Reparo: QoE-Aware Live Video Streaming in low-rate Networks by Intelligent Frame Recovery

Fulin Wang
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
wfl22@mails.tsinghua.edu.cn

Qing Li*
Peng Cheng Laboratory
Shenzhen, China
liq@pcl.ac.cn

Wanxin Shi
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
shiwx17@mails.tsinghua.edu.cn

Gareth Tyson
Hong Kong University of Science and Technology(GZ)
Guangzhou, China
gtyson@ust.hk

Yong Jiang
Tsinghua Shenzhen International Graduate School
Peng Cheng Laboratory
Shenzhen, China
jiangy@sz.tsinghua.edu.cn

Lianbo Ma
Northeastern University
Shenyang, China
malb@swc.neu.edu.cn

Peng Zhang
Tencent Cloud
Shenzhen, China
rocalzhang@tencent.com

Yulong Lan
Tencent Cloud
Shenzhen, China
dragonlan@tencent.com

Zhicheng Li
Tencent Cloud
Shenzhen, China
vultureli@tencent.com

ABSTRACT

Live video streaming has grown dramatically in recent years. A key challenge is achieving high video quality of experience (QoE) in low-rate networks. To tackle this problem, recent streaming approaches strategically drop video frames, thus reducing the bandwidth required. However, these methods are usually designed for video on demand (VoD) services and perform poorly in live video streaming. In this paper, we design a new live video streaming approach, Reparo, which aims to improve users' QoE in low-rate networks. On the upload client side, Reparo discards video frames such that they are never encoded or transmitted. To decide which frames should be dropped, we design a real-time Video Frame Discarding (VFD) model, which strives to minimize the impact on video quality while maximizing bandwidth savings. To complement this, Reparo further proposes a modified adaptive bitrate algorithm and two encoding modes, targeting low-frame-rate encoding. On the server side, Reparo then recovers the dropped frames using a lightweight Video Frame Interpolation Deep Neural Network (VFI-DNN). Experimental results show that, compared with vanilla DASH, Reparo reaches an SSIM gain of 0.018, or reduces bandwidth consumption by 30.86%. With an average bandwidth of 0.974Mbps, it improves QoE by 18.13% on average compared to DASH.

*Qing Li is the corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3613441

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Networks** → *Network resources allocation.*

KEYWORDS

Live video streaming; Frame Drop; Frame Interpolation

ACM Reference Format:
Fulin Wang, Qing Li, Wanxin Shi, Gareth Tyson, Yong Jiang, Lianbo Ma, Peng Zhang, Yulong Lan, and Zhicheng Li. 2023. Reparo: QoE-Aware Live Video Streaming in low-rate Networks by Intelligent Frame Recovery. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3581783.3613441

1 INTRODUCTION

Live video streaming, accounting for 17% of Internet traffic in 2022 [9], consists of three components: the upload client, the media server, and the end-user clients. The upload client first encodes video frames received from a camera [4, 7, 34, 46, 50], and then transmits them to the media server through real-time streaming protocols [1, 13, 37, 38]. Next, the media server decodes and processes the received video, and distributes it to the end user clients. Finally, the end users receive the video, often using different bitrates determined by various adaptive bitrate (ABR) algorithms [20, 24, 30, 33, 44, 53, 56]. This process is rife with challenges though. For example, the uplink bandwidth from the upload client to the media server is often insufficient [14, 61, 62]. As a result, the upload client may have to encode high-quality video frames with a lower bitrate, thereby reducing users' QoE.

There has been much work in the *spatial domain*, attempting to optimize the frame *encoding* process in live video delivery. To save the uplink bandwidth, video coding approaches optimize the video

compression process [2, 19]. Alternatively, recent super-resolution approaches downsample the video to be transmitted and enhance video quality through image enhancement [6, 18, 25, 28, 31, 32]. Nevertheless, these methods can fall short under low-rate networks.

This leads to exploration in the *time domain*, focusing on optimizing *frame rate*. Two recent studies, BETA [22] and VOXEL [36], try to improve video delivery by dropping frames. BETA strategically discards B-frames, while VOXEL drops P-frames and B-frames packets through the QUIC protocol [26]. Nonetheless, BETA and VOXEL, designed for on-demand streams, take too long to determine the frames to drop. As a result, these existing methods cannot be used for live video streaming, which has stringent latency requirements.

With these limitations in mind, we propose *Reparo*, a new live video streaming system that enhances video transmission by strategically discarding video frames. Reparo is deployed on both the upload client and the server. On the upload client, the differences between adjacent frames are extracted and fed into a **V**ideo **F**rame **D**iscarding (**VFD**) model. This model determines whether an intermediate frame between its two adjacent frames should be dropped or not. After the frame(s) to drop have been selected, the upload client determines the encoding bitrate using an ABR algorithm [56] modified for live video streaming. It then chooses an appropriate low-frame-rate encoding mode. Reparo proposes two modes, called *Hbit* and *BWSave*, which we switch between based on network conditions. The *Hbit* mode aims to improve the per-frame bitrate of the video while keeping a similar uplink bandwidth overhead, while the *BWSave* mode aims to save the uplink bandwidth without bringing severe quality degradation. On the server side, after the video frames are decoded, DNN-based interpolation is run alongside the VFD model update. Following this, the updated VFD model is sent back to the upload client.

There are several challenges when applying the frame-drop & frame-interpolation methods to live video delivery:

(1) The impact of a frame drop will vary considerably based on the frame selected. Thus, the video quality impact of a frame drop needs to be predicted in real-time at the upload client. However, the limited computing power of the upload client might not be able to perform rapid DNN-based frame interpolation. This will negatively influence the calculation of the video quality for the frame-drop & frame-interpolation process.
(2) The real-time impact of the different low-frame-rate coding modes (*Hbit* and *BWSave*) cannot be assessed in advance. Thus, it is difficult to combine the low-frame-rate coding strategies with the modified ABR algorithm for encoding bitrate selection.
(3) Since the performance of the pre-trained VFD model decays with the changing of video content, the server requires consecutive video frames to assess the effectiveness of **V**ideo **F**rame **I**nterpolation **D**eep **N**eural **N**etwork (VFI-DNN), in order to generate a dataset for updating the VFD model. However, with the limited uplink bandwidth, it is difficult to upload dropped video frames to the media server to construct this dataset.

Reparo addresses the above challenges. *First*, Reparo proposes a lightweight VFD model for the upload client, trained using historical live video. This model takes the inter-frame differences as an input and selects the best frames to drop, according to whether the server's interpolation will be effective enough. Accordingly, the inference of this lightweight model could replace the heavy computation of the DNN on the upload client.

Second, Reparo combines the extra quality gain or saved bandwidth achieved through the two low-frame-rate modes (which we term *Hbit* and *BWSave*) with network conditions. By considering the bandwidth prediction of a modified ABR algorithm [56], it dynamically selects the preferred mode to match network conditions.

Third, to continually improve performance, Reparo dynamically re-trains the VFD model at the server, using the received incomplete frames. It analyzes the differences between equally spaced video frames and checks the VFI-DNN performance to identify the frames which are successfully and unsuccessfully reconstructed. Using this binary set, the server then periodically re-trains the VFD model.

We evaluate Reparo using six different types of video content and two types of network traces. We compare Reparo-related schemes against the state-of-the-art (i.e., BETA, VOXEL and DASH) modified for live video streaming. For different bitrate levels, Reparo can achieve an average structural similarity (SSIM) gain of 0.018, or a bandwidth saving ratio of 30.86%. When a commonly-used ABR algorithm [56] is modified and utilized to select the encoding bitrate and the encoding mode for the upload client, Reparo outperforms DASH, BETA and VOXEL by 9.28 $\sim$ 18.13% in terms of QoE. In summary, we make the following contributions:

- We implement and evaluate Reparo, a live video streaming system that enhances video transmission by strategically discarding video frames.
- We propose a VFD model, which takes the features across consecutive frames as an input, and selects frames to drop based on VFI-DNN's predicted effectiveness. This takes just 213ms-593ms on various smartphones.
- We propose two encoding modes to better match the selected bitrate to the predicted bandwidth. The two modes bring 0.018 SSIM gain and 30.86% bandwidth saving ratio respectively. They improve QoE by 9.28 $\sim$ 18.13% over the baselines.
- Reparo infers VFI-DNN in real-time to recover dropped frames, and provides VFI-DNNs with fewer layers for limited computing power.
- We propose a mechanism to update the VFD model in an online fashion, using the incomplete frames received on the server. This results in 22.96% additional quality gain and 46.53% additional bandwidth savings for two modes respectively.

2 RELATED WORK

Live Video Streaming Optimization. The main bottlenecks of live video delivery are driven by limited uplink bandwidth and the associated real-time frame delivery requirements. Recent works focus on improving video coding and using super-resolution (in the spatial domain) to enhance the performance of live video delivery.

Salsify [16] integrates encoders with transport protocols, and Dave [19] uses reinforcement learning for encoding. Our proposal simplifies these methods by dropping frames before encoding to increase the average bitrate or save bandwidth without significant quality loss. This lightweight approach is real-time capable, even on low-resource devices.

In terms of enhancing video quality, super-resolution (SR) techniques have been applied. LiveNAS [25] updates the SR DNN with

high-resolution patches, LiveSRVC [6] compresses key frames for SR inference, and NeuroScaler [55] selects frames for SR DNN real-time reuse. Unlike these spatially focused solutions, our Reparo design emphasizes time-based resolution or frame rate, and can work in conjunction with SR methods to boost overall performance.

Frame/Packet drop in video delivery. Similar to us, there are existing works that optimize video delivery by dropping packets or video frames. Yahia et al. [52] drops video frames in live video streams using HTTP/2 with a fixed bitrate. Stewart et al. [45] modify the SCTP protocol to be partially reliable, thus dropping a portion of the packets without taking the characteristics of video frames into account. BETA [22] regards all B-frames as discardable, and VOXEL [36] modifies all P-frames and B-frames to be transmitted unreliably. The above schemes provide inspiration, yet none of them is well suited to the scenario where the uplink bandwidth is severely constrained in *live* video streaming.

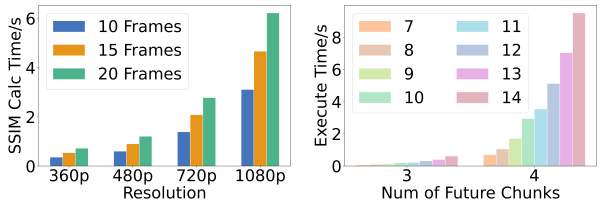
Video Frame Interpolation. Video frame interpolation (VFI) aims to create an intermediate frame between two adjacent ones to maintain coherence. Traditional VFI strategies are mainly based on optical flow [5, 17, 42]. Recently, deep neural network-based approaches [12, 21, 23, 35, 41] have shown increasingly powerful performance. In addition, lightweight video frame interpolation neural networks [11, 27] have also achieved good performance compared to heavier ones. We posit that this offers the potential to optimize video delivery systems.

3 CHALLENGES

3.1 Limitations of Frame Drop Approaches

Unlike on-demand video streaming, live video streaming has strict latency requirements. Unfortunately, existing frame drop schemes have high latency, as they must estimate the impact of frame drop on QoE. Furthermore, these methods drop frames *after* video encoding, which leads to a larger decision space for the encoding bitrate selection process and consequently slows it down. The decision space expands since it now includes the selection of the number of video frames to discard in addition to the original bitrate levels.

Latency of Selecting Frames to Drop. We use VOXEL as an example to highlight the additional latency introduced by selective frame drops, since VOXEL uses the SSIM value compared to the original dropped frames as a threshold to select unimportant frames. Suppose that there are n frames that could be dropped in a video chunk, and VOXEL drops frames at the granularity of a single frame. To measure the QoE variation of dropping different numbers of unimportant frames, it is necessary to decode the chunk and compare SSIM frame-by-frame. However, in the above process, frequently calculating the quality of the video chunks will introduce a large latency. Figure 1(a) shows the average execution times for calculating SSIM with different video resolutions. The execution times are measured on 100x 1 second video chunks taken from YouTube live videos, with corresponding original video frames before dropping as reference. We use an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz, and a C++ implementation. We see that when dropping $10 \sim 20$ frames in a 1s video chunk, even the calculation under 480p resolution will take $0.61 \sim 1.20s$, which is undesirable for live



(a) Time Cost of Calculating SSIM (b) Time Cost of MPC with Different Sizes of Decision Spaces

Figure 1: Existing Methods Bring Significant Latency

video streaming. This makes rapidly calculating the QoE variation caused by discarding frames a significant challenge.

Inefficient Bitrate Selection. VOXEL enlarges the decision space for the encoding bitrate selection algorithm at the upload client side. This is because the number of discarded video frames is not determined before executing the selection algorithm. Thus, traditional bitrate selection algorithms could be undermined by unanticipated frame drops, and the execution efficiency of both traversal-based and reinforcement learning-based bitrate selection algorithms is reduced. This fails to meet the low latency requirements of live video streaming. Moreover, VOXEL's frame drop granularity is actually at the packet level, and all P-frames in VOXEL are discardable, resulting in more complex decision space.

Figure 1(b) demonstrates that MPC [56], a traversal-based ABR algorithm, has a considerable latency overhead due to an expansion of the decision space. This implies that fewer future chunks can be traversed, and the performance of the algorithm will decline when used for bitrate selection. As a result, it is challenging to make the decision space of the live video coding bitrate selection algorithm not become inflated after frame drops with existing approaches.

Deprivation of the Chance to Increase Per-Frame Encoding Rate. Unlike dropping frames before encoding, the method of discarding frames after encoding sometimes scarcely improves video quality. Nevertheless, live video streaming with constrained uplink bandwidth often encounters persistently low video bitrates, which results in poor visual quality. If frames are dropped *before* encoding, the average encoding bitrate of the remaining frame can be increased to obtain higher-quality video frames with the same bandwidth consumption. In summary, approaches like BETA and VOXEL are unable to improve the quality of the video in low-rate networks directly.

3.2 Limitations of Frame Recovery Methods

Existing methods to recover missing frames, such as frame interpolation, are straightforward. However, they may degrade the overall video quality. Recently, lightweight video interpolation Deep Neural Networks (DNNs) [11, 27] offer the possibility to recover discarded video frames in real-time and with high quality.

Naive video recovery lowers visual quality. We use VOXEL as an example to show how simple video recovery techniques can lower visual quality. VOXEL performs packet-level frame drops, so frames may be partially or entirely lost. For both cases, VOXEL adopts the strategies of substituting dropped packets with zeroes and replacing the dropped frames with the previous frame. If many

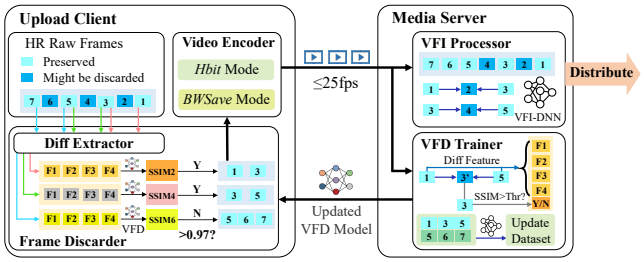


Figure 2: Reparo System Overview

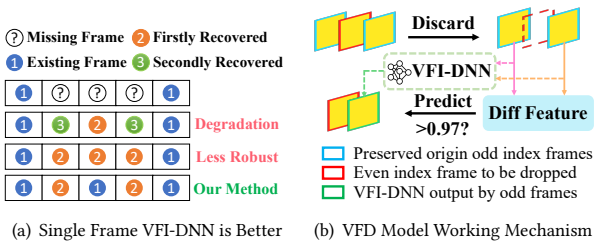


Figure 3: VFD-VFI Mechanism on the Client Side

adjacent frames have salient content changes, the visual quality will degrade significantly after frame drop and recovery. Lightweight VFI-DNNs [11, 27] can only recover quality when the input frames don't have drastic scene changes.

VFI-DNNs cannot handle everything. Although VFI-DNNs show better performance than traditional methods like optical flow-based ones [5, 17, 42], VFI-DNNs are not capable of completely recovering every randomly discarded video frame. For example, the VFI-DNN is less effective when there are drastic scene changes from one input frame to the next. Hence, accurately discarding unimportant frames for VFI-DNN to recover is a challenge.

4 REPARO: SYSTEM OVERVIEW

The main goal of Reparo is to make better utilization of the limited uplink bandwidth at the upload client side. Figure 2 shows the complete architecture of Reparo. It consists of an upload client and a media server.

Upload Client. Unlike traditional upload clients, in Reparo, the client drops unimportant frames and encodes the remaining frames with a lower frame rate. Two design components operate at the client to maximize the end users' QoE. The Frame Discarder extracts the frame difference features and uses them to decide which frames to drop. This is embedded in a Video Frame Discarding (VFD) model, which performs a binary classification for each frame (to drop or not). The video encoder first adjusts the ABR algorithm by using the average value of the video chunks' size and quality to replace information about future video chunks for selecting the encoding bitrate level. This makes the ABR algorithm available for encoding bitrate selection at the upload client. Next, Reparo proposes two low frame rate encoding modes to accommodate network conditions, utilizing different bitrates per frame for video encoding.

Server. Upon receiving frames, the server executes the VFI-DNN to recover any dropped frames after decoding the received video chunks. It then retrains the VFD model only with the incomplete frames it has received. To be more specific, the VFI processor recovers dropped frames in real-time and provides VFI-DNNs with different numbers of convolution-deconvolution pairs to better adapt to computing resources. The VFD trainer utilizes two kinds of consecutive frames to generate the dataset for VFD model updating. After re-training and updating the model, the server sends the new VFD models back to the client so they can make better frame-drop decisions.

5 UPLOAD CLIENT DESIGN

5.1 Frame Discarder

The role of the frame discarder is to select the optimal frames to drop. It calculates frame difference features to measure the scene change. It then constructs a binary classifier using these features to estimate the server's VFI-DNN effectiveness. We therefore strive to select frames that can be recovered effectively by the server's VFI-DNN. Below, we list the steps performed by the Frame Discarder.

Extracting frame difference features. Reparo first extracts features that capture the difference between adjacent frames. Figure 3(a) shows the strategies of dropping consecutive frames and dropping only even-indexed frames. When multiple consecutive frames are dropped, we can either use multi-frame recovery VFI-DNNs once, or single-frame recovery VFI-DNNs multiple times. However, the former is less adaptive to scene changes, while the latter leads to greater quality degradation. As a result, we only permit dropping even-indexed frames. To predict whether each even-indexed frame can be recovered effectively by the VFI-DNN on the server, we measure the degree of scene change between its adjacent frames in real-time. Inspired by Reducto [29], we perform this comparison using four low-level disparity features that are represented as four floating point numbers: pixel difference, edge difference, area difference, and grayscale histogram difference. The time required for the extraction of the four features on 1080p video frames can easily meet the real-time requirements of 12fps since only the differences between odd-indexed frames are extracted.

Measuring interpolation effectiveness. After extracting the above features, we determine whether the performance degradation of using the VFI-DNN (due to scene changes) is acceptable or not. The efficacy of the VFI-DNN in recovering video frames can be directly assessed using the SSIM value, with the original frame serving as a reference. Consequently, an SSIM threshold can be directly selected to ascertain the validity of the VFI-DNN approach. Given that the state-of-the-art convolutional neural network-based single-frame VFI-DNN, AdaCoF [27], has achieved an SSIM value of 0.97 on two widely-used datasets for video frame interpolation tasks (Middlebury [3] and UCF101 [43]), we set an SSIM threshold of 0.97. This threshold determines whether the output of VFI-DNN might compromise the overall quality of an individual video frame.

Identifying frames to drop. Based on the above, we next train a binary classifier to select the frames to drop. The classifier's goal is to predict if a dropped frame could be recovered by the VFI-DNN, while also reaching the minimum SSIM threshold.

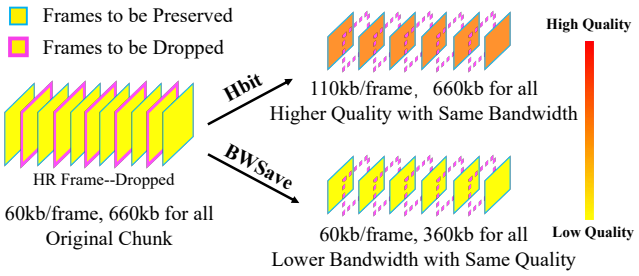


Figure 4: An Example of the Hbit and BWSave Modes

Given two odd-indexed input frames, F_m and F_n, assume that the even-indexed frame between them is F_p, and the output of the VFI-DNN when taking F_m and F_n as input is F'_p. Then, a binary classifier from the inter-frame feature differences to the decision of dropping (or preserving) F_p could be established as follows:

$$Diff(F_m, F_n) \rightarrow SSIM(F_p, F'_p) > 0.97? \quad (1)$$

The binary classifier described in Equation 1 defines our Video Frame Discarding (VFD) model, and Figure 3 (b) illustrates its operational mechanism. To ensure the efficient implementation of our VFD model and its ability to meet the real-time demands of live video streaming, we propose two design strategies. *First*, we employ a lightweight Multi-Layer Perceptron (i.e., 2 layers) model to represent the VFD binary classifier. This streamlined model can readily perform real-time inference with limited computational resources on the upload client. *Second*, we select the most recent (i.e., previous 5 minutes) video of the same live channel to pre-train the VFD model offline. This is because, for a single live channel, its previously broadcast videos tend to have greater similarity with the current live video content (than unrelated videos). Thus, these strategies ensure that the upload client can better approximate the impact of dropping a specific frame on video quality.

5.2 Video Encoder

The role of the video encoder is to select the optimal encoding bitrate after a frame drop. After identifying frames that can be discarded, video frames can naturally be encoded at a reduced frame rate. Initially, Reparo's video encoder modifies MPC's [56] input features by substituting the quality and size of future video chunks for their average value. The ABR algorithm then uses these inputs to select the level of encoding bitrates for live video streaming. Subsequently, it more effectively adapts to the gap between bandwidth prediction and the chosen bitrate level through two low frame rate encoding modes, *Hbit* and *BWSave*, which are used when the predicted bandwidth is higher or lower than the selected bitrate level respectively. To be more specific, the two modes help to improve the video quality or save the uplink bandwidth by adjusting the encoding bitrate per frame.

Encoding bitrate level selection. First, Reparo needs to choose the optimal encoding bitrate, which must consider information unavailable in live video streaming. As examples, let us take adaptive bitrate (ABR) algorithms like MPC [56] and Pensieve [33], which are used in on-demand video streaming. They rely on upcoming video chunk size and quality, historical network bandwidth, and other factors to select the best bitrate level. However, live streaming generates video content in real-time, making future video chunk size and quality unknown. To compensate, Reparo modifies the MPC algorithm, by using the average size and quality for each bitrate level as a substitute for future video chunk information. This modified MPC selects the encoding bitrate level for the upload client in live video streaming.

Encoding mode adaptation. Due to real-time constraints, Reparo selects the encoding bitrate only from a limited set of levels. This can lead to a potential gap between selected bitrate levels and predicted uplink bandwidth. To fill this gap, we propose two low frame rate coding modes: *Hbit* and *BWSave*. For *Hbit* mode, by keeping the total encoding bitrate constant before and after frame dropping, the average bitrate of preserved frames increases after dropping frames. This leads to higher quality for the remaining frames compared to the regular encoding strategy. For *BWSave* mode, aiming at a constant average bitrate of the preserved frames before and after frame dropping results in lower bandwidth consumption.

Figure 4 shows an example of the *Hbit* and *BWSave* modes. Suppose that d frames are dropped in a video chunk with n frames, and the selected encoding bitrate level is R_0. Then, the per-frame bitrate under *Hbit* mode and *BWSave* mode is $\frac{R_0}{n-d}$ and $\frac{R_0}{n}$ respectively, and the chunk's overall encoding bitrate under *Hbit* mode and *BWSave* mode can be calculated accordingly as R_0 and $\frac{(n-d)R_0}{n}$.

Intuitively, our two coding modes can be combined with bandwidth prediction methods in modified MPC [56]. If the predicted bandwidth is above the selected bitrate level, we would lean towards selecting the *Hbit* Mode for higher quality, and conversely, we would opt for the *BWSave* Mode to minimize the likelihood of rebuffering.

6 MEDIA SERVER DESIGN

6.1 VFI Processor

The role of the VFI Processor is to implement the VFI-DNN to restore the decoded low frame rate video chunks to their original frame rate of 25fps.

Recovering video frames under weak network conditions. When receiving low frame rate video, the VFI-DNN is used to restore the frame rate to 25 fps in real-time. For 480p, it recovers frames sequentially, but for 720p, Reparo uses multiple GPUs for parallel inference, distributing frames equally among 3 GPUs to speed up the process. It should be noted that Reparo does not support 1080p video, as its 4Mbps transmission requirement is sufficient to transmit video at the highest bitrate level.

Conv-Deconv Pair Num	2	3	4	5
Max Feature Channel	64	128	256	512
Memory/MB	4.4	11	34	84
SSIM-test	0.8058	0.9062	0.9105	0.9101

Table 1: Different Levels of VFI-DNNs

VFI-DNN support for different computing resources. Server resources can fluctuate when there are multiple streams, possibly

limiting original VFI-DNN inference. To address this, Reparo reduces VFI-DNN memory needs by removing some convolutional and deconvolutional blocks from AdaCoF [27]. Table 1 shows memory usage and AdaCoF performance with different conv-deconv pairs removed. Performance stays robust with more than two pairs remaining, so Reparo may use 3 or 4 pairs for real-time frame recovery when resources are limited.

6.2 VFD Trainer

The role of the VFD trainer is to update the VFD model according to changing video content. Despite similarities between historical and current live videos, these decrease over time. This requires server-side resources to infer VFI-DNN and obtain the dataset for updating the VFD model.

Algorithm 1: VFD Update Dataset Generation

Input: Received frames in a chunk F, Indices of frames Ind
Output: Dataset D

1 Initialize $D = []$;
2 **for** F_{i-1}, F_i, F_{i+1} *in* F **do**
3 **if** $Ind_{i+1} - Ind_i = 2$ ***and*** $Ind_i - Ind_{i-1} = 2$ **then**
4 $x_1 = Diff(F_{i+1}, F_i), x_2 = Diff(F_i, F_{i-1})$;
5 $F'_i = VFI(F_{i+1}, F_{i-1})$;
6 **if** $SSIM(F_i, F'_i) > 0.97^2$ **then**
7 $y_1 = y_2 = 1$;
8 **else**
9 $y_1 = y_2 = 0$;
10 **end**
11 Add $[x_1, y_1]$ and $[x_2, y_2]$ to D;
12 **else if** $Ind_{i+1} - Ind_i = 1$ ***and*** $Ind_i - Ind_{i-1} = 1$ **then**
13 $x = Diff(F_{i+1}, F_{i-1})$;
14 $F'_i = VFI(F_{i+1}, F_{i-1})$;
15 **if** $SSIM(F_i, F'_i) > 0.97$ **then**
16 $y = 1$;
17 **else**
18 $y = 0$;
19 **end**
20 Add $[x, y]$ to D;
21 **else**
22 continue;
23 **end**
24 **end**

Acquiring the update dataset. To retrain the VFD model, it is undesirable to require clients to upload additional training data. Thus, retraining must be based on the incomplete frame sequences received on the server side.

Algorithm 1 details how the training dataset for VFD model updates is created. After receiving an incomplete frame sequence, the server examines every three consecutive frames. When the distances between the middle and end frames are equal, we use the middle frame as a reference to calculate the SSIM of the frame created by applying the VFI-DNN to the two end frames. Then, two given SSIM thresholds for different distances are used to classify positive and negative samples. When the distances are unequal, the data from the three frames is discarded.

Line 3 involves distances of two between the middle frame and the sides, with a relaxed VFI-DNN threshold of 0.97^2. If the SSIM is above this, the difference features between terminal frames are labeled as positive samples; otherwise, they are negative. Line 13 deals with distances of one between the middle frame and the sides. If the SSIM value exceeds 0.97, the differences become a positive sample; otherwise, the sample is labeled negative. Line 22 concerns unequal distances between the middle frame and sides, causing deviations in VFI-DNN on the side frames, rendering these frames unsuitable for training data.

Updating the VFD model and sending it to the client. Using the above method, the media server generates training data and updates the VFD model with three iterations over the given data. It then sends the updated model back to the upload client to replace the outdated VFD model every 3 seconds, which is feasible due to the typically abundant downlink bandwidth from the server to the upload client.

7 EVALUATION

We pose the following evaluative questions. Can Reparo achieve video quality improvements and bandwidth savings in low-bandwidth networks? Does Reparo provide better QoE for end users? How effective are the VFD and VFI-DNN models?

7.1 Evaluation setup

Testbed Setup. Reparo is implemented in Python, and a testbed is built to generate the data for evaluation. Our pipeline is based on Intel(R) Xeon(R) Gold 5218 CPUs @ 2.30GHz, and NVIDIA GeForce RTX 2080 Super GPUs. It should be noted that no GPU resources are provided for the client.

Model Training. For the VFI-DNN training, our VFI-DNNs are all trained on the vimeo-triplet dataset [51] like AdaCoF [27]. The AdaCoF is not updated online as it generalizes well. The VFD model uses a simple MLP with two hidden layers of size 100 and 10. Its size is only 30KB and it only takes 10-50ms to infer on smartphones. We use scikit-learn [39] to implement the model.

Evaluation Videos. We use six different types of 1080p video clips that last at least 10 minutes from YouTube (including Podcast [57], Skit [58] and Sports [59]) and Twitch (including Chatting [47], LoL [48] and Saddummy [49]). They are first transcoded into 1080p video in H.264 software codec [50] with a bitrate of 4.8Mbps and a frame rate of 25fps. The first 5 minutes of the videos are used to train the VFD model, while the remaining part of the videos are used for the video streaming simulation. The VFD model's versatility across scenarios is proved by including the "Skit" YouTube video [58], whose training and evaluation parts are totally different. For each video, we provide seven different bitrates on top of Pensieve [33] for the encoding bitrate selection algorithm. The resolution is set from 240p to 1080p according to [10]. Finally, as live streaming requires shorter buffers than VoD, we set the length of the buffer length to 10s.

Network Traces. To simulate the uplink bandwidth, we use a 4G uplink dataset [40] containing 123 traces with an average bandwidth

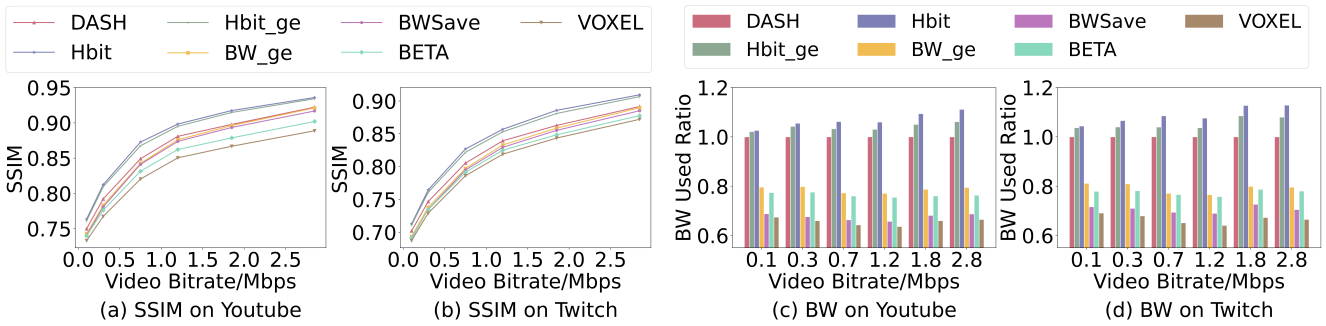


Figure 5: Quality Gain and Bandwidth Consumption of Reparo's Two Modes Across Different Video Bitrates

of 0.617Mbps, and 105 traces from the FCC 2019 dataset [15] with an average bandwidth of 1.391Mbps. It should be noted that the 105 traces from FCC are selected based on the criterion that the average bandwidth is less than 2 Mbps. These two kinds of traces together have an average value of 0.974Mbps and could be used for simulating a bandwidth-constrained environment.

QoE Calculation. To measure the QoE, we rely on the linear QoE model proposed by Pensieve [33]:

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| \quad (2)$$

where μ is set to 4.3 like Pensieve [33], and the gain of SSIM is calculated by its effective bitrate function generated by linear interpolation like NAS [54].

Baselines. To demonstrate that Reparo could enhance users' QoE and video quality, we compare Reparo with some baseline methods under the H.264 codec [50].

- **DASH**: The upload client encodes the original video frames with a bitrate decided by the DASH bitrate adaptation algorithm, and no frame drop or interpolation methods are used.
- **Reparo-g**: The upload client drops video frames with a pre-trained generic VFD model, and recovers them with VFI-DNN. *Hbit* and *BWSave* modes are denoted as **Hbit_ge & BWSave_ge**.
- **BETA-Live & VOXEL-Live [22, 36]**: The upload client encodes every Group of Pictures (GOP) once to acquire the temporal encoding order. Then, an extra choice of dropping the first 50% of the unimportant frames (i.e. B frames for BETA-Live and P, B frames for VOXEL) to save bandwidth will be provided for the encoding bitrate selection algorithm.
- **VFI-only**: The upload client drops all the even-indexed frames, and utilizes the VFI-DNN to reconstruct these dropped frames.
- **VFD-only**: The upload client makes use of our VFD model to identify the frames that could be dropped, but the frames are simply replaced by the last frame in front of it.

7.2 Results

We use SSIM, bandwidth save ratio and QoE as our metrics. The first two are utilized for the evaluation of Reparo's two fixed modes separately. QoE shows Reparo's performance when its two modes are selected according to real network conditions.

Quality Improvement and Bandwidth Savings for Fixed Encoding Modes. Figure 5(a) and 5(b) displays the average SSIM value of all YouTube and Twitch video chunks generated by Reparo and the other baseline methods at seven bitrate levels on top of Pensieve [33]. To calculate the per-video SSIM, we extract the average per-frame SSIM of all video frames. Figure 5(c) and Figure 5(d) show the normalized average bandwidth consumption of each video chunk from the YouTube and Twitch streams. It should be noted that BETA-Live and VOXEL-Live baselines are regarded as fixed strategies here to show their capacities for saving bandwidth and their quality loss.

Reparo surpasses all baselines, with three key observations. *First*, compared to vanilla DASH streaming, Reparo's *Hbit* and *BWsave* modes improve performance. The *Hbit* mode gains 0.018 SSIM, equal to a 41.27%-56.11% bitrate improvement, while *BWsave* mode saves 30.86% bandwidth on average. *Second*, Reparo with our updated VFD model achieves a 22.96% quality improvement and 46.53% more bandwidth saving than using the generic VFD model. *Third*, Reparo's BWsave mode outperforms BETA-Live in quality and bandwidth overhead. Despite Reparo saving less bandwidth than VOXEL-Live, its visual quality is even lower than the previous bitrate level of Reparo, making Reparo with the updated VFD model superior to all baselines.

Overhead of Low Frame Rate Encoding. It should also be noted that the VFD-VFI mechanism introduces extra overhead because the VFD model cannot correctly identify every frame that can be discarded, and the lowered frame rate will hurt the efficiency of video encoding. In *HBit* mode, 4%-12% extra bandwidth is consumed. While in *BWSave* mode, the VFD-VFI mechanism brings an average SSIM loss of 0.007, but these overheads are not significant compared to the average SSIM gain of 0.018 and the average bandwidth saving ratio of 30.86%.

Overall QoE Improvements under Low-rate Network. Figure 6(a) and Figure 6(b) present the average QoE attained by the different methods under 4G uplink [40] and FCC [15] traces. We observe that Reparo achieves a 36.33% overall QoE gain on 4G uplink traces, and a 10.12% overall QoE gain on the FCC traces compared to vanilla DASH for the six different types of popular videos.

After merging two kinds of network traces together, we get an overall QoE gain of 18.13% on DASH, 13.26% on VOXEL-Live, 9.28% on BETA-Live, and 3.3% on Reparo-g. In addition, we multiply the rebuffer penalty in FCC traces with a factor of 15 to make the figure

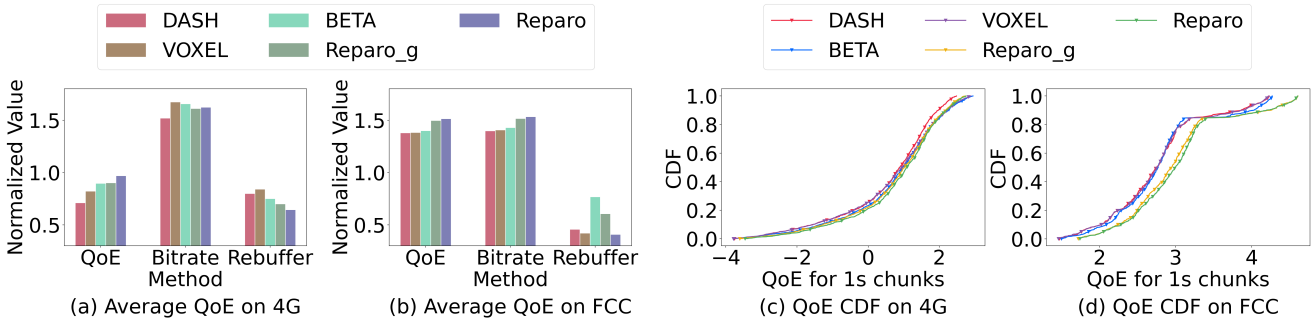
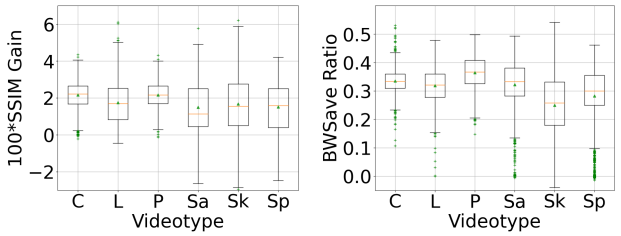


Figure 6: QoE Gain of Reparo

more clear, and all the other values are original. We see that Reparo significantly reduces the rebuffer penalty and gains better bitrate utility compared to DASH.

Figure 6(c) and Figure 6(d) present the CDF curves of QoE under the 4G and FCC traces for all video chunks of our six videos. We see that the majority of the Reparo curve is located below the other curves, showing that the QoE of Reparo is better than that of the others.



(a) SSIM Gain on 6 Types of Videos (b) BWSave Ratio on 6 Types of Videos

Figure 7: Reparo's Chunk-level Gain

Reparo's Chunk Level Effectiveness. To further explain the effectiveness of Reparo, Figure 7 presents Reparo's chunk-level gain compared to DASH across the six types of videos. Figure 7(a) illustrates the SSIM gain on every video chunk brought by Reparo's *Hbit* mode. On six types of videos, more than 75% of chunks could reach an SSIM gain of at least 0.005 from *Hbit* mode. Figure 7(b) shows the chunk-level bandwidth save ratio brought by Reparo's *BWSave* mode. On the six types of videos, more than 75% of chunks reach a bandwidth save ratio of at least 20% from *BWSave* mode.

Mobile Phone	Processor	VFD Pipeline
Galaxy A90 5G	SDM 855	593ms
OnePlus 9	SDM 888	520ms
iPhone 13	A15 bionic	213ms

Table 2: Time Cost of VFD Workflow on 1s Chunks

VFD-VFI Overhead. Since Reparo adds the VFD-VFI mechanism compared to vanilla DASH, we next demonstrate that the VFD-VFI pipeline can operate in real-time. For the VFD pipeline on the client side, we use three types of mobile devices to measure the execution speed of the VFD process, implemented by Pydroid for Android and Pyto for iOS. Table 2 shows the delay of executing the whole VFD pipeline on a 1s video chunk. We see it easily meets real-time requirements of under 600ms.

Resolution	fps	Delay on Single GPU	GPUs
240p	46	21ms	1
360p	20	50ms	1
480p	15	66ms	1
720p	16	112ms	3

Table 3: VFI-DNN Inference Latency

For VFI-DNN inference, Table 3 shows the frames per second achievable, alongside the computational time required. Reparo's VFI-DNN can reach a speed of 15 frames per second, at 480p resolution under single-GPU inference. This exceeds the minimum requirement of 12 fps. For 720p video, we use 3 GPUs to infer VFI-DNN in parallel, distributing the video frames equally among 3 GPUs for inference, thus speeding up the inference to 16 fps.

8 CONCLUSION AND FUTURE WORKS

In this paper, we have proposed Reparo, a new live video streaming system for low bandwidth networks. Reparo requires less uplink bandwidth between the upload client and the media server. It works on a novel video-frame-discard & video-frame-interpolation framework, adapting the frame-drop approach to the ingest part of the live video stream. Reparo also exploits the potential of frame-drops by selecting low frame-rate coding modes. Compared to DASH, our evaluation shows that Reparo achieves 18.13% improvement in users' QoE based on real-world network traces. Future work will focus on exploring the VFD model's generalizability, attempting to categorize the videos into different classes, and adopting a more generalized pre-trained VFD model for each class.

ACKNOWLEDGMENTS

This work is supported in part by the National Key R&D Program of China under grant No. 2022YFB3105000, the National Natural Science Foundation of China under grant No. 61972189, the Major Key Project of PCL under grant No. PCL2023AS5-1, and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

REFERENCES

[1] Vijay K Adhikari, Yang Guo, Fang Hao, Volker Hilt, Zhi-Li Zhang, Matteo Varvello, and Moritz Steiner. 2014. Measurement study of Netflix, Hulu, and a tale of three CDNs. *IEEE/ACM Transactions On Networking* 23, 6 (2014), 1984–1997.

[2] Gonca Bakar, Riza Arda Kirmizioglu, and A Murat Tekalp. 2018. Motion-based rate adaptation in WebRTC videoconferencing using scalable video coding. *IEEE Transactions on Multimedia* 21, 2 (2018), 429–441.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. 2011. A database and evaluation methodology for optical flow. *International journal of computer vision* 92 (2011), 1–31.

[4] James Bankoski, John Koleszar, Lou Quillio, Janne Salonen, Paul Wilkins, and Yaowu Xu. 2011. *VP8 data format and decoding guide.* Technical Report.

[5] Thomas Brox and Jitendra Malik. 2010. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence* 33, 3 (2010), 500–513.

[6] Ying Chen, Qing Li, Aoyang Zhang, Longhao Zou, Yong Jiang, Zhimin Xu, Junlin Li, and Zhenhui Yuan. 2021. Higher quality live streaming under lower uplink bandwidth: an approach of super-resolution based video coding. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 74–81.

[7] Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, et al. 2018. An overview of core coding tools in the AV1 video codec. In *2018 Picture Coding Symposium (PCS)*. IEEE, 41–45.

[8] Xianhang Cheng and Zhenzhong Chen. 2021. Multiple video frame interpolation via enhanced deformable separable convolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[9] Cisco. Janurary 2022. Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[10] Daniel Weinberger. Feburary 2019. Choosing the Right Video Bitrate for Streaming HLS and DASH. https://bitmovin.com/video-bitrate-streaming-hls-dash/.

[11] Xiangling Ding, Pu Huang, Dengyong Zhang, and Xianfeng Zhao. 2022. Video Frame Interpolation via Local Lightweight Bidirectional Encoding with Channel Attention Cascade. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1915–1919.

[12] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 2758–2766.

[13] Ericsson. Janurary 2019. Amazon Kinesis Video Streams Developer Guide. https://docs.aws.amazon.com/pdfs/kinesisvideostreams/latest/dg/kinesisvideo-dg.pdf#how-playback.

[14] Ericsson. June 2016. Uplink speed time-to-content - Mobility Report - Ericsson. https://www.ericsson.com/en/reports-and-papers/mobility-report/articles/uplink-speed-and-slow-time-to-content.

[15] Federal Communications Commission. December 26, 2018. Raw Data - Measuring Broadband America - Eighth Report. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-eighth.

[16] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 2018. Salsify:{Low-Latency} Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. 267–282.

[17] Evan Herbst, Steve Seitz, and Simon Baker. 2009. Occlusion reasoning for temporal interpolation using optical flow. *Department of Computer Science and Engineering, University of Washington, Tech. Rep. UW-CSE-09-08-01* (2009).

[18] Pan Hu, Rakesh Misra, and Sachin Katti. 2019. Dejavu: Enhancing videoconferencing with prior knowledge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 63–68.

[19] Siqi Huang and Jiang Xie. 2021. DAVE: Dynamic Adaptive Video Encoding for Real-time Video Streaming Applications. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 1–9.

[20] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 187–198.

[21] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2462–2470.

[22] Cyriac James, Mea Wang, and Emir Halepovic. 2019. BETA: bandwidth-efficient temporal adaptation for video streaming over reliable transports. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 98–109.

[23] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. 2018. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9000–9008.

[24] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 97–108.

[25] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-enhanced live streaming: Improving live video ingest via online learning. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 107–125.

[26] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*. 183–196.

[27] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. 2020. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5316–5325.

[28] Qing Li, Ying Chen, Aoyang Zhang, Yong Jiang, Longhao Zou, Zhimin Xu, and Gabriel-Miro Muntean. 2022. A Super-Resolution Flexible Video Coding Solution for Improving Live Streaming Quality. *IEEE Transactions on Multimedia* (2022).

[29] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. 2020. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 359–376.

[30] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733.

[31] Zhenxiao Luo, Zelong Wang, Jinyu Chen, Miao Hu, Yipeng Zhou, Tom ZJ Fu, and Di Wu. 2021. Crowdsr: enabling high-quality video ingest in crowdsourced livecast via super-resolution. In *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 90–97.

[32] Zhenxiao Luo, Zelong Wang, Miao Hu, Yipeng Zhou, and Di Wu. 2022. LiveSR: Enabling Universal HD Live Video Streaming with Crowdsourced Online Learning. *IEEE Transactions on Multimedia* (2022).

[33] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*. 197–210.

[34] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, Yaowu Xu, and Ronald Bultje. 2013. The latest open-source video codec VP9-an overview and preliminary results. In *2013 Picture Coding Symposium (PCS)*. IEEE, 390–393.

[35] Simon Niklaus, Long Mai, and Feng Liu. 2017. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 261–270.

[36] Mirko Palmer, Malte Appel, Kevin Spiteri, Balakrishnan Chandrasekaran, Anja Feldmann, and Ramesh K Sitaraman. 2021. VOXEL: cross-layer optimization for video streaming with imperfect transmission. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 359–374.

[37] Roger Pantos and William May. 2017. *HTTP live streaming*. Technical Report.

[38] H Parmar and M Thornburgh. 2012. Real-time messaging protocol (rtmp) specification. *Adobe specifications, December* (2012).

[39] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[40] Darijo Raca, Jason J Quinlan, Ahmed H Zahran, and Cormac J Sreenan. 2018. Beyond throughput: A 4G LTE dataset with channel and context metrics. In *Proceedings of the 9th ACM multimedia systems conference*. 460–465.

[41] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1164–1172.

[42] Stefan Roth and Michael J Black. 2007. On the spatial statistics of optical flow. *International Journal of Computer Vision* 74, 1 (2007), 33–50.

[43] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[44] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.

[45] Randall Stewart, Michael Ramalho, Qiaobing Xie, Michael Tuexen, and Phillip Conrad. 2004. *Stream control transmission protocol (SCTP) partial reliability extension.* Technical Report.

[46] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions*

on circuits and systems for video technology 22, 12 (2012), 1649–1668.

[47] Twitch Video Dataset (Chatting). 2022. surprise chatty chat stream before pilates. https://www.twitch.tv/videos/1595603627.

[48] Twitch Video Dataset (Legend). 2022. Porter Robinson Star Guardian Stream Tour with LilyPichu. https://www.twitch.tv/videos/1536751224.

[49] Twitch Video Dataset (Saddummy). 2021. Highlight: Seo Saebom] 07/23 3 new characters/new regions, Genshin Impact returned with 2.0 update! https://www.twitch.tv/videos/1100326518.

[50] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.

[51] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *International Journal of Computer Vision* 127, 8 (2019), 1106–1125.

[52] Mariem Ben Yahia, Yannick Le Louedec, Gwendal Simon, Loutfi Nuaymi, and Xavier Corbillon. 2019. HTTP/2-based frame discarding for low-latency adaptive video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 15, 1 (2019), 1–23.

[53] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 495–511.

[54] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 645–661.

[55] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: neural video enhancement at scale. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 795–811.

[56] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.

[57] Youtube Video Dataset (Pod Cast). December 31, 2018. Jason Lake | Founder CEO of compLexity | The Eavesdrop Podcast Ep. 14. https://www.youtube.com/watch?v=CpQZRLxJE5M.

[58] Youtube Video Dataset (Skit). June 2, 2016. KIDS DEMAND SKYLANDERS IMAGINATORS! Kidnapping Hostage Situation! (Gameplay Reveal Trailer Skit) 4k. https://www.youtube.com/watch?v=PPquOgXjGvo.

[59] Youtube Video Dataset (Sport). October 24, 2019. Punishers 17u vs Team Vegas, Bigfoot Hoops Grand Opening 4-28-2018. https://www.youtube.com/watch?v=ppb7hmQA6sU.

[60] Christopher Zach, Thomas Pock, and Horst Bischof. 2007. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition: 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007. Proceedings 29*. Springer, 214–223.

[61] Cong Zhang and Jiangchuan Liu. 2015. On crowdsourced interactive live streaming: a twitch. tv-based measurement study. In *Proceedings of the 25th ACM workshop on network and operating systems support for digital audio and video*. 55–60.

[62] Xiao Zhu, Subhabrata Sen, and Z Morley Mao. 2021. Livelyzer: analyzing the first-Mile ingest performance of live video streaming. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 36–50.

A AN EXAMPLE RESULT OF VFI-DNN

A.1 VFI-DNN Outperforms Naive Approaches

Figure 8 displays the performance of VFI-DNN [27], the traditional video frame interpolation method [60], and the approach of directly replacing the missing intermediate frame with the preceding frame. The traditional optical-flow-based video frame interpolation method struggles to recover the intermediate video frames even between two adjacent frames with minor scene changes, resulting in performance inferior to simply replacing the intermediate frame with the one preceding it. In contrast, VFI-DNN demonstrates the capability to recover the missing intermediate video frame with significantly higher quality.

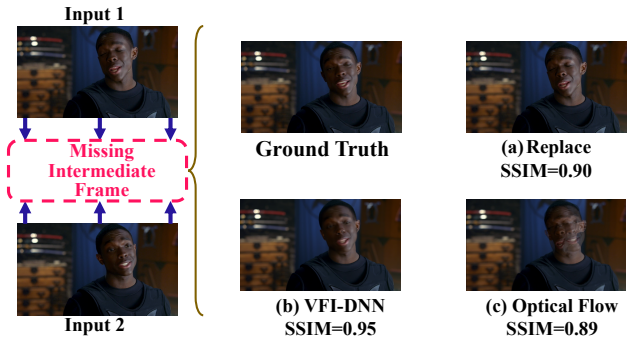


Figure 8: The Quality Comparison of VFI-DNN and the Other Approaches

A.2 VFI-DNN Cannot Handle Everything

The performance of VFI-DNN is constrained by the extent of scene changes. For instance, Figure 9 illustrates that VFI-DNN is less effective when confronted with drastic scene changes between two consecutive input frames. Consequently, it is essential to predict the performance of the video frame interpolation DNN on the upload client to identify frames that cannot be adequately recovered by our VFI-DNN after being dropped.

Figure 9: Good and Bad Results of VFI-DNN

B SINGLE FRAME VFI-DNN IS BETTER

Figure 10 presents a case study based on a 40-second clip from one of our evaluation videos, "Legend" [48]. In this study, for each preserved frame, multiple consecutive frames are discarded and subsequently recovered using the state-of-the-art multi-frame VFI-DNN [8]. By employing the original dropped frames as a reference for calculating the SSIM, it becomes apparent that as the number of dropped consecutive frames increases, the SSIM value decreases.

In comparison to the single-frame interpolation model, the multi-frame interpolation model demonstrates less robustness and is unsuitable for the frame-drop & frame-interpolation process proposed by Reparo. This is attributed to two primary factors. Firstly, employing the multi-frame model indicates that multiple consecutive frames are dropped, making it challenging for VFI-DNN to capture the nonlinear motion of objects between these frames. Secondly, the multi-frame interpolation model is typically utilized to enhance the frame rate of the original video, such as increasing the frame rate of game videos from 30 fps to 60 fps or even 120 fps. In the context of the frame-drop & frame-interpolation process, after the video frames are discarded, the gap between the remaining frames should inherently be larger. Consequently, the multi-frame model leads to a more significant degradation of video quality.

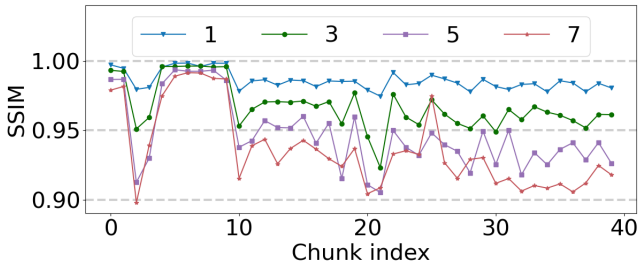
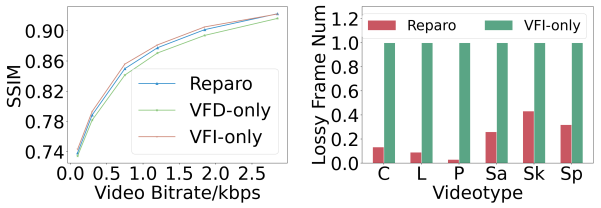


Figure 10: The Performance of Multi-frame VFI-DNN when Discarding Multiple Consecutive Frames

C VFD-VFI ABLATION STUDY

We finally evaluate Reparo's *Hbit* mode to verify if the VFD-VFI mechanism effectively handles quality degradation from inappropriate frame drops. Therefore, we calculate the number of interpolated frames with an SSIM loss over 0.05 as an additional metric after removing the VFD model and VFI-DNN.

Figure 11(a) shows the average SSIM value of VFD-only, VFI-only, and Reparo under Hbit mode. Reparo with only the VFD model has a 0.007 lower SSIM, demonstrating VFI-DNN's effectiveness, and contributing to 39% of the overall quality gain. Although Reparo with only VFI-DNN has a 0.003 higher SSIM, Figure 11(b) reveals that the VFI-only approach generates many frames with considerable SSIM degradation. In contrast, Reparo filters 56.87 ~ 96.91% lossy frames, enhancing the user experience. To conclude, both the VFD model and VFI deep neural network are effectively designed.



(a) SSIM Result of Ablation (b) Number of Lossy Frame Generated

Figure 11: VFD & VFI Ablation Study of Reparo