RESEARCH-ARTICLE

Logan: Loss-tolerant Live Video Analytics System

KICHANG YANG, Seoul National University, Seoul, South Korea

MINKYUNG JEONG, Seoul National University, Seoul, South Korea

JUHEON YI, Nokia Bell Labs, Murray, NJ, United States

JINGYU LEE, Seoul National University, Seoul, South Korea

KYOUNGSOO PARK, Seoul National University, Seoul, South Korea

YOUNGKI LEE, Seoul National University, Seoul, South Korea

Open Access Support provided by:

Seoul National University

Nokia Bell Labs

Published: 04 December 2024

Citation in BibTeX format

ACM MobiCom '24: 30th Annual International Conference on Mobile Computing and Networking
November 18 - 22, 2024
DC, Washington D.C., USA

Conference Sponsors:
SIGMOBILE

Logan: Loss-tolerant Live Video Analytics System

Kichang Yang
Seoul National University
Seoul, Republic of Korea
kichang96@snu.ac.kr

Minkyung Jeong
Seoul National University
Seoul, Republic of Korea
jmk7895@snu.ac.kr

Juheon Yi
Nokia Bell Labs
Cambridge, UK
juheon.yi@nokia-bell-labs.com

Jingyu Lee
Seoul National University
Seoul, Republic of Korea
jingyu.lee@hcs.snu.ac.kr

KyoungSoo Park
Seoul National University
Seoul, Republic of Korea
kyoungsoo@gmail.com

Youngki Lee
Seoul National University
Seoul, Republic of Korea
youngkilee@snu.ac.kr

Abstract

Cloud-based live video analytics with tight latency bound is gaining importance to support emerging applications such as UAVs and augmented reality. However, existing systems often struggle to meet stringent latency constraints under fluctuating network conditions with packet losses and late-arriving packets. We propose a loss-tolerant live video analytics system called Logan, which effectively accepts packet losses while maintaining high accuracy by utilizing the inherent resilience in DNNs. We design i) Codec-aware Inpainting, which accurately recovers the frame error from packet losses ii) Fast-Forward Recovery that prevents the remaining unrecovered error from propagating over future frames indefinitely. Our results show a 3× improvement (33.2%→99.9%) in SLO satisfaction rate compared to the reliable transmission scheme with <1% accuracy drop under a 5% packet loss rate.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Computer systems organization** → **Real-time system architecture**; • **Information systems** → **Multimedia streaming**.

Keywords

Live Video Analytics, Packet Loss, Mobile Systems

ACM Reference Format:
Kichang Yang, Minkyung Jeong, Juheon Yi, Jingyu Lee, KyoungSoo Park, and Youngki Lee. 2024. Logan: Loss-tolerant Live Video Analytics System. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3636534.3690695

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3690695

1 Introduction

Cloud-based live video analytics, vital for applications like Unmanned Aerial Vehicles (UAVs) and Augmented Reality (AR), require consistently meeting tight end-to-end latency requirements. For example, an autonomous UAV in delivery services must detect nearby obstacles within 150 ms to avoid collisions [1]. Current systems relying on reliable video transmission [18, 37, 43, 45, 76] struggle to satisfy such requirements due to frequent packet losses and late arrivals.

The end-to-end latency in video analytics systems is determined by network latency (the time it takes to send video data to the server) and computing latency (the analytics deep neural network (DNN) inference time). Prior works [25] have been successful in achieving stable computing latency since the compute resources are controllable (e.g., isolate GPU resources). However, stabilizing network latency presents a greater challenge due to its inherent uncontrollability, such as unpredictable channel fluctuation and congestion. Bitrate adaptation techniques such as traditional congestion control [5, 6, 26] and DNN-accuracy-aware frame degradation [18, 45, 76] mitigate packet losses and latency variations by reducing data transmission but fail to eliminate them due to their reactive nature.

This challenge has led us to develop a fundamentally different approach, *loss acceptance & recovery*. We enable analytics on the frames with missing packets, implementing error recovery strategies that diminish the effects of unpredictable network conditions. This strategy signifies a paradigm shift from dependence on the inherently variable nature of network resources to leveraging the more manageable computational resources, fundamentally transforming how packet loss and delays are addressed. When combined with systems that ensure computing latency [25], our method offers a pathway to achieving consistent end-to-end latency in cloud-based video analytics platforms.

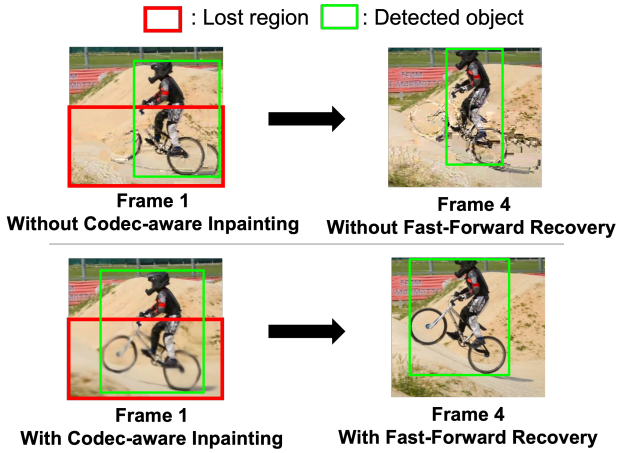


Figure 1: Effect of Codec-aware Inpainting (Frame 1) and Fast-Forward Recovery (Frame 4).

Our approach is supported by two key opportunities. First, recent video inpainting DNNs are capable of contextual recovery of the missing regions in a frame. By analyzing patterns across various video contents, they excel at filling in gaps by drawing on intact areas from both the current and preceding frames. Second, analytics DNNs exhibit resilience to errors, maintaining effective performance even when faced with incomplete input data. Their probabilistic operation prove durable against minimal accuracy declines in scenarios where inputs undergo quantization [35] or suffer from impairments such as noise, blur, and compression artifacts [15, 64, 80].

Using the opportunities, we propose Logan, a loss-tolerant live video analytics system. It consists of two synergistic components: (i) Codec-aware Inpainting that accurately recovers the error for the current frame's packet loss and (ii) Fast-Forward Recovery mechanism that limits the propagation of the remaining errors after Codec-aware Inpainting by utilizing the retransmitted late-arriving stale packets. The two components are co-designed to mutually enhance the performance: Fast-Forward Recovery improves the quality of the inputs for Codec-aware Inpainting, and Codec-aware Inpainting provides the optical flow information for future Fast-Forward Recovery. Our extensive evaluation shows that Logan achieves 99.9% latency SLO (Service Level Objective) satisfaction (3× compared to conventional systems with reliable transmission) with <1% accuracy drop under a loss rate of 5% (common in practical environments). Figure 1 shows a visual example of our techniques (details in Section 4).

First, Codec-aware Inpainting, a fast and accurate error recovery inspired by the recent advances in video inpainting DNN models, performs accurate recovery even under burst packet losses. It is designed to understand object boundaries and complex movement, with data generation that mimics packet loss patterns of encoded videos. We incorporate codec awareness, often overlooked by prior video inpainting models [21, 44, 72], to further improve accuracy and latency. Specifically, the model leverages the already present motion vectors adjacent to the lost area to accelerate the optical flow estimation. Also, it adopts a *difficulty-aware sharpness control* scheme where the model strategically prioritizes positional accuracy over sharpness for challenging scenes, cooperating with the future frames' residual values for error recovery.

Next, Fast-Forward Recovery mechanism challenges the conventional assumption in video streaming pipelines that once a frame is decoded, additional packets of the frame are discarded afterward. The key idea is to utilize the late-arriving retransmitted packets of the past frames to mitigate error propagation. It employs a fast-forward re-decoding method to correct the root error and recover its propagation by revisiting past frames. This method enhances the corrupted regions in the current frame in two folds: (i) directly incorporating information from late-arriving packets and (ii) indirectly improving the past frame quality for inpainting.

Our key contributions are summarized as follows:

- We shift the conventional packet loss management from relying on *uncontrollable network resources* to utilizing more *controllable computational resources* to achieve a stable end-to-end analytics latency under challenging network conditions. Our system enables many emerging video analytics applications with tight latency SLO.
- Logan is the first live video analytics system that leverages the error resilience of DNNs to process incomplete frames with high accuracy and real-time performance.
- We design an end-to-end system composed of Codec-aware Inpainting that accurately conceals the errors even under burst packet losses and Fast-Forward Recovery that effectively limits the error propagation.
- We extensively evaluate system performance under various packet loss patterns and network conditions with simulation and real-world traces. Specifically, Logan achieves 99.9% latency SLO satisfaction with <1% accuracy drop under a loss rate of 5%.

2 Motivation

2.1 Motivating Scenarios

We consider video analytics apps that stream the video to a cloud server for latency-sensitive analysis; we assume sole on-device processing is limited due to memory, processor, and thermal constraints. Delays caused by retransmissions are often not tolerable, as they compromise the timely processing essential for these applications. Following are example scenarios (see Table 1):

Unmanned Aerial Vehicle (UAV) delivery. Consider scenarios where autonomous UAVs are deployed for rescue purposes. For safe navigation and operation, these UAVs

	Unmanned Aerial Vehicle (UAV)	eXtended Reality (XR)	Underwater Imaging
Latency SLO	Collision avoidance 150 ms [1] Search and rescue 500 ms [75]	AR object tracking 200 ms [42] 3D reconstructed telepresence 150 ms [34]	Target hunting 200 ms
Communication	Cellular, Satellite	Cellular	Acoustic
RTT	LEO ~100 ms MEO ~250 ms GEO ~600 ms [13]	20~100 ms [48, 70]	Distance ÷ 1500m/s [2]

Table 1: Example video analytics applications.

must be capable of detecting obstacles within 150 ms while operating at the speed of 60 km/h as specified by industry standards [1]. The reliance on satellite networks, essential for operations across extensive areas, introduces an RTT of up to 50~100 ms even with Low Earth Orbit (LEO) satellites and longer with Geostationary Orbit (GEO) satellites. Accounting for the initial transmission time (an RTT) and the DNN analysis time (e.g., 33 ms for a DNN operating at 30 FPS), additional retransmission violates the latency constraint.

Augmented Reality (AR) criminal chasing. AR glasses worn by a police officer while chasing a criminal display a bounding box to indicate the criminal's location. The latency of such AR applications must be below 150~200 ms [34, 42] to ensure responsiveness and seamless user experience. However, RTT in urban cellular networks can peak at 100 ms and suffer from fluctuations due to handovers and queuing [48, 70]. Awaiting retransmissions compromise the latency requirements, severely degrading user experience.

2.2 Packet Loss and Late Arrival

Packet losses are common in the scenarios above, which rely on high-mobility wireless networks. Packet losses mainly stem from (i) wireless attenuation and interference and (ii) core network congestion. 2023 Microsoft Teams video conferencing application trace shows losses between 1~10% are common, occasionally exceeding 20% [57]. Extreme conditions, like air-to-ground, satellite, and underwater communications, exacerbate the problem [9, 73]. These losses lead to significant violations of latency Service Level Objectives (SLO), affecting not only the frames experiencing losses but also many subsequent frames since the application is blocked until retransmission completes. For example, 5% of packet losses can result in over 50% of frames failing to meet the latency SLO, as in Figure 2 (RTT=100 ms, SLO=200 ms).

Moreover, late-arriving packets due to latency fluctuation (often considered losses in real-time streaming systems [**?**]) have a similar effect with lost packets in terms of latency SLO violation. Recent cellular network measurements have documented considerable latency variability, resulting in up to 10% of video playback time spent on stall even with state-of-the-art bitrate adaptation techniques [48]. Our discussion of packet loss handling encompasses *both actual packet losses and late-arriving packets* that fail to meet latency deadlines.

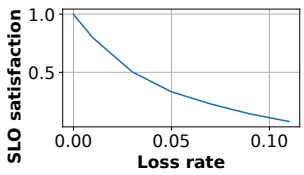


Figure 2: SLO satisfaction rate of reliable protocols under various packet loss rates.

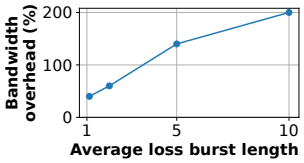


Figure 3: Required redundancy to achieve 99% SLO satisfaction under various loss burstiness levels. (5% loss rate, RS code [54])

3 Approach

Logan aims to provide stable end-to-end latency to video analytics applications, even amidst variable network conditions. The end-to-end latency in video analytics systems is composed of network latency (the duration to transmit video data to the server) and computing latency (the DNN inference time). Strategies like isolating server GPU resources [25] or preempting ongoing computations [27, 69] can secure stable computing latency. However, these methods are not viable for systems reliant on video transmission across unpredictable networks.

Adapting DNN computation is a practical solution to mitigate fluctuations in network latency. Systems can, for instance, opt for less resource-intensive models like those in the EfficientDet series for detection tasks [61] or reduce input resolutions to adhere to tight deadlines [24, 28]. However, this approach faces limitations with extreme network variability, particularly when network delays, intensified by retransmissions, surpass the latency SLO of the application.

Adjusting network resource utilization serves as another approach to managing network latency challenges. Bitrate adaptation techniques [6, 18, 76] reduce data transmission in response to packet loss occurrence, aiming to decrease loss rate and latency fluctuation. Although these reactive strategies lessen the frequency of losses, they either fail to completely prevent losses due to the quick shifts in channel conditions [29, 57] or lead to severe accuracy degradation. Forward Error Correction (FEC) combats this by adding redundancy bits to facilitate error correction, diminishing the necessity for packet retransmission. However, FEC necessitates significant bandwidth overhead to achieve desired latency SLOs, especially when the loss is bursty (a common

case [57]) as depicted in Figure 3. This underscores the inadequacy of relying solely on FEC in environments characterized by low or variable bandwidth, including satellite communications for drones/ships and cellular networks with high mobility (fluctuating significantly, frequently reaching <1 Mbps [30]).

3.1 Loss Acceptance & Recovery

Here, Logan takes a fundamentally different approach: *loss acceptance & recovery*– achieving stable network latency with additional *controllable* computational resources. Logan makes the network latency predictable by allowing analytics applications to operate on temporarily erroneous frames resulting from packet losses, as opposed to TCP-like reliable protocols which halt until retransmitted packets arrive and the system obtains error-free frames.[1] Logan instead recovers the errors using computational resources to mitigate the impact of packet losses on video analytics. Our approach can be integrated with existing systems that offer latency guarantees on the computing side [25] to significantly improve end-to-end reliability in computation offloading systems.

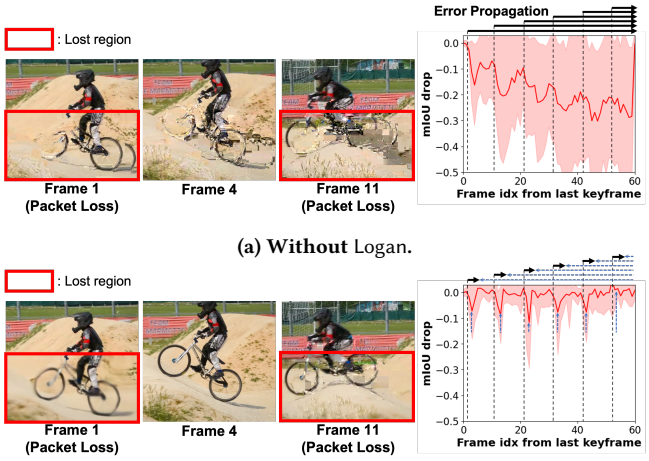
The acceptance of packet loss has not been considered a viable option due to its substantial impact on frame quality. Error concealment algorithms [10, 41, 56], primarily formulated during the initial stages of video codec development, are designed to fill in the missing pixels using the information available in the current and previous frames. However, these algorithms often fail to accurately reconstruct the diverse content within videos due to their handcrafted nature, as demonstrated in Figure 4a and further analyzed in Section 7.

We identify two key characteristics of DNNs, (i) contextual recovery of generative DNNs and (ii) error resilience of analytics DNNs, as enablers of our approach.

Contextual recovery. Contrary to traditional manually-designed error concealment strategies, generative DNNs can accurately comprehend object semantics and complex motion patterns due to their capacity to learn from the vast diversity of video content. This enables them to interpolate and extrapolate missing regions by analyzing the unimpaired regions of the current and previous frames. Recent advancements in video inpainting DNN models [21, 44, 72] have demonstrated significant success in reconstructing missing pixels, albeit with higher latency (hundreds of milliseconds) and a focus on addressing sparse losses. Frame 1 in Figure 4b shows an example of a frame that has undergone our inpainting model after packet loss.

Error resilience. DNNs inherently exhibit a degree of resilience for information loss in their inputs due to their probabilistic nature. For example, various studies on DNN quantization have demonstrated that DNNs can withstand precision loss in inputs [35] and various types of information loss, such as noise, blur, and compression distortions to some extent [15, 64, 80]. Incorporating a variety of distortions through data augmentation has become a de facto standard of DNN training, further improving resilience. This resilience suggests that minor frame errors resulting from packet loss may be tolerable.

[1]We define "loss" as packet loss and "error" as frame distortions resulting from packet loss.



(a) Without Logan.

(b) Logan.

Figure 4: Error recovery of Logan**. The graph reports the accuracy drop averaged over all videos in DAVIS dataset (±1 standard deviation) where packets of frames 1, 11, 21, 31, 41, 51 in each GoP are lost.**

4 Logan Overview

Logan is our realization of *loss acceptance & recovery*. Logan aims to limit the error to a level that can be tolerated by the error resilience capabilities of analytics DNNs, ensuring that the application can continue its operations.

Logan comprises two major components. (1) Codec-aware Inpainting that quickly and accurately recovers the packet loss in the current frame to analyze, and (2) Fast-Forward Recovery mechanism that shortens the duration of the propagation of the remaining error. Figure 4b shows the visual example of the effect of Logan. Codec-aware Inpainting accurately recovers the frame errors (Frame 1), and Fast-Forward Recovery shortens the duration of the error propagation to later frames, preventing the accumulation of errors, as shown in the graph on the right.

4.1 System Architecture

Figure 5 illustrates the architecture. Logan is compatible with any existing video analytics applications without retraining existing analytics models. Initially, the user specifies the latency SLO of the application and registers the analytics

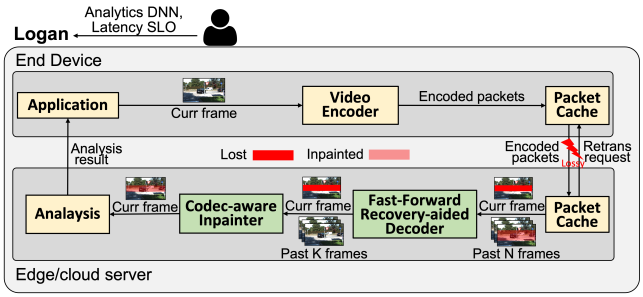


Figure 5: System architecture of Logan.

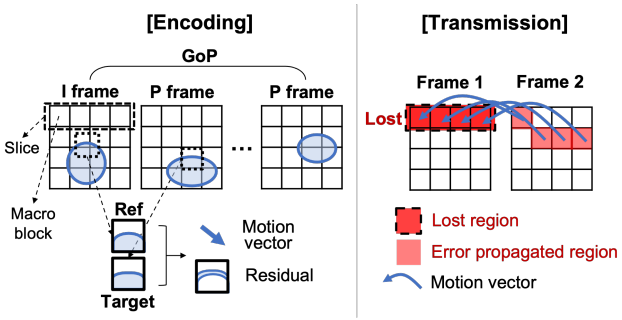


Figure 6: Video encoding and transmission process.

DNN model to Logan. Subsequently, Logan processes the video from the application with the registered DNN and returns the analysis results.

The internal of Logan includes: i) An end device (e.g., autonomous drone) that captures video from its camera and streams it to the server, ii) An edge/cloud server that analyzes the video and sends the analysis result (e.g., bounding boxes) back to the device. The end device encodes the video with a video codec and caches the packets for potential retransmission. Upon packet arrival, the server requests retransmission of any lost packets and proceeds with the subsequent processing steps. As the latency SLO for a frame approaches, Fast-Forward Recovery module collects retransmitted packets and runs sequential re-decoding from the past frames to recover propagated errors. Then, Codec-aware Inpainting module leverages the recovered frames to fill in missing regions of the current frame. Finally, the inference engine executes the analytics DNN on the recovered frames, and the analysis results are transmitted back to the end device. [2]

4.2 Background: Video Streaming

We briefly cover video streaming backgrounds to help understand our techniques in Section 5 and 6.

Video Codec. Figure 6 depicts the H.264 video encoding process. An encoded video consists of successive frame groups called the Group of Pictures (GoP). Each GoP includes an independently decodable I frame (also called a keyframe) and multiple P frames responsible for encoding the variance from reference frames.[3] Further, each encoded frame is divided into small pixel blocks, typically 16x16, known as macroblocks. Macroblocks are classified into two categories: (i) inter-predicted macroblocks, which comprise motion vectors and residuals, and (ii) intra-predicted macroblocks with intra-prediction modes and residuals. A motion vector identifies the region in the *past frame* resembling the macroblock, while an intra-prediction mode identifies the region in the *current frame*. The residual encodes additional information, representing the disparity between the macroblock and the indicated region.

Transmission. The unit of data transmission is a slice, which comprises a sequence of macroblocks. Typically, each slice is encapsulated in a network packet and transmitted to the receiver. Each slice represents a non-overlapping region and is independently decodable. In case a slice is missing due to packet loss, the pixel values in the corresponding region become unknown. Furthermore, regions in subsequent frames that depend on it (dependency indicated by motion vectors) are also affected.

5 Codec-aware Inpainting

5.1 Overview

Recent progress in video inpainting DNN models has achieved notable success in generating missing pixels [21, 44, 72]. Nonetheless, their integration into live video analytics faces two main obstacles. Firstly, existing research prioritizes the enhancement of scene generation quality without sufficiently addressing latency issues. Generating a single scene can take hundreds of milliseconds [44, 46], making these models unsuitable for apps with strict latency SLOs. Secondly, these models are designed to mend sparse holes in uncompressed images, leading to less effective performance when addressing burst losses in compressed video formats [44].

To overcome the identified challenges, we craft an accurate and lightweight video inpainting model. Our innovation lies in redesigning inpainting models to be video codec-aware. Specifically, our model leverages (i) motion vectors and (ii) residuals from encoded videos to enhance the efficiency of SOTA inpainting models while achieving high accuracy.

First, motion vectors near the loss areas provide critical clues for error correction. While traditional error concealment algorithms like Boundary Matching Algorithm [41, 56, 63] partially incorporate this information, recent video inpainting models overlook such codec-specific details. This

[2]Note that the loss of analysis results is not a focus of our consideration, as addressing this is relatively straightforward. Given the small size of these results compared to video data, employing a strategy such as adding high redundancy and regularly sending the result multiple times should suffice to address the losses.

[3]B frames, seldom used in real-time video, are omitted.

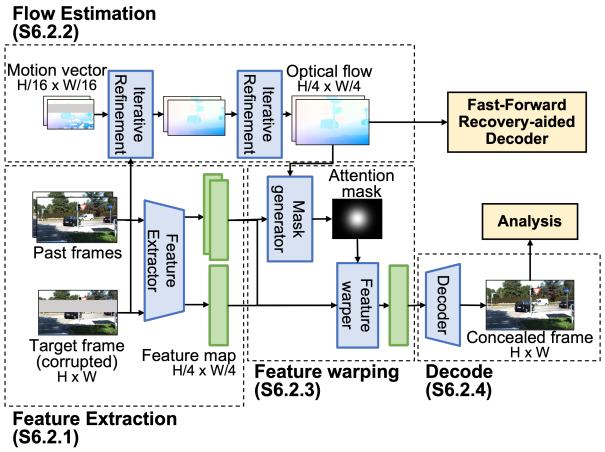


Figure 7: Codec-aware Inpainting model overview.

oversight leads to inefficiencies, exemplified by the high latency in the optical flow estimation process inside the inpainting models; for instance, RAFT [62], a state-of-the-art optical flow estimation DNN requiring approximately 100 ms for a 1024×436 frame. Second, residuals, or the information unexplained by the motion vectors, are also critical for accurate reconstruction, yet they are often underused by existing inpainting models. Our model distinguishes itself by effectively leveraging residuals, focusing on generating only the missing information that the residuals do not cover, thereby streamlining the recovery process.

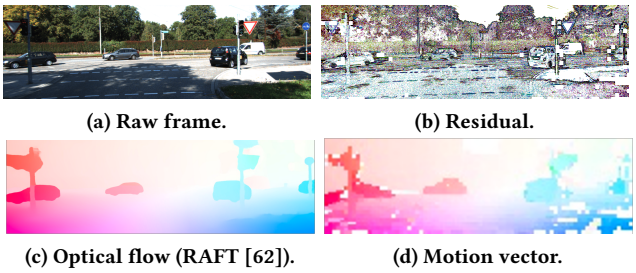
5.2 Model Design with Codec Awareness

With the above insights, we introduce an inpainting model outlined in Figure 7. Our model encompasses four stages: (i) feature extraction for encoding semantic information, (ii) flow estimation for estimating optical flow[4], accelerated by motion vectors as an initial guess, (iii) feature warping to fill in the features of missing regions, and (iv) decoding to generate a recovered frame from the warped features. We detail the four stages below.

5.2.1 Feature extraction. Our model takes a corrupted frame and two of its previous frames (recovered by Fast-Forward Recovery in Section 6) as inputs. Each frame undergoes feature extraction DNN comprised of nine convolution layers with a filter size of 3×3 and ReLU activation. For a frame of size H×W, the model extracts the feature of size H/4×W/4 and proceeds with the feature warping at this resolution.

5.2.2 Flow estimation with motion vectors. Flow estimation is the core component guiding the alignment of valid regions in the past frames with the corrupted regions in the target frame. However, accurate flow estimation DNN models are computationally intensive (e.g., RAFT [62] takes ≃100 ms for a 1024×436 frame), unsuited for real-time frame recovery.

[4]Optical flow differs from motion vectors in that it aims to represent the true object motions instead of simple block similarity.



(a) Raw frame. (b) Residual.

(c) Optical flow (RAFT [62]). (d) Motion vector.

Figure 8: Optical flow, motion vector, and residuals of an example frame.

Our model enhances the flow estimation process by utilizing motion vectors, which have shown a strong correlation with optical flows, as depicted in Figure 8. Motion vectors can be efficiently extracted from encoded videos, offering a low-computational-cost starting point for optical flow estimation. With these motion vectors as preliminary estimates, we refine them to obtain accurate optical flows efficiently.

We initially preprocess motion vectors to facilitate the conversion into optical flow. This involves filling the missing vectors from lost packets using co-located vectors in previous frames and applying average filtering to minimize noise. With these preprocessed motion vectors (at a resolution H/16×W/16, where H and W are the frame height and width) as an initial estimate, we employ a coarse-to-fine iterative refinement strategy, a common approach in optical flow estimation DNNs [32, 53, 60]. In each refinement iteration, a low-resolution coarse input is refined through a series of convolution layers to produce a finer estimate at double the resolution. We perform two iterations of refinement, with each round employing a lightweight SPyNet [53] architecture. The final optical flow estimate at a resolution of H/4×W/4 is then used for feature warping.

5.2.3 Feature warping with residuals. The feature warping stage utilizes the estimated flow to warp the frame features from past frames into the corrupted regions of the current frame. Notably, our model introduces a *Difficulty-aware Sharpness Control* mechanism to minimize error propagation along with leveraging residual values of future frames.

Difficulty-aware sharpness control. Mobile devices operating in diverse environments frequently encounter inputs with varying levels of complexity. In scenarios where generating a precise output becomes challenging, we strategically focus on minimizing the impact of error propagation. This is achieved by incorporating information from the residuals of future frames. Particularly, the residuals predominantly contain 1) object edge information and 2) details of newly appearing objects, as shown in Figure 8b. This is because it is

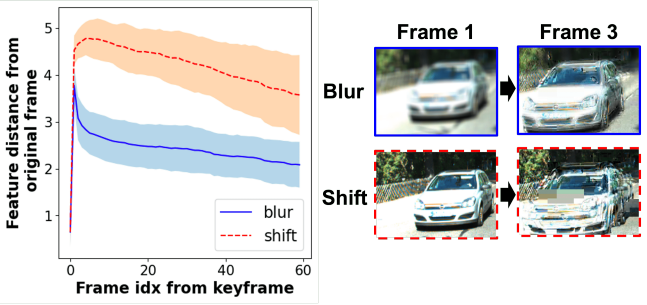


Figure 9: Feature distance (LPIPS [78]) between frames in the lossy video and the original frame for two recovery strategies. Packet loss injected in frame 1.

challenging to find an exact matching macroblock in the previous frame for areas rich in edges or featuring new objects, leading to uninformative motion vectors and rich residuals. Thus, the difficulty-aware sharpness control mechanism is tailored to favor *sharpness error*—which can be recovered using future residuals—over *position error*.

Common video inpainting models generate sharp results for human perception by incorporating adversarial loss [7, 44, 77]. However, attempting to generate sharp frames under insufficient inputs (limited number of frames, high loss rate and burstiness) results in severe position distortion (shift) and drops in accuracy for both current and future frames. Instead, we find that less sharp, but less distorted recovery is more helpful. Figure 9 shows an example. We assume two error recovery strategies: 1) sharp but high position error (original frame shifted right by 10 pixels), and 2) blurred, but less position error (original frame with a blur kernel of size 9 applied). Feature distance is measured using widely used LPIPS [78]. The distance quickly decreases for the latter case, as the residual facilitates the recovery of edge details omitted during initial recovery. Conversely, for the first case, the distance decreases slowly (even increases occasionally) because position error severely propagates to subsequent frames as unpredictable artifacts (Frame 3).

Implementation of sharpness control. To regulate output sharpness, the feature warping process (Figure 7) adjusts the degree of feature aggregation from previous frames based on estimated difficulty. The warping process involves computing a weighted sum of features from various locations in previous frames, which provide contextual information for inpainting the corrupted region. Aggregating features from multiple locations tends to result in blurry predictions, while focusing on features from a single location leads to sharp predictions. We make the aggregation weights (i.e., difficulty-aware masks) learnable by deriving them from the estimated flow and features using convolution layers. This enables the model to learn the difficulty level based on the given input. For difficult inputs, the model assigns a high weight to a single location, whereas for less complex inputs, the weights are distributed across locations.

5.2.4 Decode stage. The decoder module generates the final recovered frame by utilizing the warped features obtained from the previous stage. However, a significant challenge arises when certain regions within the frame cannot be adequately explained solely by reference to past frames, particularly with newly emerging objects. Precise recovery of these regions poses a fundamental challenge, as accurate prediction solely based on historical frames is inherently limited. Traditional video inpainting methods often resort to generating synthetic, plausible content using computationally intensive image inpainting techniques [21, 72].

In contrast, Codec-aware Inpainting avoids restoring potentially false plausible content and instead employs a lightweight decoder composed of four 3×3 convolution layers. The responsibility for recovering newly emerging objects is deferred to future residuals. These residuals inherently encapsulate essential information about newly appearing objects, enabling their accurate restoration without adding unnecessary computational overhead.

5.2.5 Training data generation. The inpainting model can be trained on any large video dataset through a self-supervised approach. Training data is generated directly from raw videos without any labels by masking the frames. We use the Youtube-VOS dataset [71] for training. To demonstrate our model's general applicability, we intentionally refrain from fine-tuning it on the datasets used in our evaluations.

To ensure alignment with our specific context, we have developed a customized approach for generating training data that faithfully simulates network packet loss. Unlike conventional video inpainting methods that generate sparse random-shaped masks [21, 44, 77], we generate masks that mimic encoded frame corruption patterns caused by packet losses. Specifically, we divide the frame into macroblocks of 16×16 pixels and group the consecutive macroblocks into slices. The size of each slice is randomly determined, ranging from 10 to 600 macroblocks, which is a wide range to simulate burst losses. Then, we simulate packet loss on these slices with varying rates, ranging from 0 to 30%.

6 Fast-Forward Recovery

6.1 Overview

The minor errors left after inpainting are insignificant individually, but they propagate to subsequent frames. With successive occurrences of packet losses, these errors accumulate, leading to analysis failure. Figure 10 illustrates this error accumulation when the bottom half of a frame is lost every ten frames; we evaluated the impact on segmentation

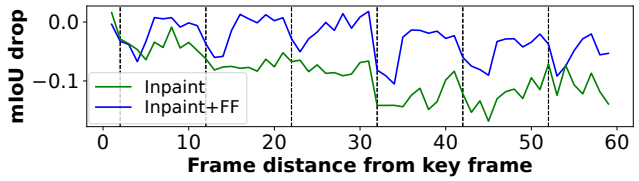


Figure 10: segmentation accuracy (mIoU) drop when packets are lost every 10 frames (dotted vertical lines indicate packet loss occurrence). Averaged over videos in the DAVIS dataset. FF: Fast-Forward Recovery.

accuracy by measuring the mean Intersection over Union (mIoU) on the videos of DAVIS dataset [52].

We propose Fast-Forward Recovery to restrict the duration of error propagation, thereby limiting the total error to a manageable amount, as shown in the blue line of the figure. The key idea is to utilize the late-arriving retransmitted packets of past frames. As the errors from these missing packets have been propagating up to the current frame, the arrival of these packets can help improve the quality of the current frame. In particular, the quality of the current frame improves in two key ways: 1) by enhancing regions directly dependent on previously lost regions in past frames, and 2) by improving the quality of recent frames used as input for inpainting of the current frame, thereby enhancing the inpainting quality.

Our mechanism differs from conventional video streaming in that it breaks the assumption that any additional late-arriving packets of the frame are discarded once decoded. Existing systems either await complete frame retransmission (high latency) or proceed with erroneous frames (low accuracy due to error propagation).

6.2 Fast-Forward Mechanism

Now, we detail how to effectively incorporate this new information from past frames to enhance the current frame.

Retransmission protocol. We use a retransmission logic similar to WebRTC, a widely used video streaming protocol. When the receiver detects a discontinuity in packet sequence numbers, it sends a NACK request to the sender. The sender initiates the retransmission upon receiving this request if the packet has not been recently retransmitted. [5]

Fast-forward onset strategy. Upon the arrival of a new frame for inference, we intentionally delay the initiation of

[5]If retransmissions are likely to disrupt initial transmissions, we can allocate bandwidth separately for initial transmissions and retransmissions, similar to strategies employed by video streaming protocols such as SRT [59]. We did not adopt this approach because the typical loss rate is less than 10%, making the bandwidth overhead of retransmission relatively insignificant.

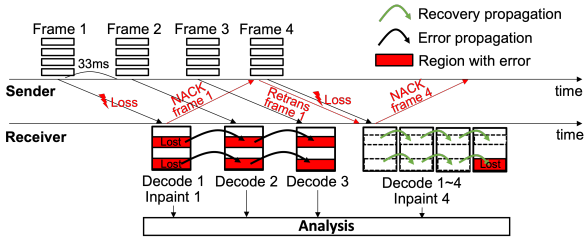


Figure 11: Fast-Forward Recovery mechanism example.

Fast-Forward Recovery to maximize the inclusion of late-arriving packets. By monitoring the average latency encompassing decoding, inpainting, and inference, we calculate the available slack time within the latency SLO, guiding the optimal timing for Fast-Forward Recovery. When initiating the process is expected to exceed the latency SLO, Logan opts for the standard decoding process without fast-forwarding.

Fast-forward procedure. Once initiated, our mechanism takes all currently available packets (including late-arrived packets not used for previous decoding) as inputs and generates re-decoded (past and current) frames as output. Such re-decoding can be performed in real-time given the minimal decoding latency with modern hardware (1 ms for HD frame on AMD EPYC 7313 CPU).

First, we prepare inputs for re-decoding. In particular, it identifies all retransmitted packets that arrived after the previous fast-forward operation. If such packets exist, it identifies the set of *effective frames* affected by the retransmitted packets, i.e., all frames between the oldest frame associated with the retransmitted packets and the current frame. Then, it loads the cached packets that arrived earlier for the effective frames. The combination of newly arrived packets and the cached packets is given as input to the decoder.

Next, the decoder sequentially re-decodes all the effective frames, starting from the oldest frame, denoted as frame j. Due to the stateful nature of the decoder, we need to revert its state back to the moment right before frame j was initially decoded. The necessary state for decoding frame j includes all frames upon which frame j could depend, extending back to the most recent keyframe. Therefore, we maintain a cache of all frames within the current GoP and use them to revert the decoder's state. After state restoration, the decoder proceeds with the original decoding process from frame j up to the current frame. This process rectifies the frame errors with the lost packet and propagates this recovery to subsequent frames.

Figure 11 illustrates our Fast-Forward Recovery mechanism with a 30 FPS streaming scenario under 100 ms RTT. Two packets in frame 1 are lost, responsible for the upper and lower regions (denoted as red "Lost" regions). Logan requests retransmissions for these lost packets, proactively employs error concealment on frame 1, and proceeds with the DNN

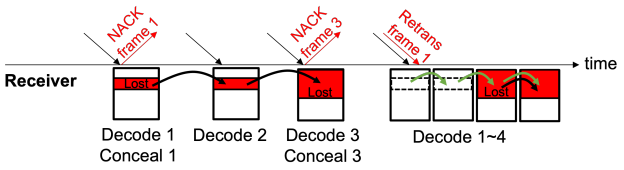


Figure 12: Error concealment needs to provide motion vectors for recovery propagation.

inference. The remaining unconcealed errors after error concealment propagate to the subsequent frames (frames 2 and 3). When the retransmitted packets for frame 1 arrive before the decoding time of frame 4, Logan re-decodes frame 1 using the newly arrived packets and re-decodes frames 2, 3, and 4 to propagate the recovery. As a result, the upper region of frame 4 becomes completely error-free. Notably, even the lower region of frame 4, which is lost, benefits during error concealment, as the inputs (frames 3, 2, etc.) now become error-free due to recovery. This approach ensures that information loss does not accumulate indefinitely (only across 3 frames in this example), contributing to the overall reliability of the video contents under analysis.

Cache reset. Upon the arrival of a new keyframe, the cached frames and packets from the previous GoP are discarded, as the frames in the new GoP are independent of those in the preceding GoP. The Fast-Forward Recovery is not applied to the frames of the previous GoP even if new packets arrive. Furthermore, packets arriving too late to enhance the current frame quality are discarded, with the cut-off empirically set at 30 frames prior to the current frame. The memory overhead incurred by caching the frames and packets is minimal; typically <50MB for a 30 FPS 1280×720 video.

6.3 Co-Design with Inpainting Model

While fast-forwarding, we may encounter a missing region whose retransmitted packets have not arrived yet. In such cases, we take the optical flows from our inpainting model and compute the residuals based on these motion vectors. We then utilize such obtained motion vectors and residuals for the re-decoding of the region. Figure 12 exemplifies the recovery propagation process where packets of frames 1 and 3 are lost. Fast-Forward Recovery mechanism is activated at frame 4 decoding, triggered by the arrival of the retransmitted packet from frame 1. During the re-decoding of frame 3, the algorithm leverages the recovered frame 2, along with the motion vectors in the error-concealed regions of frame 3, to propagate the recovery through frame 3.

7 Evaluation

7.1 Setup

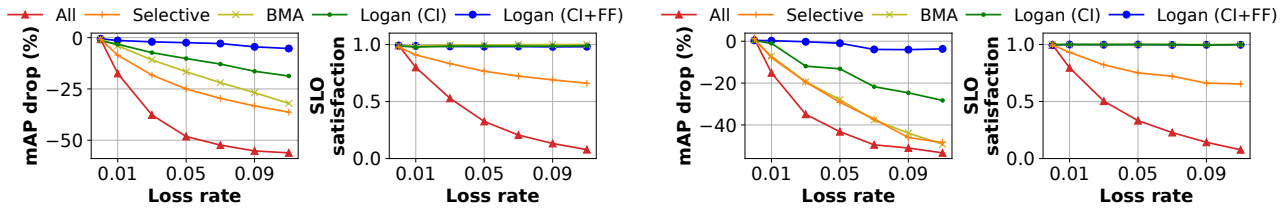
Implementation. We implement Logan in C++. We use LibTorch 1.13.1 and Torchvision 0.14.1 for inpainting and DNN inference. We use FFmpeg for video encoding/decoding and OpenCV for image processing. We use H.264 codec with *zerolatency* option to enable real-time streaming. We use the Ringmaster [57] protocol for video streaming but modify it to use NACK instead of ACK to optimize bandwidth usage under ACK losses. We do not use bitrate adaptation by default to isolate the effect of retransmission, except in Section 7.3 where we adapt bitrate according to cellular bandwidth trace. The cloud server is implemented on an A+ Server 4124GS-TNR with AMD EPYC 7313 and NVIDIA RTX 3090 GPUs. We use the loopback network interface with the Linux tc tool to emulate packet loss, propagation delay, and bandwidth.

Tasks, Models, Datasets. We use two tasks: object detection and semantic segmentation. We selected these two tasks because they present considerable challenges in terms of loss tolerance, given their requirement for fine-grained output. Our system can readily adapt to higher-level tasks like video captioning, question answering, etc, since these tasks generally allow greater tolerance of information loss in each frame. For detection, we use YOLO-v5 [38] with different capacities (n/s/m/l, default: m). For segmentation, we use FPN [40] with ResNet101 backbone.

We use four datasets: (1) UAVDT dataset [17] comprised of 50 1080×544 videos UAV videos featuring UAV footage from various urban locations (detection) (2) KITTI dataset [23] comprised of 21 1248×384 videos of street scenes with vehicles and pedestrians (detection) (3) DAVIS dataset [52] consisting of 90 854×480 videos of various human and animal activities (segmentation) (4) Drone Racing dataset [3] consisting of 18 640×480 videos of first-person view indoor drone racing (detection). All videos are encoded at 30 FPS.

Packet Loss Model. Due to the lack of publicly available packet loss traces for modern wireless networks, we evaluate our system with i) a loss simulation model, ii) emulation with real-world bandwidth traces, and iii) real-world packet loss traces we collected. For the loss simulation model, we use a widely-used Gilbert-Elliot (GE) model [19]. GE is a Markov model with "good" and "bad" states, each with different transition and loss probabilities. We set the loss probability in good and bad states as 0 and 1. We randomly sample the bad- to good-state transitioning probability (the inverse of burstiness) within a range from 0.9 to 0.1. We set the good- to bad-state transition probability to match the desired loss rate, which we vary from 0% to 19% (default: 5%). The value ranges are set to match measurement results from prior video streaming works [57]. Details on the real-world bandwidth and loss traces are provided in Section 7.3.

Network Bandwidth and Video Bitrate. We configured the network bandwidth to $video\_bitrate \times (1+loss\_rate)$ that accounts for video and its retransmission proportional to the loss rate. This emulates a scenario where bitrate adaptation is performed to match the available bandwidth. When using



(a) UAVDT dataset.

(b) KITTI dataset.

Figure 13: Performance under various loss rates.

FEC, we proportionally increased the bandwidth to account for the parity bit ratio. The default encoding bitrate is set at 4 Mbps for the UAVDT and KITTI datasets and 2 Mbps for the DAVIS and Drone Racing datasets.

RTT. Network RTT is set to 100 ms by default, following previous works on video streaming [57**?**] but we also report performance under various RTTs.

Metrics. We use two main metrics: i) average accuracy drop compared to lossless inference and ii) ratio of frames that satisfy latency SLO. We use mAP (mean Average Precision) for detection accuracy and mIoU (mean Intersection over Union) for segmentation accuracy. For frames that violate the latency SLO, we calculate the accuracy using the most recently available inference results. The time taken to transmit analytics results back is omitted from both the system latency and latency SLO calculations, as it remains constant across all comparison scenarios. The default latency SLO is set to 150 ms.

Comparison schemes. We use the following baselines.

- **All retransmission (All)** transmits all the frames reliably. Conventional video analytics systems [8, 18, 43, 45, 76] that use TCP and other protocols (e.g., DASH, HLS) built on top of TCP fall in this category.
- **Selective retransmission (Selective)** selectively retransmits only the important frames and uses BMA to conceal remaining error, similar to prior works [36, 51]. The importance of a frame is determined based on its dependency level; frames that serve as a reference for many subsequent frames are considered crucial. The importance of keyframes with loss is always set as 100% following [51]. We selectively retransmit the top 30% of the lost frames.
- **No retransmission with BMA (BMA)** does not request any retransmission and uses BMA for error concealment. We use the FFmpeg implementation of BMA.
- **No retransmission with Codec-aware Inpainting (**Logan **(CI))** does not request any retransmission and uses our Codec-aware Inpainting model to conceal the error.
- Logan **(CI+FF)** uses the our Codec-aware Inpainting model and the Fast-Forward Recovery mechanism.

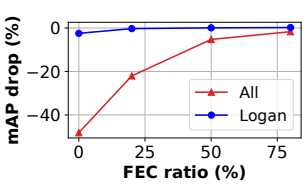


Figure 14: Performance with various FEC ratios.

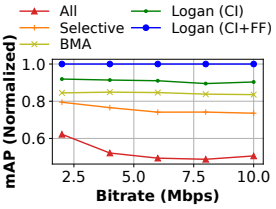


Figure 15: Performance under various bitrates.

7.2 End-to-end Performance

7.2.1 Various Loss Rates. Figure 13 shows the performance under various packet loss rates for UAVDT and KITTI datasets. Logan (blue lines) consistently outperforms the baselines. **All** fails to meet the latency SLO due to excessive retransmission. **Selective** improves this, but retransmission for important frames still violates the latency SLO. An accuracy drop also exists. **BMA** shows the lowest latency but has low inference accuracy due to inaccurate error recovery. **Inpaint** improves the average inference accuracy compared to **BMA** (20% at 5% loss rate). Logan further improves performance by limiting error propagation with Fast-Forward Recovery. It maintains the latency within the SLO (99.97% at 5% loss rate, with occasional misses attributed to DNN latency spikes from hardware issues) while keeping the accuracy drop minimal (0.9% at 5% loss rate) compared to **All**.

7.2.2 Integration with FEC. We show the system performance when FEC is integrated in Figure 14. We use Reed-Solomon code [54], widely used in commercial video streaming products [57]. In case of error correction failure, it requests retransmission. We use 0%~80% parity bit ratios. The result shows that even Logan with 0% redundancy achieves similar performance to the baseline with 80% redundancy, demonstrating improved tradeoff and the potential of significant bandwidth saving.

7.2.3 Various Bitrates. Figure 15 shows the system performance under various video bitrates. We vary the bitrate from 2 to 10 Mbps, by 2 Mbps steps. **All** exhibits poor performance at high bitrates since the number of packets per frame

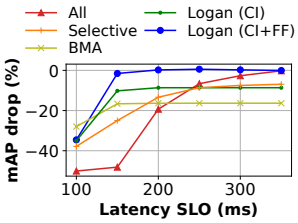


Figure 16: Performance under various latency SLOs.

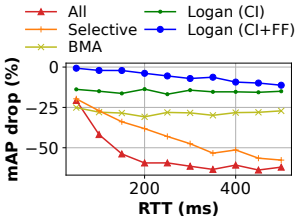


Figure 17: Performance under various RTTs.

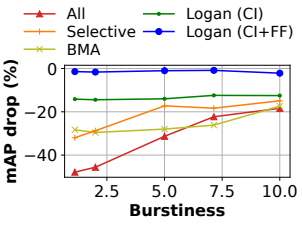


Figure 18: Performance under various burstiness.

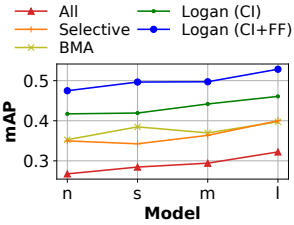


Figure 19: Performance under various model sizes.

increases. This elevates the likelihood of each frame experiencing at least one packet loss, resulting in more number of frames undergoing retransmission and violating the latency SLO. Logan maintains superior performance relative to the baselines, suggesting its effective integration with bitrate adaptation mechanisms.

7.2.4 Various Latency SLOs. Figure 16 depicts the performance under various latency SLOs for UAVDT dataset, ranging from 100 ms to 350 ms, incremented by 50 ms. Logan outperforms baselines except when the SLO is set to 100 ms, which leaves insufficient time for the Codec-aware Inpainting (taking ≃15 ms). When latency SLO is sufficiently high, Logan and **All** equally achieve the best performance since there is enough time to wait for all retransmitted packets.

7.2.5 Various RTTs. Figure 17 shows the performance under various RTT conditions for KITTI dataset. We vary the RTT from 50 ms to 500 ms by 50 ms and set the latency SLO as one-way delay plus 100 ms. Logan consistently outperforms baselines. The effect of the Fast-Forward Recovery slowly decreases as RTT increases due to the increasing staleness of the retransmitted packets leading to less error recovery.

7.2.6 Various Burstiness. Figure 18 shows the performance under various loss burstiness levels for KITTI dataset, where the burstiness value indicates how many consecutive packets are lost on average (loss rate fixed to 5%). **All** and **Selective** perform better in bursty loss scenarios since the loss occurs less frequently, triggering retransmission (i.e., violating SLO) for only a few frames. Similarly, the **BMA** and **Inpainting**

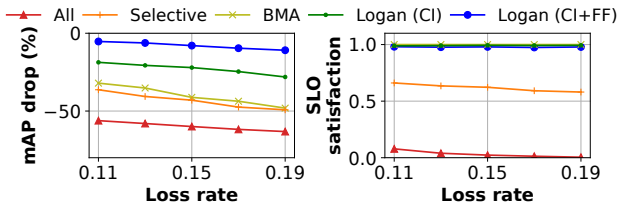
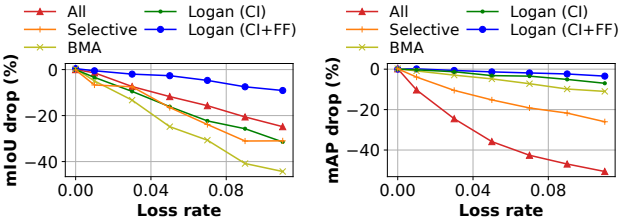


Figure 20: Performance under high loss rates.



(a) DAVIS dataset. **(b) Drone Racing dataset.**

Figure 21: Performance of various tasks.

perform slightly better under high burstiness since only a few frames are affected by the packet loss (although the error is severe for the few affected frames). Regardless of the level of burstiness, Logan consistently achieves the highest performance, demonstrating its robustness across different packet loss conditions.

7.2.7 Various Model Capacities. Figure 19 demonstrates the performance under various model capacities for KITTI dataset. We use YOLOv5 n, s, m, l. The accuracy of the systems slightly increases as the model capacity increases. Logan consistently outperforms the baselines for all model capacities.

7.2.8 Higher loss rates. Figure 20 presents a stress test of the system with the UAVDT dataset at loss rates up to 19%. As the loss rate increases, the performance gap with the baselines widens, with Logan showing 99% SLO satisfaction and an 11% accuracy drop at a 19% loss rate.

7.2.9 Other datasets. Figure 21 presents the performance metrics for additional datasets, revealing a pattern consistent with that observed in the UAVDT and KITTI datasets, albeit with some differences. For the DAVIS dataset, **All** exhibits moderate performance due to the small motion within the dataset, which allows the recent inference results to perform effectively. Also, **Selective** performs worse than **All** since the segmentation task is more sensitive to small errors. For the Drone Racing dataset, the task is relatively straightforward—detecting the nearby large gates through which the drone navigates. Consequently, **BMA** also demonstrates robust performance, as the analytics model is sufficiently resilient to accommodate significant errors in the frame for

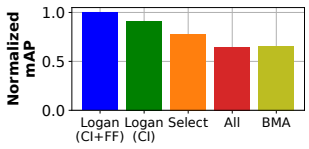


Figure 22: Peformance on real-world cellular bandwidth traces

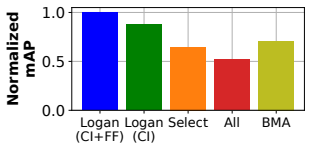


Figure 23: Performance on real-world packet loss traces.

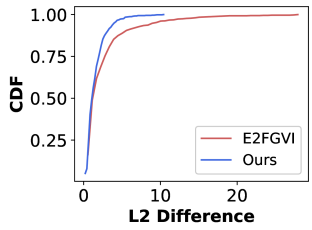


Figure 24: CDF of L2 differences between inpainted and ground truth frames.

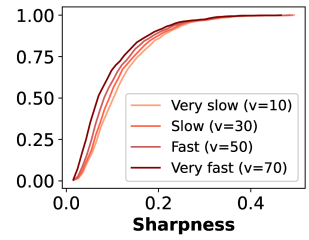


Figure 25: CDF of sharpness values with varying motion magnitudes (v).

such a simplified task. Note that our target scenarios, such as autonomous UAV delivery and AR criminal chasing, are likely to present a much higher level of complexity due to smaller and more diverse objects.

7.3 Performance with Real-world Traces

Bandwidth traces. We evaluate our system with real-world LTE bandwidth traces [47] collected on buses within the New York Metropolitan area, where bandwidth fluctuates from 10 Kbps to 12 Mbps (average: 7 Mbps). We use Mahimahi [49] bandwidth emulator with CoDel [50] queueing discipline. We use packet interval-based bandwidth estimation on the receiver side similar to SRT [59] video streaming protocol and send it to the sender for bitrate adaptation, resulting in an average packet loss rate of 6.4% with highly bursty pattern. Figure 22 shows that Logan outperforms the baselines. Employing a bitrate adaptation strategy that more aggressively increases the bitrate could lead to higher packet loss rates, further widening the performance gap.

Packet traces. We also collected packet loss traces in an urban indoor setting using a SAMSUNG Galaxy S23 Ultra on a 5G network. These traces contain latency and loss information for each packet. We transmitted a 720p, 30 FPS video to a server using SRT [59] with bitrate adaptation. The average packet loss rate was 6.1%, showing a sparser pattern than bandwidth emulation experiment. Thus, **BMA** performs relatively better than other baselines, while ours performs best as shown in Figure 23.

7.4 Performance of codec-aware inpainting

Comparison to conventional video inpainting model. We compare our inpainting model with a state-of-the-art flow-based E2FGVI [44] on NVIDIA Titan RTX (both half-precision). Our model shows a substantial latency reduction, processing a 1248x384-sized frame in 25 ms (15 ms on RTX 3090) compared to E2FGVI's 178 ms, thanks to our lightweight flow estimation and efficient decoder without complex content hallucination.

Figure 24 compares the L2 difference between the concealed and original frames. Codec-aware Inpainting achieves a lower difference due to its blurry output for difficult regions. This is particularly effective for fast-moving objects in the KITTI dataset, also achieving better object detection accuracy (E2FGVI 0.490 vs Ours 0.502).

Difficulty-aware sharpness control. We further analyze Codec-aware Inpainting's adaptability to input difficulty. Since the inpainting accuracy typically drops with large motions [44], we synthesize new datasets using KITTI dataset with varying motion magnitudes. The results are presented in Figure 25. With motions of higher magnitude, the model generates less sharp output, where the sharpness score is 0.09 for the motion of 70 pixels and 0.12 for the motion of 10 pixels. The trend suggests that the model appropriately renders more challenging regions in a blurrier manner.

8 Discussion and Future Works

Improving the DNN error resilience. While we focused on utilizing the error resilience already present in the DNNs, some works try to improve the error resilience of the DNNs through improved model architecture [14, 16] or training methodologies [55, 79]. Logan can leverage these robust DNNs to further improve the accuracy, though integrating such strategies requires additional training. Theoretical exploration of DNN resilience for explicit error management is also valuable but a long-term endeavor. We focused on demonstrating the potential of DNN's resilience and developing mechanisms that will remain relevant in the future, with theoretical advances enabling more sophisticated policies.

Supporting other codecs. While we focus on H.264, the mechanism is similar for other codecs. For instance, H.265 supports slice-based encoding similar to H.264, but it operates with a processing unit called a Coding Tree Unit (CTU), sized at 64x64. H.265 also offers an alternative mode, tile-based encoding, which differs from slice-based encoding in that the tiles typically have a more rectangular shape. Given variations in the shape of the transmission units across different codecs, Codec-aware Inpainting model can be improved by training with data generated using strategies tailored to each specific codec.

For codecs that lack error resilience features like slice coding (e.g., VP9), a single packet loss can make the entire frame and all subsequent frames undecodable. While Codec-aware Inpainting could still be applied, the tolerance of Logan would be lower as it cannot utilize packets that arrive after the lost one. Implementing error resilience features in the codec that allows leveraging partial frame information would improve the effectiveness of our system in such cases.

Integration with MCS control / Congestion control. The Modulation and Coding Scheme (MCS) determines the data rates of a wireless link. A more aggressive MCS increases data rates at the cost of a higher packet loss rate. Given the inherent robustness of Logan to packet losses, there is a potential for jointly optimizing Logan with MCS control logic to favor more aggressive settings, enhancing data rates without the usual risks. This could also apply to congestion control, potentially improving overall network throughput.

On-device processing. While increasingly powerful on-device computing offers reduced dependency on network connectivity and low latency for lightweight analysis models, server offloading provides significant benefits, including greater compute and memory capacity for running large models on high-resolution videos and ease of management (e.g., codebase and model management, failure recovery). Combined with the cost-effectiveness of installing a centralized powerful server compared to equipping each device with GPUs, offloading creates opportunities to deploy smaller devices on a larger scale, such as in drone delivery scenarios. Our paper explores the offloading approach, targeting contexts where its advantages can be fully leveraged.

9 Related Works

Video Error Concealment and Video Inpainting. Most of existing methods utilize the spatio-temporal redundancy of the motion vectors to estimate the motion vectors in the lost region. The methods range from using the average of surrounding macroblocks' motion vectors [65] to Boundary Matching Algorithms [10, 22, 39, 56, 63] that estimate the motion vector that results in the smoothest pixel boundary.

Video inpainting tasks in the computer vision community differ from video error concealment in that they do not assume any knowledge about the video codec and fill in the hole by looking only at the images. To preserve temporal consistency along the time dimension, the video inpainting DNNs typically incorporate 3D convolution [7, 31, 66], temporal transformer [4, 46, 74, 77], and optical flow [21, 44, 72]. However, they suffer from high computational complexity and show limited performance for encoded videos with packet losses due to the lack of codec awareness.

Loss-tolerant DNN inference. Some works exploit the DNN robustness to reduce the network overhead by allowing losses. However, most [33, 58, 68] assume a split inference setting where the intermediate feature map, instead of the video, is transmitted. The large size and low coding efficiency of feature maps incur significant transmission overhead while they become relatively loss-resilient. Another study [67] transmits the packetized video frames for coarse-grained activity recognition task but assumes the packet loss does not affect the subsequent frames, which actually behaves like an image encoding. We are the first to leverage the loss tolerance of DNNs in live video analytics with frame-wise fine-grained outputs, where a single packet loss affects multiple frames.

Loss-resilient Codecs. Several recent studies [11, 12] explored the use of neural autoencoders which can reduce the impact of packet loss on frame quality. However, the encoding and decoding overhead becomes significant, especially for mobile devices with limited computational resources and battery. Salsify [20] implemented a functional codec to enable the sender to re-synchronize to the receiver's state when packet loss occurs. Logan uses a similar state injection mechanism during Fast-Forward Recovery, re-decoding the frames with the late-arriving packets. However, Logan emphasizes the analysis of corrupted frames before they are fully recovered through retransmission or state synchronization.

10 Conclusion

In this paper, we have proposed Logan, a pioneering live video analytics system designed to thrive in the challenging landscape of cloud-based applications with stringent latency constraints where packet losses and late arrivals are prevalent. Our novel *loss acceptance & recovery* leverages the inherent error resilience of DNNs, a new axis that enables a stable end-to-end latency under fluctuating network. With Codec-aware Inpainting that accurately recovers the errors of packet losses and Fast-Forward Recovery that effectively limits the error propagation, Logan achieves 99.9% latency SLO satisfaction with <1% accuracy drop under 5% loss rate. We believe that shifting conventional packet loss management from solely relying on uncontrollable network resources to leveraging controllable computational resources will enable many emerging video analytics applications.

Acknowledgments

We sincerely thank our anonymous shepherd and reviewers for their valuable comments. This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP 2022-0-00531, Development of In-Network Computing Techniques for Efficient Execution of AI Applications) and National Research Foundation (NRF) funded by the Korean government (MSIT) (No. RS-2024-00463802). Youngki Lee is the corresponding author of this work.

References

[1] 3GPP. [n. d.]. *Unmanned Aerial System (UAS) support in 3GPP*. Technical Specification (TS) 22.125. 3rd Generation Partnership Project (3GPP). http://www.3gpp.org/DynaReport/22125.htm

[2] Mohammad Furqan Ali, Dushantha Nalin K Jayakody, Yury Alexandrovich Chursin, Soféine Affes, and Sonkin Dmitry. 2020. Recent advances and future directions on underwater wireless communications. *Archives of Computational Methods in Engineering* 27 (2020), 1379–1412.

[3] Michael Bosello, Davide Aguiari, Yvo Keuter, Enrico Pallotta, Sara Kiade, Gyordan Caminati, Flavio Pinzarrone, Junaid Halepota, Jacopo Panerati, and Giovanni Pau. 2024. Race Against the Machine: a Fully-annotated, Open-design Dataset of Autonomous and Piloted High-speed Flight. *IEEE Robotics and Automation Letters* (2024).

[4] Jiayin Cai, Changlin Li, Xin Tao, Chun Yuan, and Yu-Wing Tai. 2022. Devit: Deformed vision transformers in video inpainting. In *Proceedings of the 30th ACM International Conference on Multimedia*. 779–789.

[5] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based congestion control. *Commun. ACM* 60, 2 (2017), 58–66.

[6] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.

[7] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. 2019. Free-form video inpainting with 3d gated convolution and temporal patchgan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9066–9075.

[8] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 155–168.

[9] Weiqi Chen, Hua Yu, Quansheng Guan, Fei Ji, and Fangjiong Chen. 2018. Reliable and opportunistic transmissions for underwater acoustic networks. *IEEE Network* 32, 4 (2018), 94–99.

[10] Xiaoming Chen, Yuk Ying Chung, and Changseok Bae. 2008. Dynamic multi-mode switching error concealment algorithm for H. 264/AVC video applications. *IEEE Transactions on Consumer Electronics* 54, 1 (2008), 154–162.

[11] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. {GRACE}:{Loss-Resilient}{Real-Time} Video through Neural Codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 509–531.

[12] Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. 2022. Swift: Adaptive video streaming with layered neural codecs. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 103–118.

[13] Jörg Deutschmann, Saeid Jahandar, Kai-Steffen Hielscher, and Reinhard German. 2023. Internet via Satellite: GEO vs. LEO, OpenVPN vs. Wireguard, and CUBIC vs. BBR. In *Proceedings of the 1st ACM MobiCom Workshop on Satellite Networking and Computing*. 19–24.

[14] Steven Diamond, Vincent Sitzmann, Frank Julca-Aguilar, Stephen Boyd, Gordon Wetzstein, and Felix Heide. 2021. Dirty pixels: Towards end-to-end image processing and perception. *ACM Transactions on Graphics (TOG)* 40, 3 (2021), 1–15.

[15] Samuel Dodge and Lina Karam. 2016. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*. IEEE, 1–6.

[16] Samuel Dodge and Lina Karam. 2017. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119* (2017).

[17] Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, and Qi Tian. 2018. The unmanned aerial vehicle benchmark: Object detection and tracking. In *Proceedings of the European conference on computer vision (ECCV)*. 370–386.

[18] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. 2020. Server-driven video streaming for deep learning inference. 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*. 557–570.

[19] Edwin O Elliott. 1963. Estimates of error rates for codes on burst-noise channels. *The Bell System Technical Journal* 42, 5 (1963), 1977–1997.

[20] 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.

[21] Chen Gao, Ayush Saraf, Jia-Bin Huang, and Johannes Kopf. 2020. Flow-edge guided video completion. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*. Springer, 713–729.

[22] Zhi-Wei Gao and Wen-Nung Lie. 2004. Video error concealment by using Kalman-filtering technique. In *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, Vol. 2. IEEE, II–69.

[23] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 3354–3361.

[24] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Joseph E Gonzalez, and Ion Stoica. 2022. D3: a dynamic deadline-driven approach for building autonomous vehicles. In *Proceedings of the Seventeenth European Conference on Computer Systems*. 453–471.

[25] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. 2020. Serving {DNNs} like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 443–462.

[26] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review* 42, 5 (2008), 64–74.

[27] Mingcong Han, Hanze Zhang, Rong Chen, and Haibo Chen. 2022. Microsecond-scale Preemption for Concurrent GPU-accelerated DNN Inferences. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 539–558. https://www.usenix.org/conference/osdi22/presentation/han

[28] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. 2016. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. 123–136.

[29] Yishu Han, Salah Eddine Elayoubi, Ana Galindo-Serrano, Vineeth S Varma, and Malek Messai. 2018. Periodic radio resource allocation to meet latency and reliability requirements in 5G networks. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 1–6.

[30] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z Morley Mao, Feng Qian, and Zhi-Li Zhang. 2022. Vivisecting mobility management in 5G cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 86–100.

[31] Yuan-Ting Hu, Heng Wang, Nicolas Ballas, Kristen Grauman, and Alexander G Schwing. 2020. Proposal-based video completion. In

Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16. Springer, 38–54.

[32] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. 2018. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8981–8989.

[33] Sohei Itahara, Takayuki Nishio, and Koji Yamamoto. 2021. Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–6.

[34] International Telecommunication Union (ITU). 2001. *ITU-T G.1010 : End-user multimedia QoS categories*. ITU-T Recommendation G.1010. International Telecommunication Union (ITU). https://www.itu.int/rec/T-REC-G.1010-200111-I

[35] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713.

[36] 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.

[37] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 253–266.

[38] Glenn Jocher. 2020. *YOLOv5 by Ultralytics*. https://doi.org/10.5281/zenodo.3908559

[39] Hyun-Soo Kang, Yong-Woo Kim, and Tae-Yong Kim. 2006. A temporal error concealment method based on edge adaptive boundary matching. In *Advances in Image and Video Technology: First Pacific Rim Symposium, PSIVT 2006, Hsinchu, Taiwan, December 10-13, 2006. Proceedings 1*. Springer, 852–860.

[40] Alexander Kirillov, Kaiming He, Ross Girshick, and Piotr Dollár. 2017. A unified architecture for instance and semantic segmentation. In *CVPR*.

[41] Wai-Man Lam, Amy R Reibman, and Bede Liu. 1993. Recovery of lost or erroneously received motion vectors. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 5. IEEE, 417–420.

[42] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. 2020. Towards streaming perception. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 473–488.

[43] 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.

[44] Zhen Li, Cheng-Ze Lu, Jianhua Qin, Chun-Le Guo, and Ming-Ming Cheng. 2022. Towards an end-to-end framework for flow-guided video inpainting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 17562–17571.

[45] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge assisted real-time object detection for mobile augmented reality. In *The 25th annual international conference on mobile computing and networking*. 1–16.

[46] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenxiu Sun, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. 2021. Fuseformer: Fusing fine-grained information in transformers for video inpainting. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14040–14049.

[47] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. 2020. Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion. *Computer Networks* 182 (2020), 107515.

[48] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. 2021. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 610–625.

[49] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: accurate {Record-and-Replay} for {HTTP}. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 417–429.

[50] Kathleen Nichols, Van Jacobson, Andrew McGregor, and Jana Iyengar. 2018. Controlled Delay Active Queue Management. RFC 8289. https://doi.org/10.17487/RFC8289

[51] 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.

[52] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. 2017. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675* (2017).

[53] Anurag Ranjan and Michael J Black. 2017. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4161–4170.

[54] Irving S Reed and Gustave Solomon. 1960. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics* 8, 2 (1960), 300–304.

[55] Andras Rozsa, Manuel Gunther, and Terrance E Boult. 2016. Towards robust deep neural networks with BANG. *arXiv preprint arXiv:1612.00138* (2016).

[56] Ruolin Ruan, Ruimin Hu, and Zhongming Li. 2009. An effective video temporal error concealment method. In *2009 4th International Conference on Computer Science & Education*. IEEE, 683–685.

[57] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[58] Jiawei Shao and Jun Zhang. 2020. Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 1–6.

[59] Maria Sharabayko, Maxim Sharabayko, Jean Dube, Joonwoong Kim, and Jeongseok Kim. 2021. *The SRT Protocol*. Internet-Draft draft-sharabayko-srt-01. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-sharabayko-srt/01/ Work in Progress.

[60] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. 2018. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8934–8943.

[61] Mingxing Tan, Ruoming Pang, and Quoc V Le. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10781–10790.

[62] Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 402–419.

[63] Tanaphol Thaipanich, Ping-Hao Wu, and C-C Jay Kuo. 2008. Low-complexity video error concealment for mobile applications using OBMA. *IEEE Transactions on Consumer Electronics* 54, 2 (2008), 753–761.

[64] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. 2016. Examining the impact of blur on recognition by convolutional networks. *arXiv preprint arXiv:1611.05760* (2016).

[65] Masahiro Wada. 1989. Selective recovery of video packet loss using error concealment. *IEEE Journal on Selected areas in Communications* 7, 5 (1989), 807–814.

[66] Chuan Wang, Haibin Huang, Xiaoguang Han, and Jue Wang. 2019. Video inpainting by jointly learning temporal structure and spatial details. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5232–5239.

[67] Haodong Wang, Kuntai Du, and Junchen Jiang. 2022. Minimizing packet retransmission for real-time video analytics. In *Proceedings of the 13th Symposium on Cloud Computing*. 340–347.

[68] Song Wang and Xinyu Zhang. 2022. NeuroMessenger: Towards Error Tolerant Distributed Machine Learning Over Edge Networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2058–2067.

[69] Xiaorui Wu, Hong Xu, and Yi Wang. 2020. Irina: Accelerating dnn inference with efficient online scheduling. In *Proceedings of the 4th Asia-Pacific Workshop on Networking*. 36–43.

[70] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption. 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*. 479–494.

[71] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas S. Huang. 2018. YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark. *CoRR* abs/1809.03327 (2018). http://arxiv.org/abs/1809.03327

[72] Rui Xu, Xiaoxiao Li, Bolei Zhou, and Chen Change Loy. 2019. Deep flow-guided video inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3723–3732.

[73] Fei Yan, Hongbin Luo, Shan Zhang, Zhiyuan Wang, and Peng Lian. 2023. A comparative study of IP-based and ICN-based link-state routing protocols in LEO satellite networks. *Peer-to-Peer Networking and Applications* 16, 6 (2023), 3032–3046.

[74] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. 2020. Learning joint spatial-temporal transformations for video inpainting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, 528–543.

[75] Yong Zeng, Ismail Guvenc, Rui Zhang, Giovanni Geraci, and David W Matolak. 2020. *UAV Communications for 5G and Beyond*. John Wiley & Sons.

[76] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. 2018. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 236–252.

[77] Kaidong Zhang, Jingjing Fu, and Dong Liu. 2022. Flow-guided transformer for video inpainting. In *European Conference on Computer Vision*. Springer, 74–90.

[78] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

[79] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*. 4480–4488.

[80] Yiren Zhou, Sibo Song, and Ngai-Man Cheung. 2017. On classification of distorted images with deep convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1213–1217.