CFA. The three stages of CFA workflow are implemented in a backend cluster and distribute frontend clusters.***

Implementation and Deployment

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.

Implementation and Deployment

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.

Evaluation

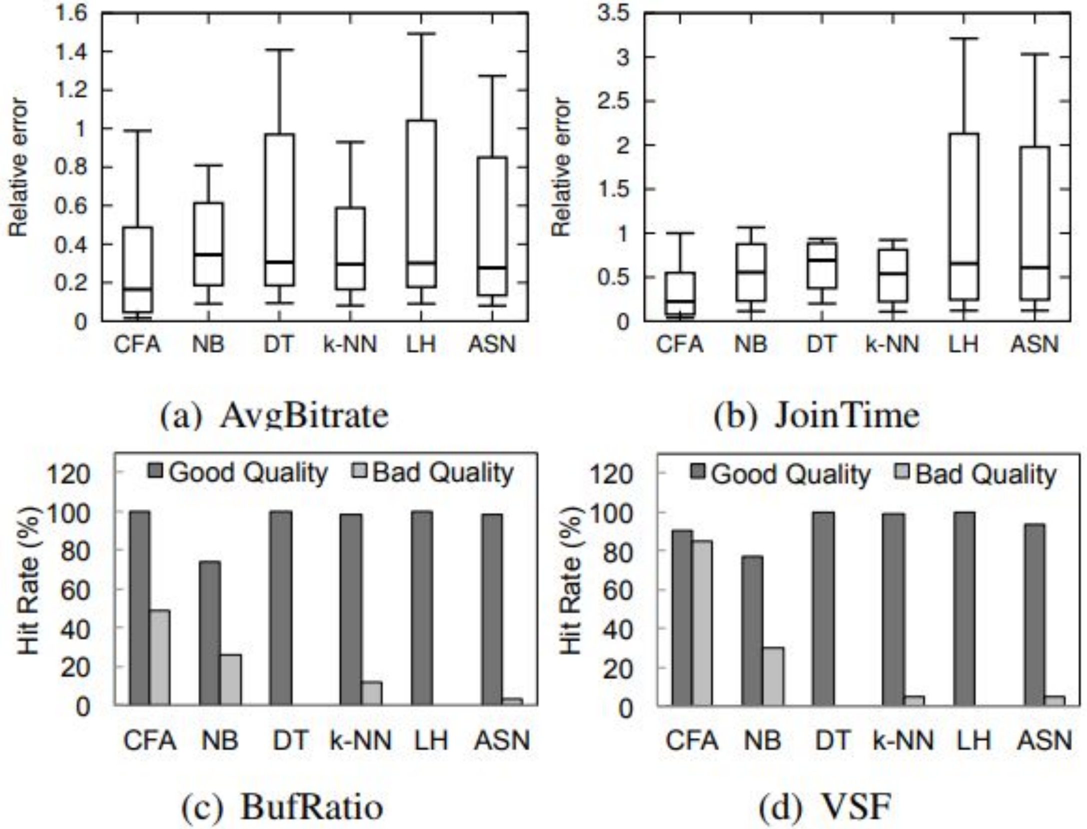


Figure 8: ***Distributions of relative prediction error (*$\{5, 10, 50, 90, 95\}$*%iles) on AvgBitrate and JoinTime and hit rates on BufRatio and VSF. They show that CFA outperforms other algorithms.***

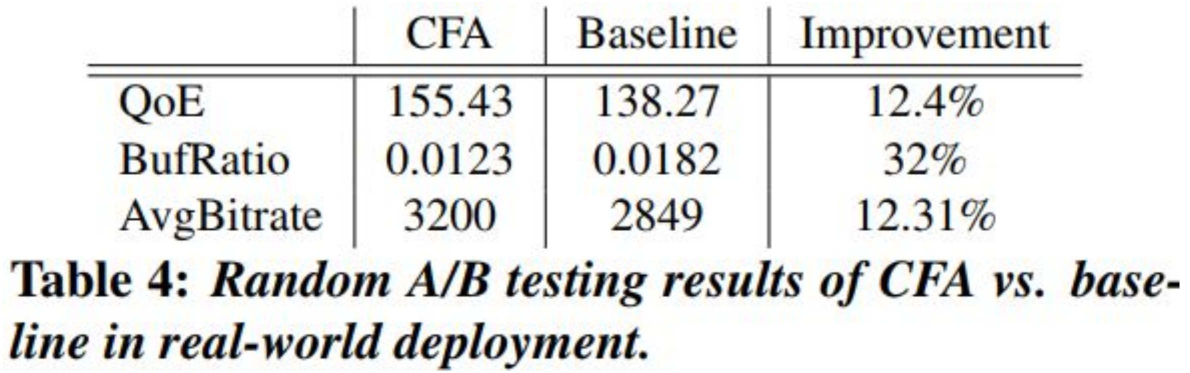
Evaluation

CFA predicts video quality with 30% less error than competing machine learning algorithms.

Using CFA-based prediction, we can improve video quality significantly; e.g., 32% less BufRatio, 12% higher AvgBitrate in a pilot deployment.

CFA is responsive to client queries and makes predictions based on the most recent critical features and quality measurements.

Evaluation



	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.***

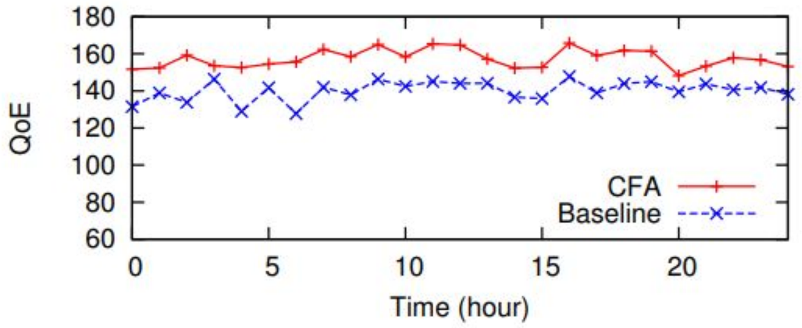
As a pilot deployment, we integrated CFA in a production system that provides a global video optimization service. We deployed CFA on one major content provider and used it to optimize 150,000 sessions each day. We ran an A/B test (where each algorithm was used on a random subset of clients) to evaluate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

Evaluation

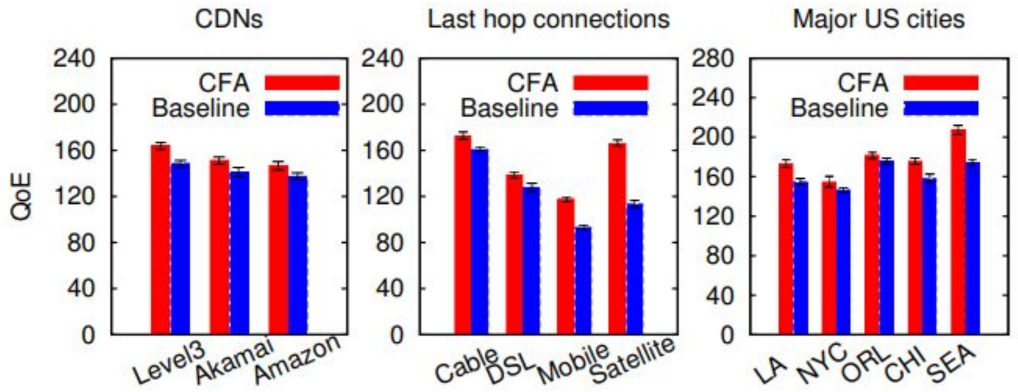
Table 4 compares CFA with the baseline random decision maker in terms of the mean BufRatio, AvgBitrate and a simple QoE model **(QoE = $-370 * \text{BufRatio} + \text{AvgBitrate}/20$)**

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.

Evaluation



(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).***

Evaluation

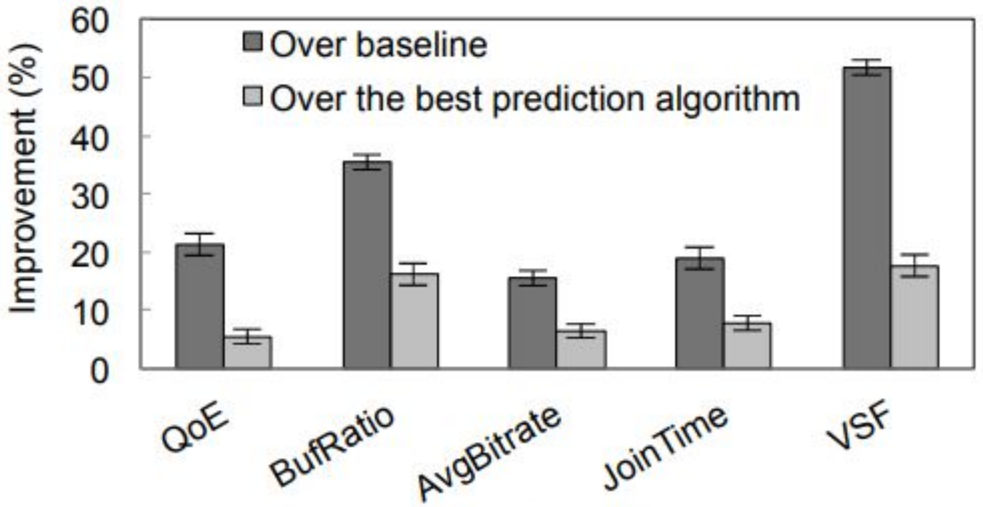
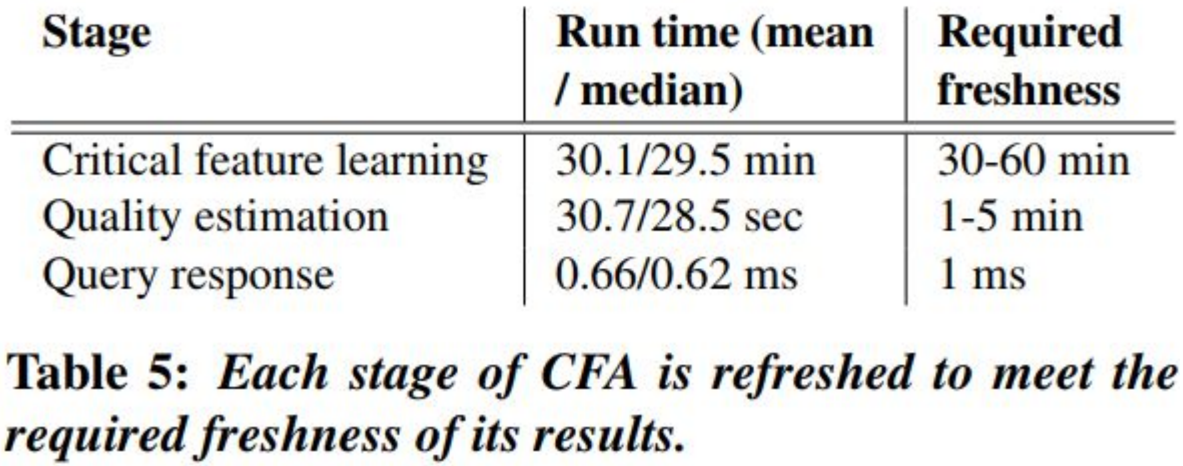


Figure 10: ***Comparison of quality improvement between CFA and strawmen.***

Evaluation



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.***

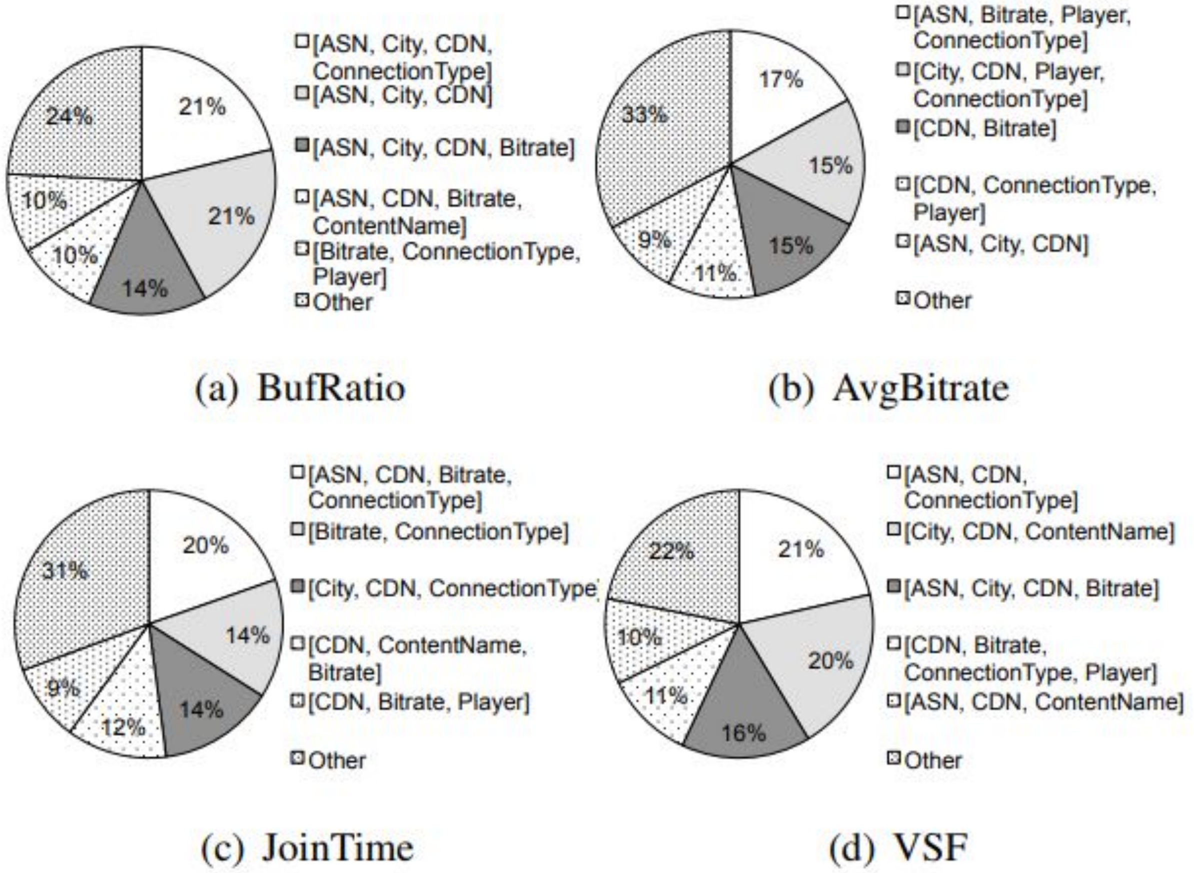


Figure 12: ***Analyzing the types of critical features: This shows a breakdown of the total number of sessions assigned to a specific type of critical features.***

	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.***

Conclusions

Many prior research efforts posited that quality prediction could lead to improved QoE.

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.

Conclusions

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.