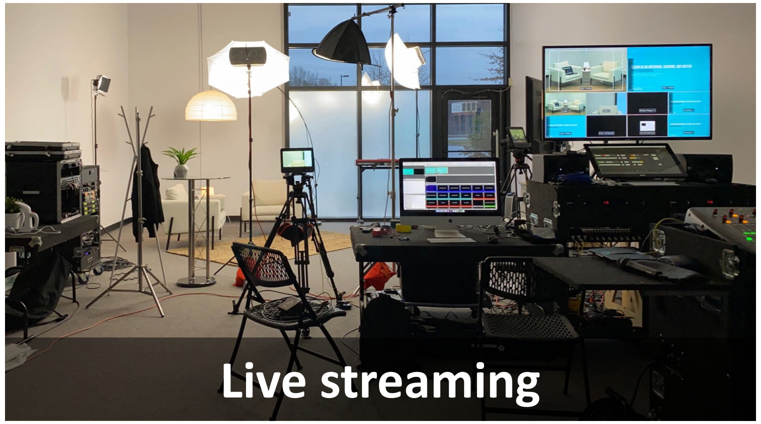
GRACE: Loss-Resilient Real-Time Video through Neural Codecs

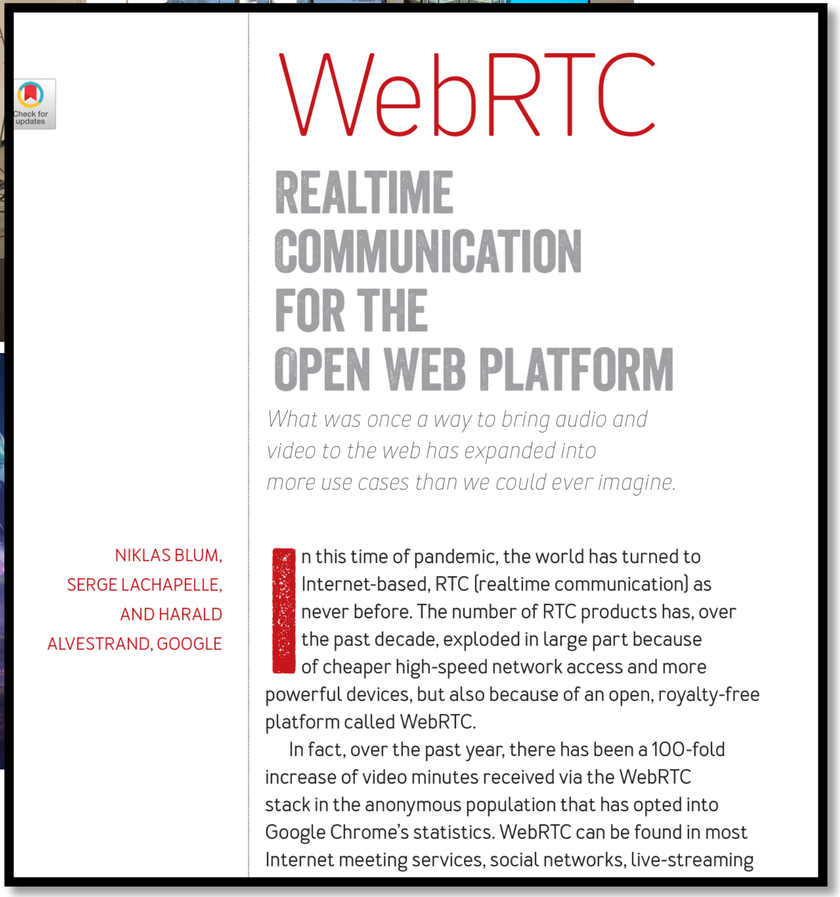
Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang,
Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang,
Francis Y. Yan, Amrita Mazumdar, Nick Feamster, Junchen Jiang

Real-time video streaming is prevalent



Real-time video streaming is prevalent

"...over the past year, there has been a ***100-fold*** *increase of video minutes received via the WebRTC stack..."*



WebRTC

REALTIME COMMUNICATION FOR THE OPEN WEB PLATFORM

What was once a way to bring audio and video to the web has expanded into more use cases than we could ever imagine.

NIKLAS BLUM, SERGE LACHAPELLE, AND HARALD ALVESTRAND, GOOGLE

In this time of pandemic, the world has turned to Internet-based, RTC (realtime communication) as never before. The number of RTC products has, over the past decade, exploded in large part because of cheaper high-speed network access and more powerful devices, but also because of an open, royalty-free platform called WebRTC.

In fact, over the past year, there has been a 100-fold increase of video minutes received via the WebRTC stack in the anonymous population that has opted into Google Chrome's statistics. WebRTC can be found in most Internet meeting services, social networks, live-streaming

Goal: stream video w/ constantly low delay and high bitrate

(No time to wait for retransmission)

Challenge: packet losses are hard to avoid

~1.5% frames lose* 20%-80% of their packets burstly within a frame.[1,2]

*Frame-level packet loss includes both packets dropped in network and packets not received by the decoding deadline

[1] Hairpin: Rethinking Packet Loss Recovery in Edge-based Interactive Video Streaming (NSDI'24)

[2] Tambur: Efficient loss recovery for videoconferencing via streaming codes (NSDI'23)

Common intuition: lowering packet loss → better user-perceived quality

Is one video obviously better than another video?

Video A:

Video B:

A is much better

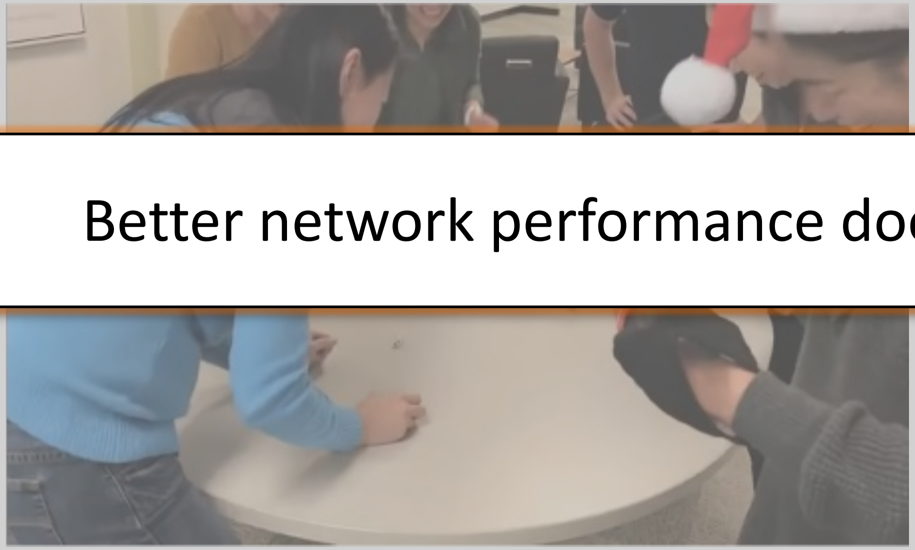
No significant difference

B is much better

Survey: which video is better?

A: 50% loss between 2-2.15 sec

B: 15% loss between 2-2.15 sec



Better network performance doesn't always improve user perceived quality

A is obviously better

No significant difference

B is obviously better

How receiver deals with packet losses

Error concealment: "guesses" the missing piece based on arrived data **at receiver side**

Video content packets

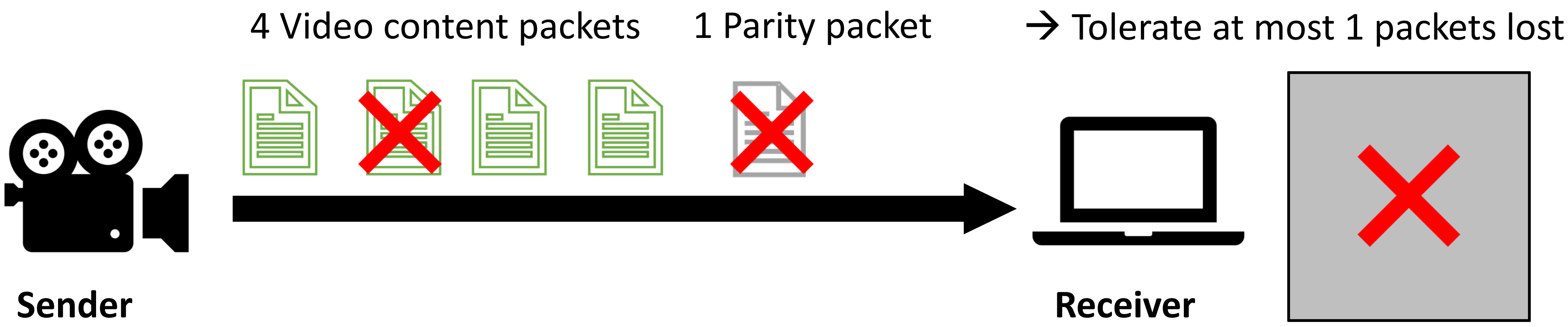
I will use most efficient compression without any redundancy

I will guess the content in the lost packets

Issue: Guessing missing data is fundamentally difficult without redundant information

How sender deals with packet losses

Forward error correction (FEC): protect **at sender-side**

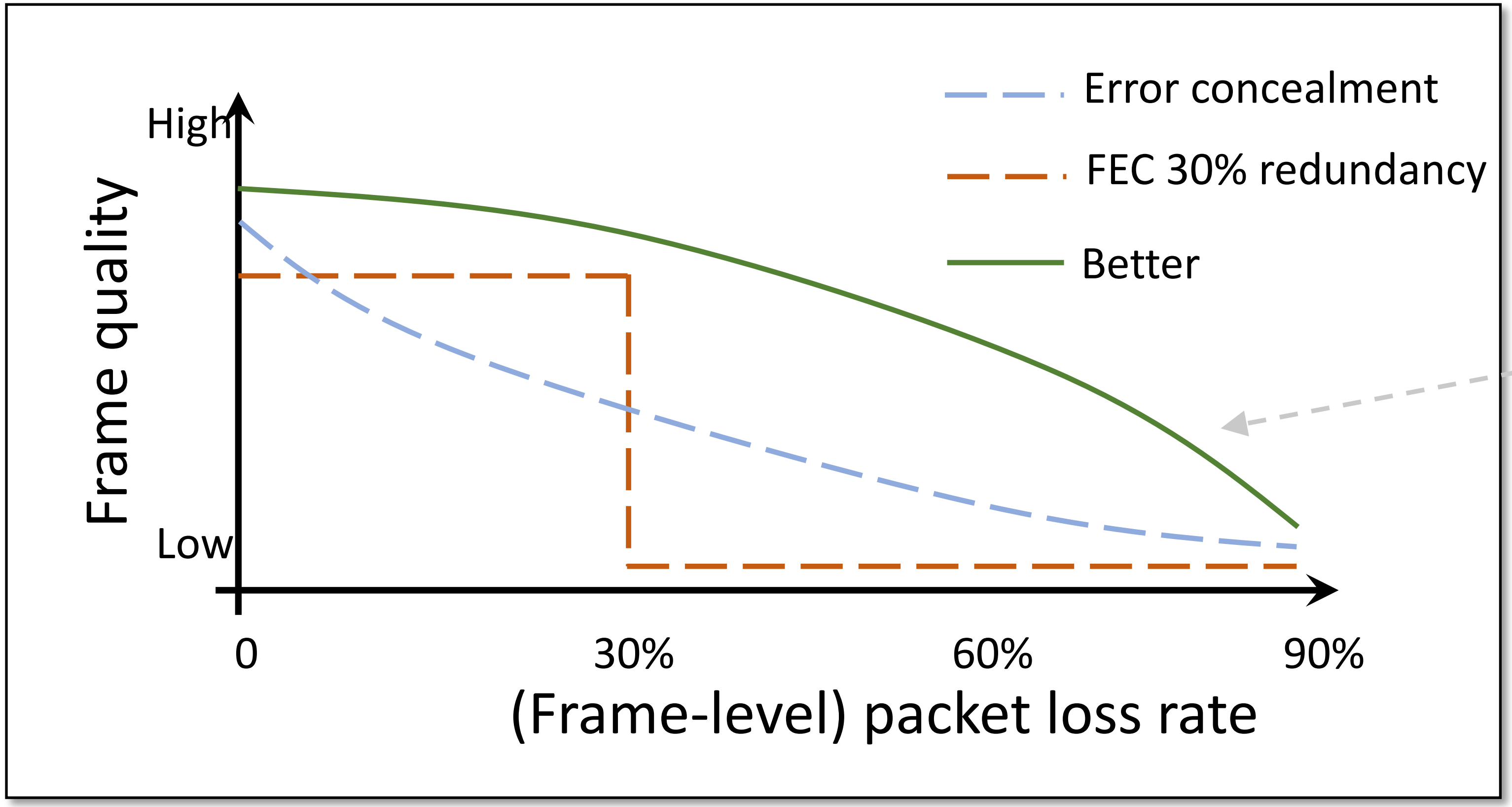


I’ll sacrifice bitrate and send extra parity packets to protect against packet losses

Too many packets lost, I cannot decode

Issue: FEC is effective only if packet loss is lower than its redundancy rate

Summary of existing solutions



How to improve loss-resilience (i.e., green line)?

What's missing?

The video sender and receiver are often jointly optimized for *compression efficiency.*

However, the video sender and receiver are rarely ***jointly optimized for loss resilience.***

Our insight: **jointly optimizing** the encoder and decoder under different packet losses considerably improves loss resilience

Specifically, GRACE achieved better loss resilience by jointly **training neural** encoder and decoder under **a range** of **simulated packet losses**

Our insight: **jointly optimizing** the encoder and decoder under different packet losses considerably improves loss resilience

Q1: How to train neural video codec?

Specifically, GRACE achieved better loss resilience by jointly **training neural** encoder and decoder under **a range** of **simulated packet losses**

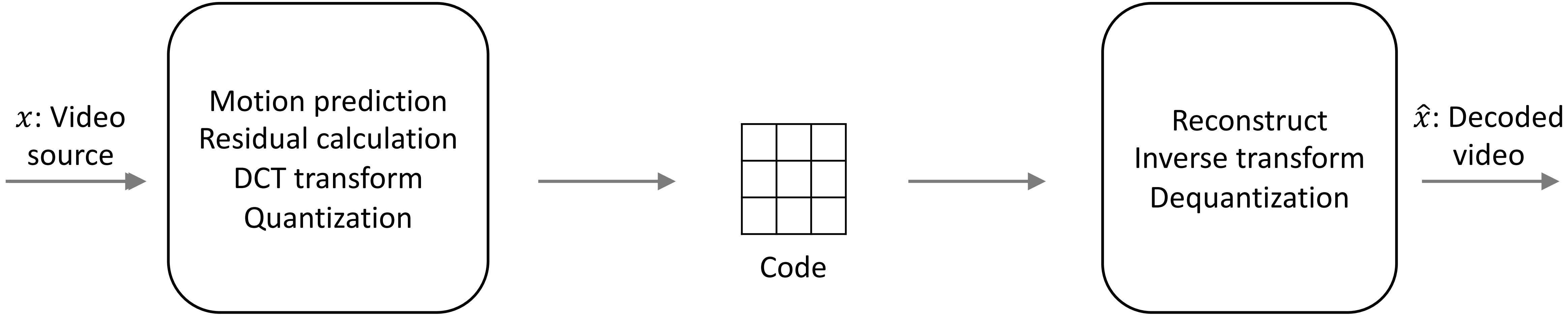
Q3: How to choose the packet loss rates in training

Q2: How to simulate packet loss in training?

Video codecs and neural video codecs

The encoder takes in video and producing the code

Decoder reconstructs the video based on the code



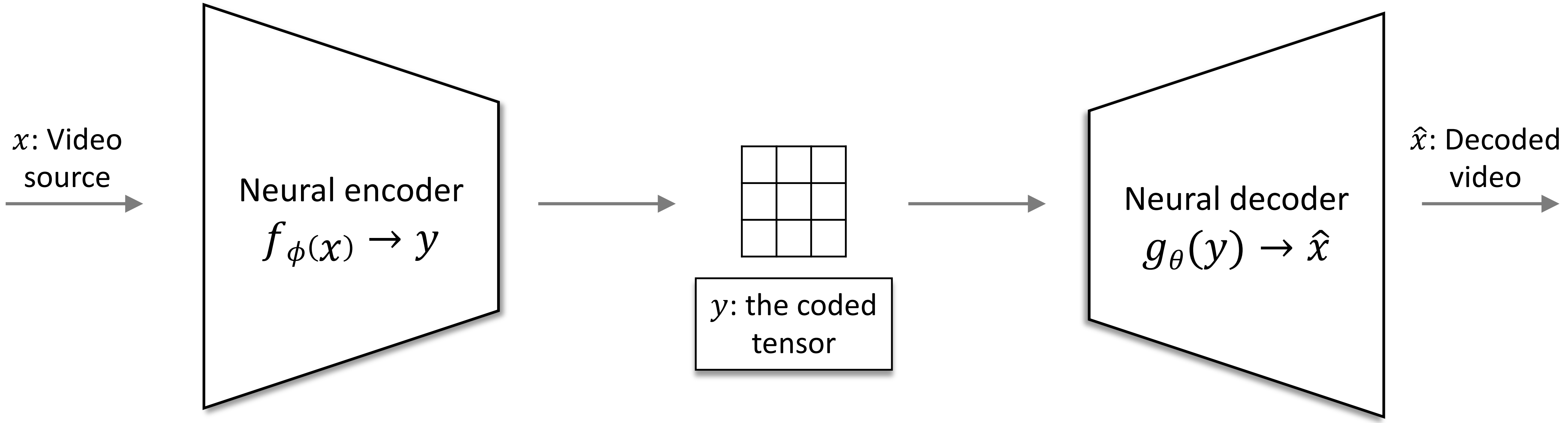
For simplicity, the entropy encoding step is omitted, since it does not change the code

Video codecs and **neural video codecs**

Replacing modules in traditional codecs with neural networks

Encoder and decoder can be seen as trainable functions f_ϕ and g_θ

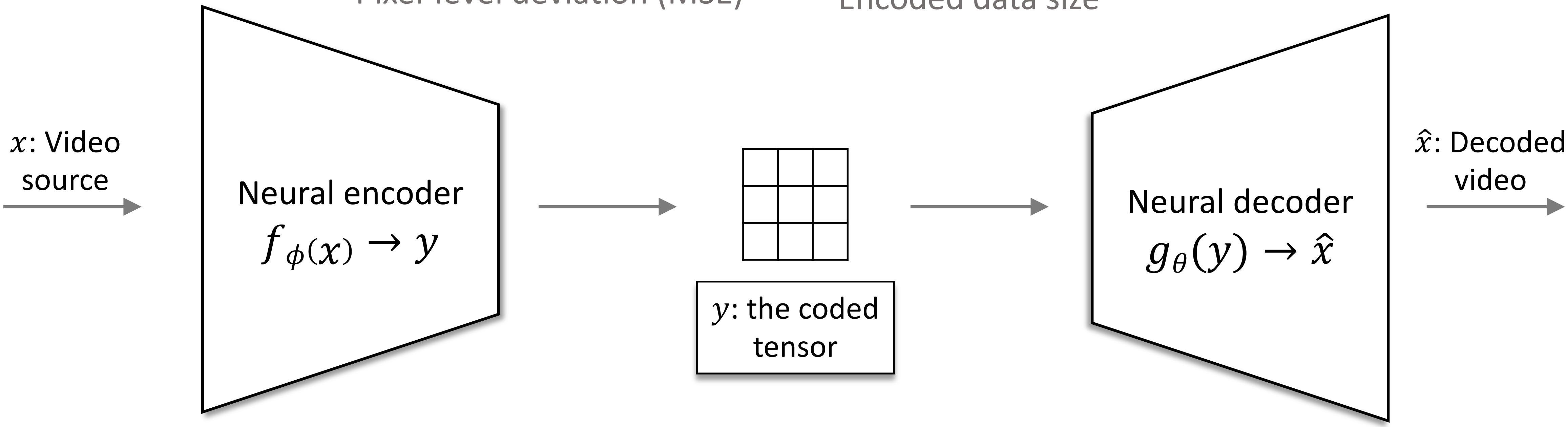
Specifically, GRACE employs the NN architecture from DVC (CVPR’19)



How to train a neural video codec

<u>Training objective</u>

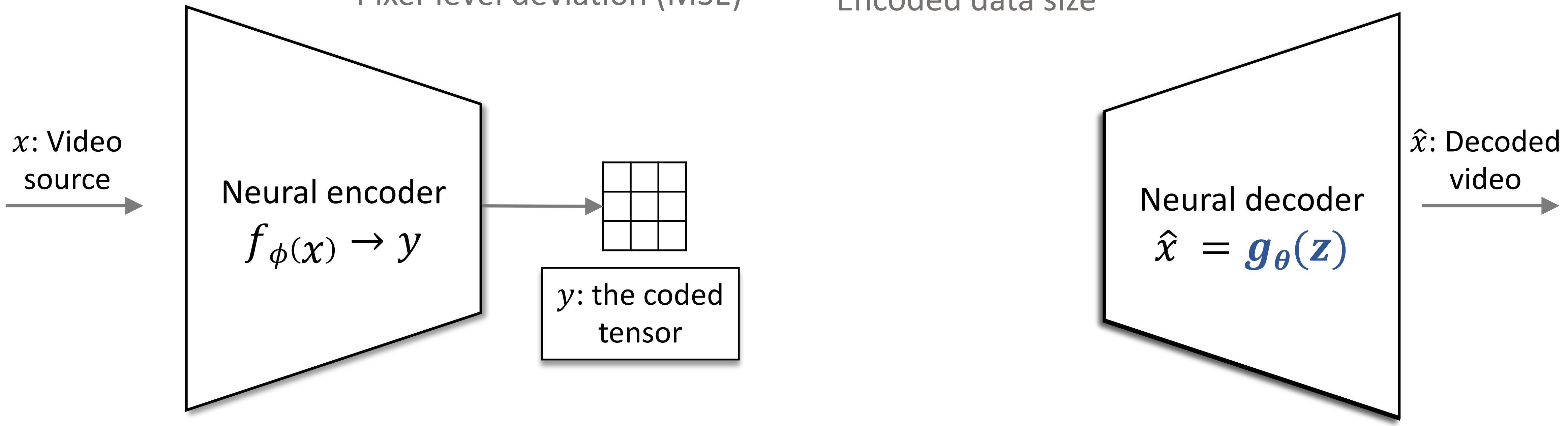
Minimize $\underbrace{Error(x, \hat{x})}_{\text{Pixel-level deviation (MSE)}} + \alpha \cdot \underbrace{Size(y)}_{\text{Encoded data size}}$ where $\begin{aligned} y &= f_\phi(x) \\ \hat{x} &= g_\theta(y) \end{aligned}$



How to train a neural video codec **with packet loss**

Training objective

$$\textbf{Minimize}\ \underbrace{Error(x, \hat{x})}_{\text{Pixel-level deviation (MSE)}} + \alpha \cdot \underbrace{Size(y)}_{\text{Encoded data size}} \quad \text{where} \quad \begin{aligned} y &= f_\phi(x) \\ \hat{x} &= g_\theta(y) \end{aligned}$$

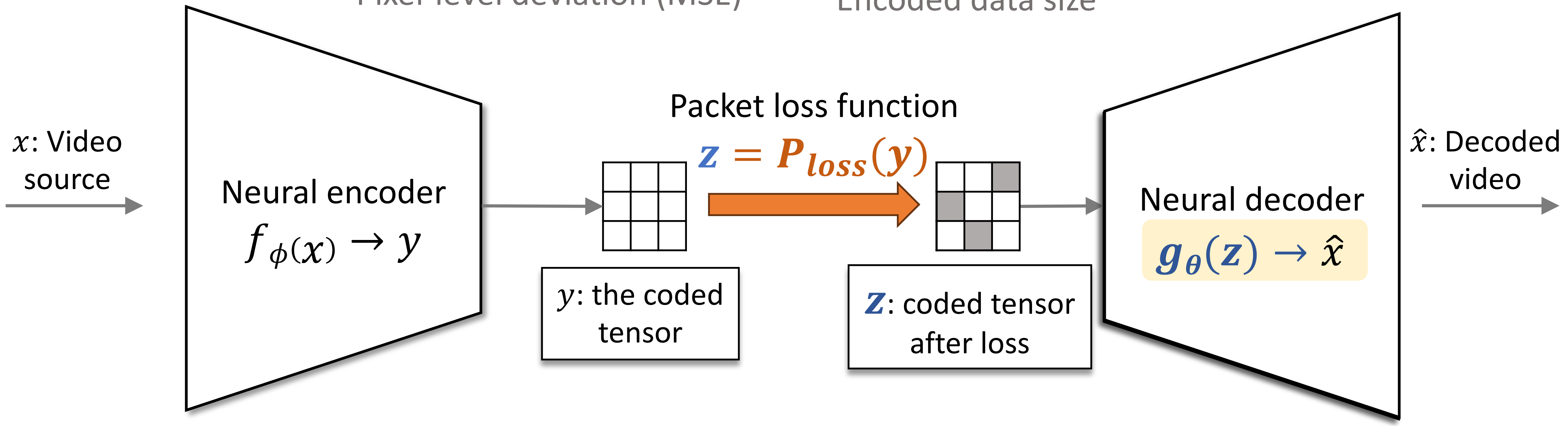


How to train a neural video codec **with packet loss**

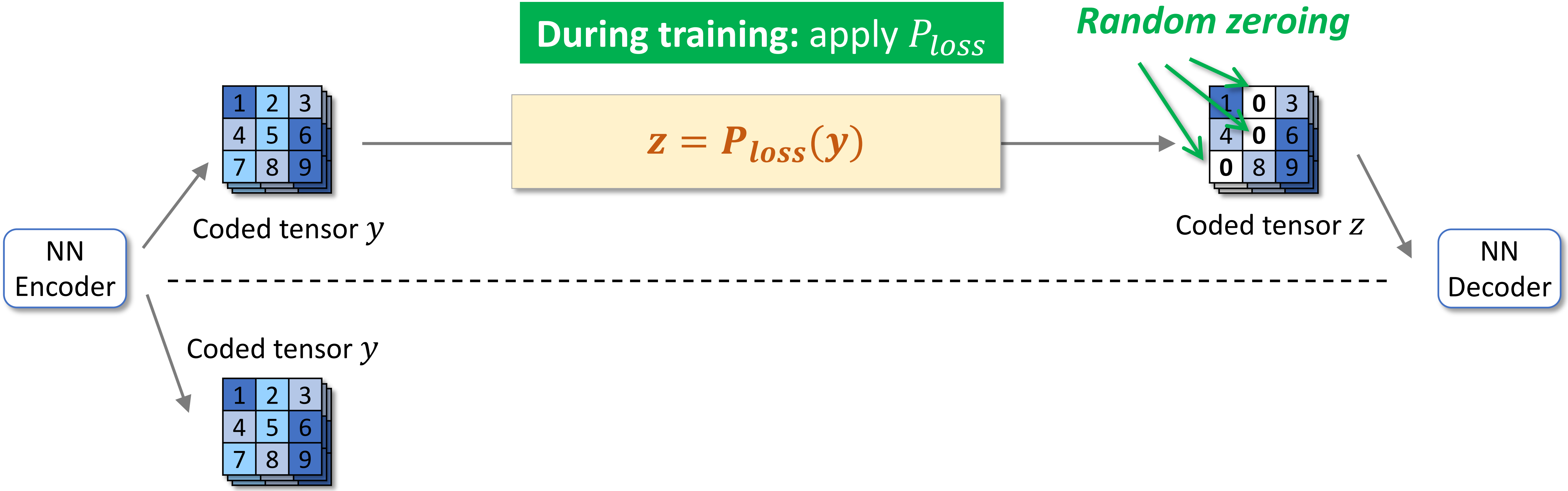
<u>Training objective</u>

Minimize $\underbrace{Error(x, \hat{x})}_{\text{Pixel-level deviation (MSE)}} + \alpha \cdot \underbrace{Size(y)}_{\text{Encoded data size}}$ where

$$y = f_\phi(x)$$
$$z = P_{loss}(y)$$
$$\hat{x} = g_\theta(z)$$

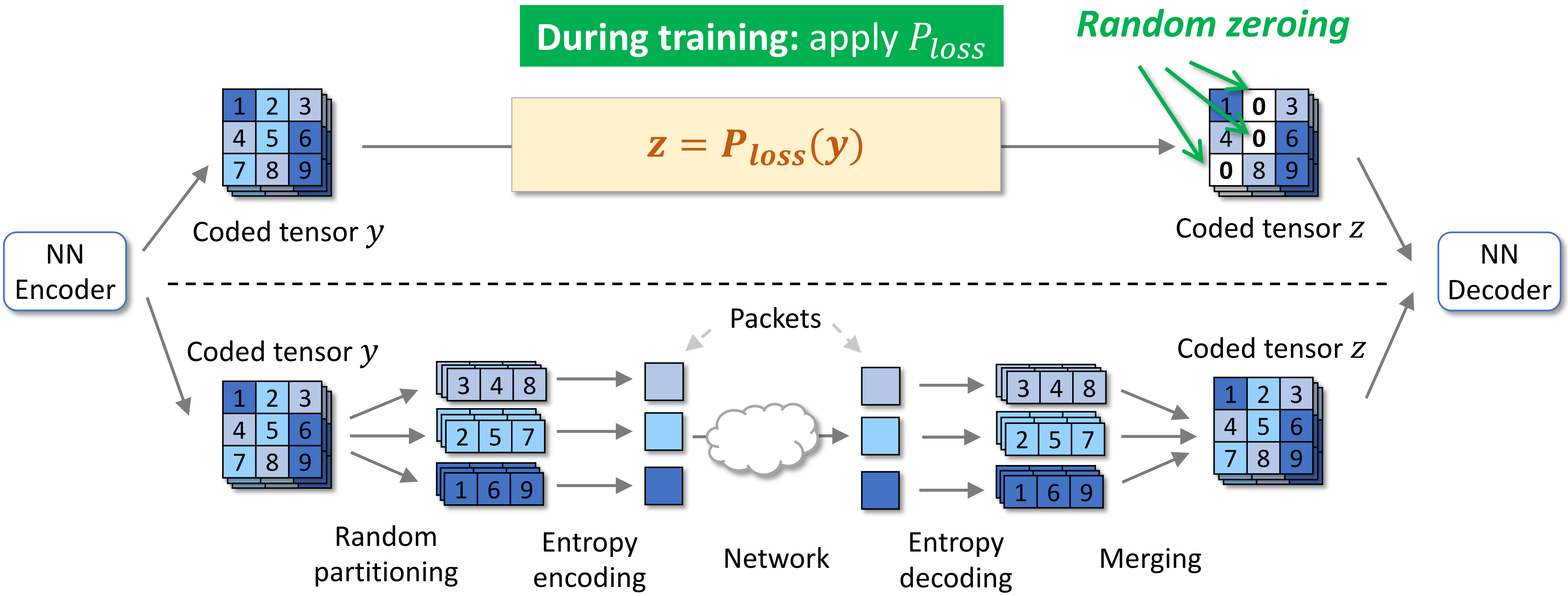


Packet loss during training and inferencing



During inferencing: packet loss should ***<u>have the same effect</u>*** as during training

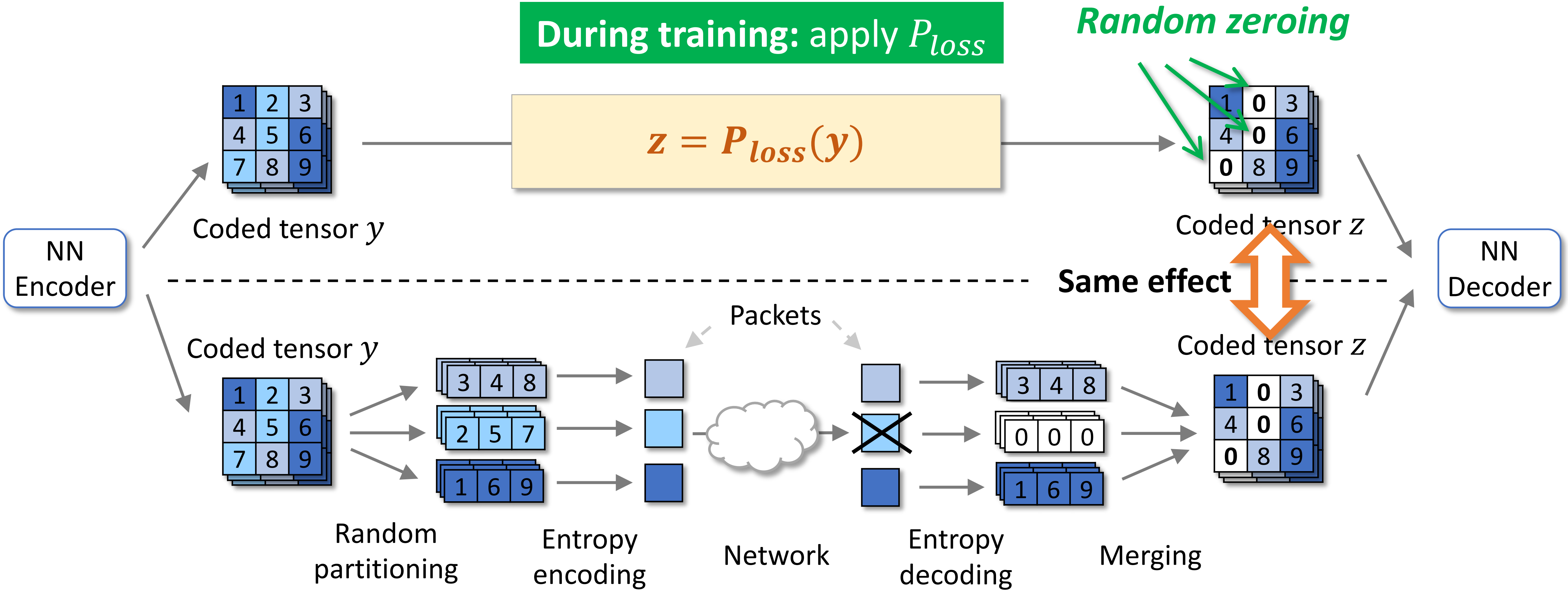
Packet loss during training and inferencing



During inferencing: packet loss should ***have the same effect*** as during training

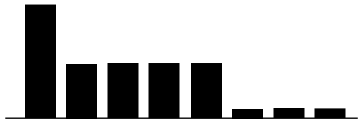
Packet loss during training and inferencing

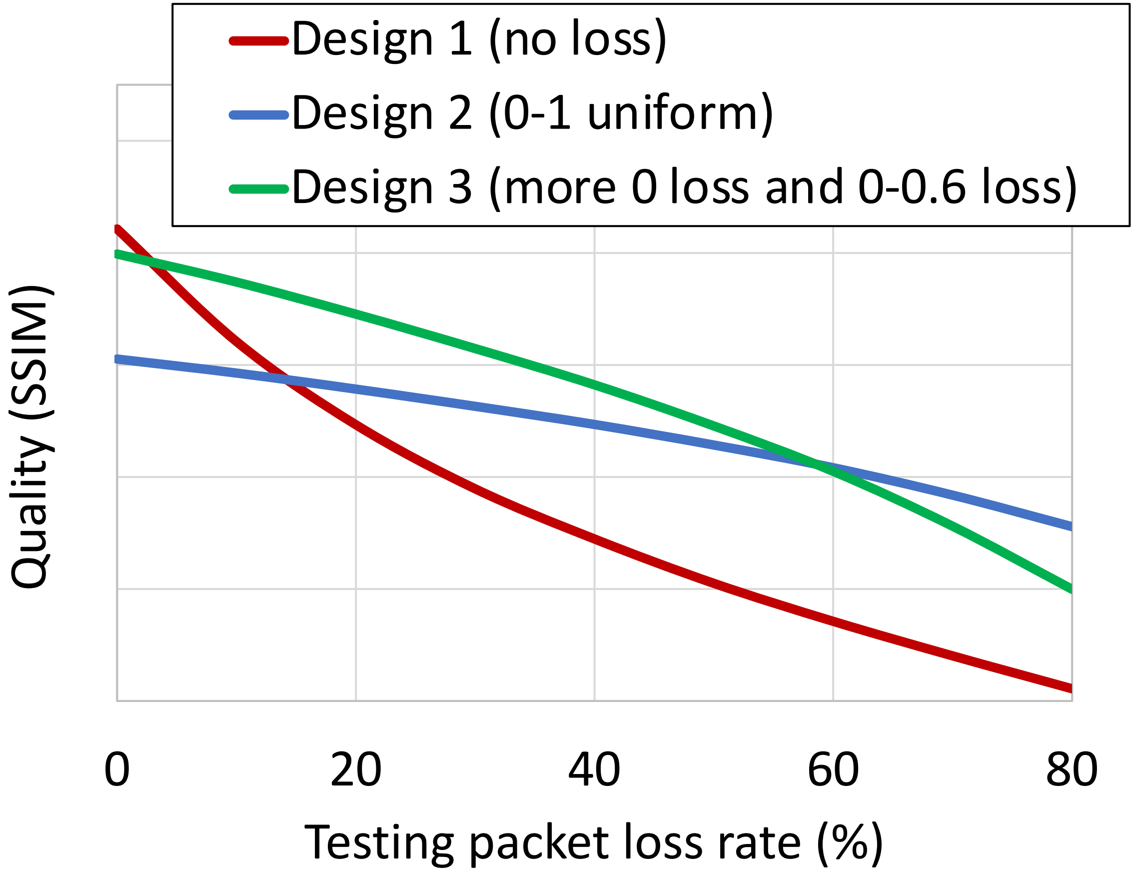
During training: apply P_{loss}



During inferencing: packet loss should ***<u>have the same effect</u>*** as during training

How to choose packet loss rate during training

Choice	pkt loss rates in training	Testing quality
Design 1 0 loss in training	 0% 90%	Poor quality under high loss rates
Design 2 uniform (0-100%)	 0% 90%	Poor quality under 0% loss
Design 3 more 0% loss more 0-60%	 0% 90%	Better quality under low loss rates Decent quality under high loss rates



Other issues addressed in the paper

- Real time (30fps) encoding/decoding on both GPU and mobile devices
- Fast bitrate adaptation
- Catch-up logic to minimize error-propagation

GRACE: Loss-Resilient Real-Time Video through Neural Codecs

Yihua Cheng[1], Ziyi Zhang[1], Hanchen Li[1], Anton Arapin[1], Yue Zhang[1], Qizheng Zhang[2], Yuhan Liu[1], Kuntai Du[1], Xu Zhang[1], Francis Y. Yan[3], Amrita Mazumdar[4], Nick Feamster[1], Junchen Jiang[1]
[1]*The University of Chicago*, [2]*Stanford University*, [3]*Microsoft*, [4]*NVIDIA*

Abstract

In real-time video communication, retransmitting lost packets over high-latency networks is not viable due to strict latency requirements. To counter packet losses without retransmission, two primary strategies are employed—encoder-based forward error correction (FEC) and decoder-based error concealment. The former encodes data with redundancy before transmission, yet determining the optimal redundancy level in advance proves challenging. The latter reconstructs video from partially received frames, but dividing a frame into independently coded partitions inherently compromises compression efficiency, and the lost information cannot be effectively recovered by the decoder without adapting the encoder.

We present a loss-resilient real-time video system called GRACE, which preserves the user's quality of experience (QoE) across a wide range of packet losses through a new neural video codec. Central to GRACE's enhanced loss resilience is its *joint training of the neural encoder and decoder under a spectrum of simulated packet losses*. In lossless scenarios, GRACE achieves video quality on par with conventional codecs (e.g., H.265). As the loss rate escalates, GRACE exhibits a more graceful, less pronounced decline in quality, consistently outperforming other loss-resilient schemes. Through extensive evaluation on various videos and real network traces, we demonstrate that GRACE reduces undecodable frames by 95% and stall duration by 90% compared with FEC, while markedly boosting video quality over error concealment methods. In a user study with 240 crowdsourced participants and 960 subjective ratings, GRACE registers a 38% higher mean opinion score (MOS) than other baselines. We make the source codes and models of GRACE public at https://uchi-jcl.github.io/grace.html.

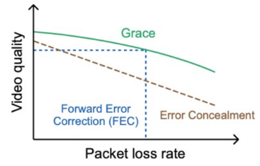


Figure 1: *Illustration of the video quality achieved by different loss-resilient schemes, operating under the same bandwidth budget, across varying packet loss rates. Actual experimental results are shown in Figure 8.*

Loss-resilient techniques generally fall into two categories. First is encoder-side forward error correction (FEC), such as Reed-Solomon codes [100], fountain codes [76,77], and more recently, streaming codes [28,86]. FEC incorporates redundancy into data prior to transmission. With a redundancy rate of $R\%$—the percentage of redundant data relative to the total data size—up to $R\%$ of lost data can be recovered. Beyond that, the video becomes undecodable, rendering a sharp collapse in video quality (Figure 1). Increasing R protects against higher losses but also entails a higher bandwidth overhead, which in turn reduces video quality. Thus, determining the optimal R in advance poses a practical challenge.

The second category is decoder-side error concealment, which reconstructs portions of a video frame affected by packet losses, through handcrafted heuristics [63,97,115] or neural networks [59,67,79,87,102]. Nevertheless, implementing error concealment requires partitioning a video frame into independently decodable units (e.g., slices [99] or tiles [64]) first, thus reducing compression efficiency. Moreover, since the encoder is not optimized for loss resilience, the lost information cannot be effectively recovered by the decoder alone

Evaluating GRACE's performance

Dataset:

Training: Vimeo-90K

Testing: 61 videos from other datasets →

Dataset	# of videos	Length (s)	Size	Description
Kinetics	45	450	720p 360p	Human actions and interaction with objects
Gaming	5	100	720p	PC game recordings
UVG	4	80	1080p	HD videos (human, nature, sports, etc.)
FVC	7	140	1080p	In/outdoor video calls
Total	**61**	**770**		

Quality metric:

SSIM in dB averaged across all frames

(calculated by $-10\log(1 - SSIM)$)

Baseline 1: WebRTC (H.265) w/ FEC

Use latest FEC technique (Tambur, NSDI 23)

https://www.usenix.org/conference/nsdi23/presentation/rudow

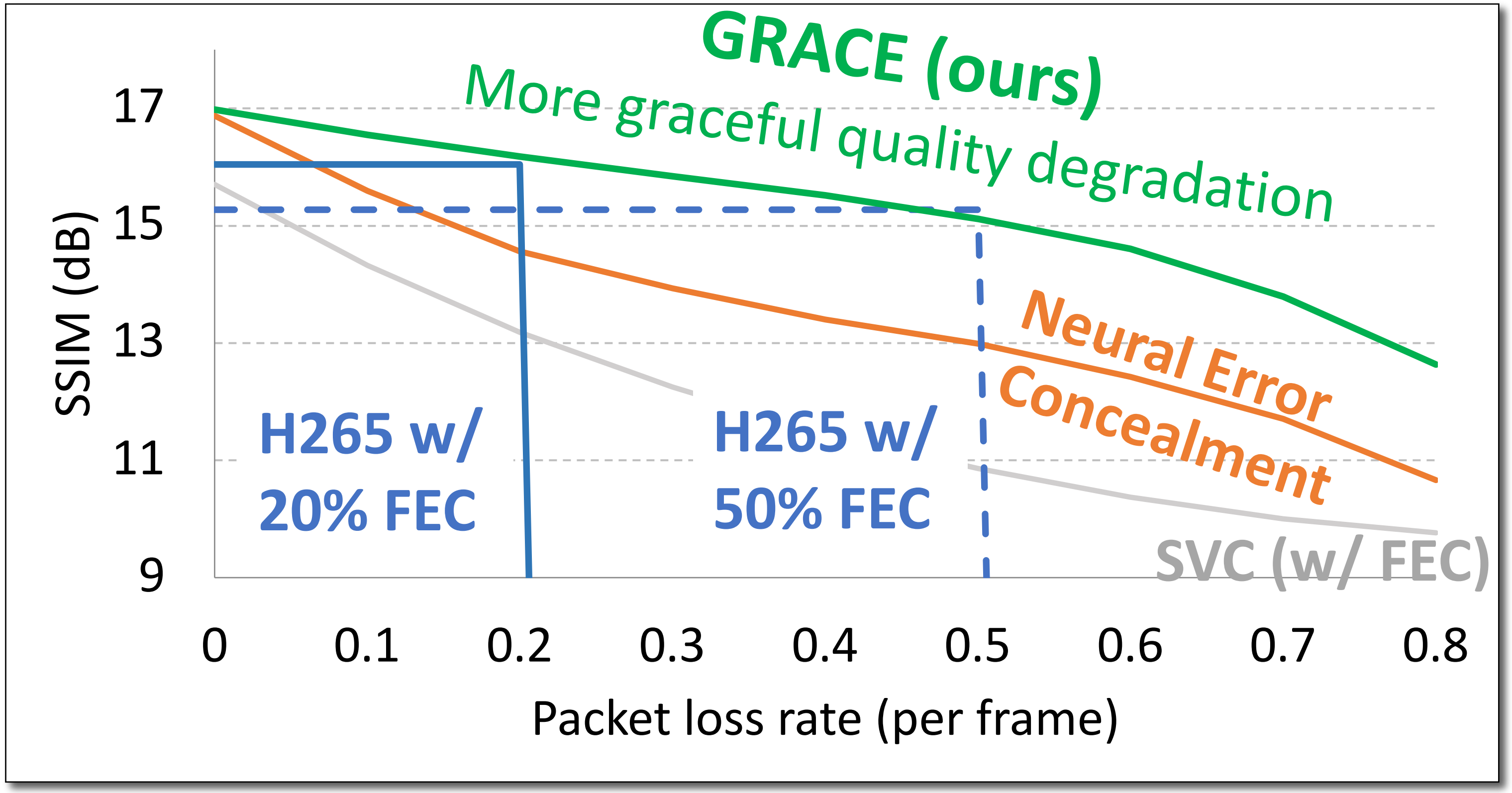
Baseline 2: Error concealment

Use flow-guided NN-based technique (ECCV 2022) https://arxiv.org/abs/2208.06768

Baseline 3: SVC w/ FEC at base layer

Quality under different packet loss rates

Target bitrate 3Mbps



Better quality under various packet loss rates than other loss-resilient schemes.

Visualization of loss-recovered images

Original frame

Error concealment
SSIM: 10.9 dB

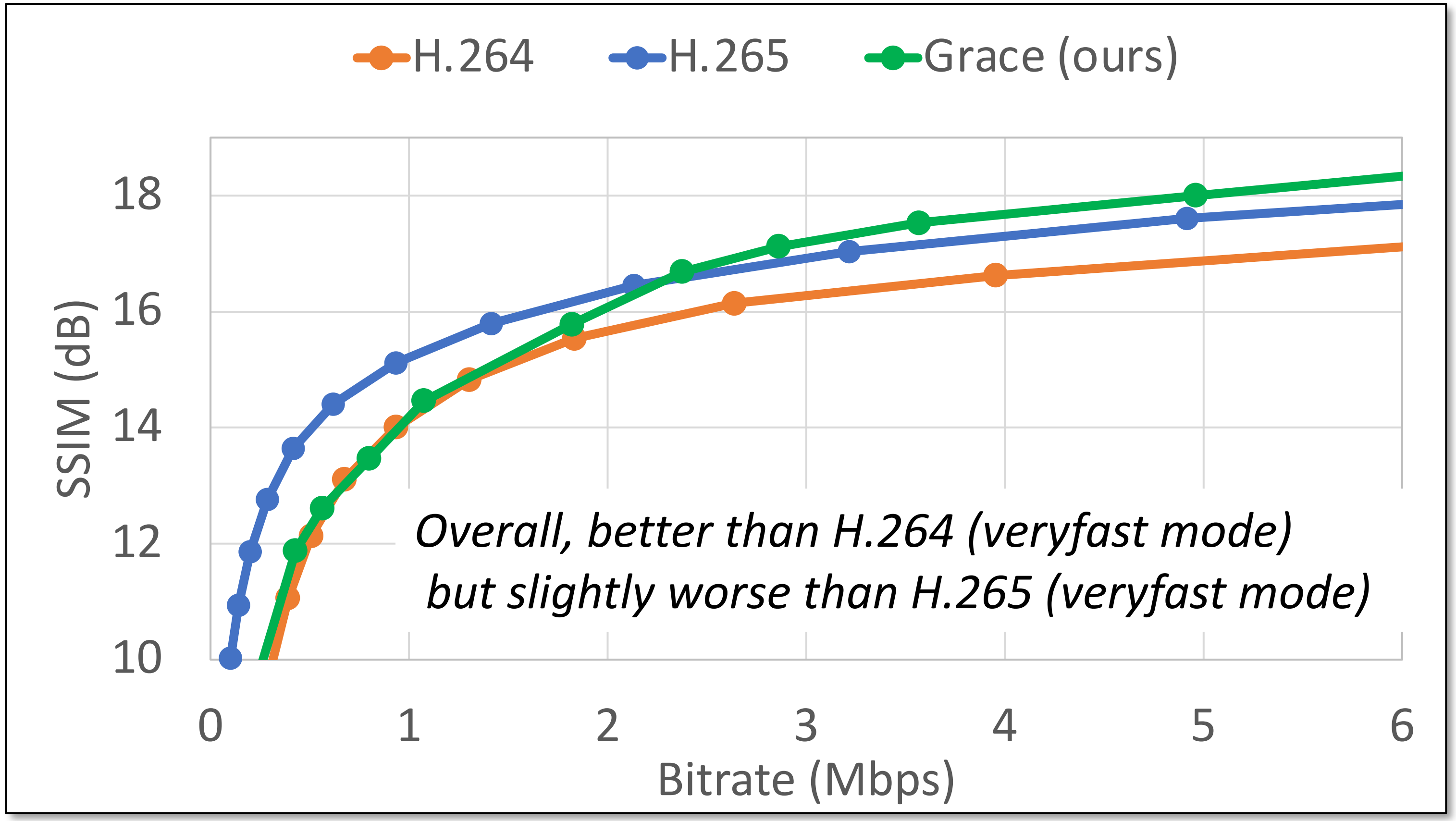
GRACE
SSIM: 12.0 dB

Sample images decoded by GRACE and error concealment under 50% packet loss applied on three consecutive frames

Visually, Grace-recovered images look more decent

Compression efficiency under 0 packet loss

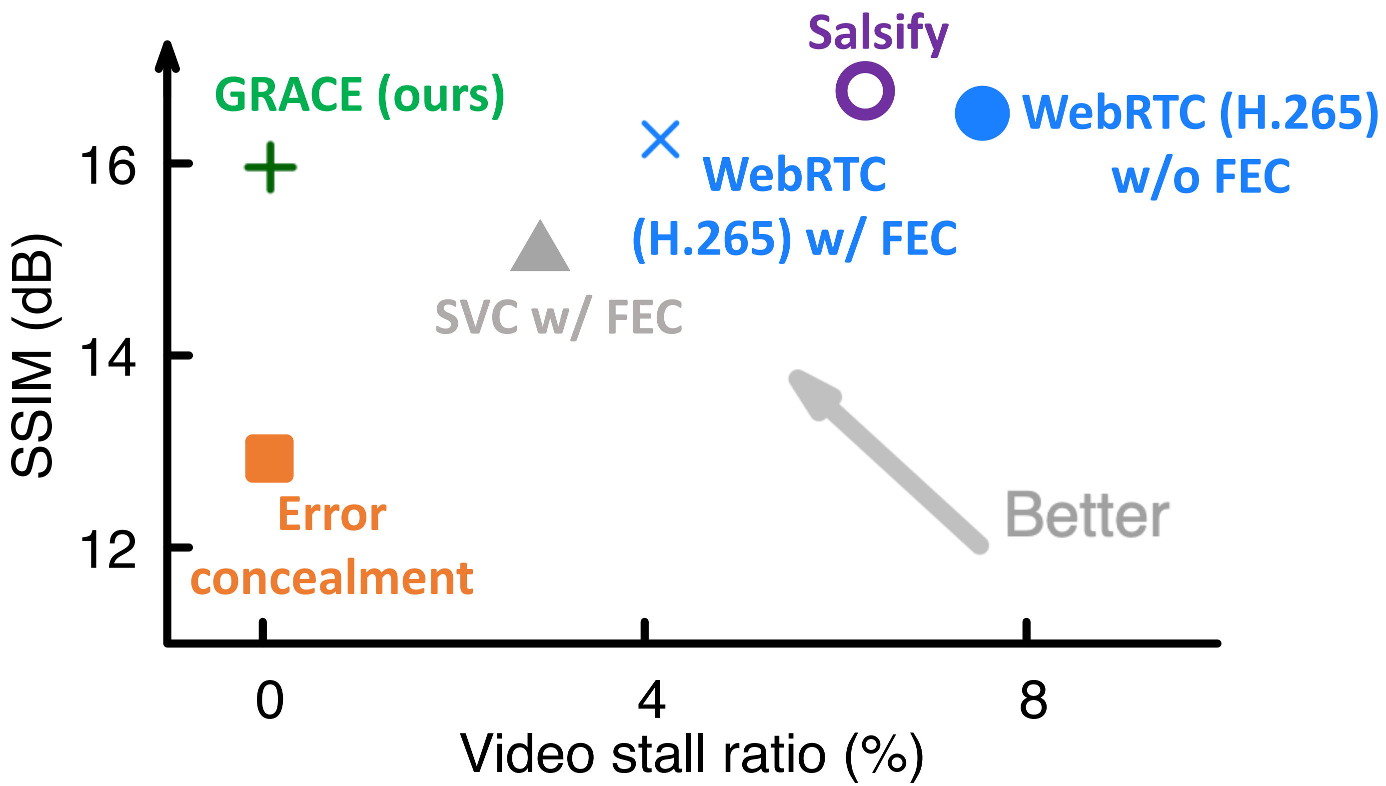
45 (720p) videos
40 "Kinetics" videos for 450s,
5 "Gaming" videos for 100 secs



GRACE doesn't sacrifice compression efficiency for loss resilience

Quality & smoothness tradeoff in real networks

End-to-end 400 sec video streaming
Bandwidth varies between 0.2-10Mbps
150ms RTT, 64KB queue
Use GCC as congestion control
Test videos from Kinetics dataset

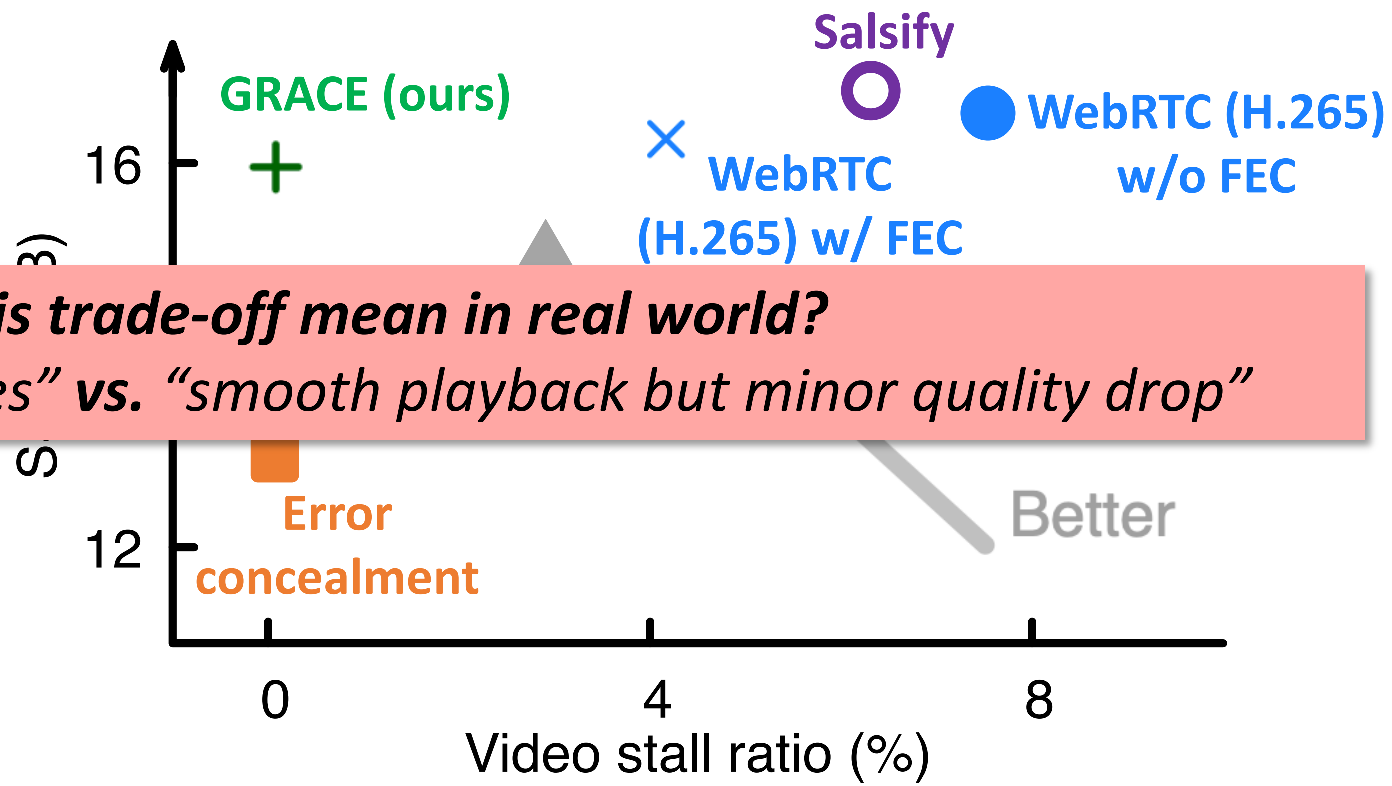


GRACE keeps smoother playback with minor quality drop in real network scenarios

Quality & smoothness tradeoff in real networks

End-to-end 400 sec video streaming

Test videos from Kinetics dataset



What does this trade-off mean in real world?

“High quality but minor glitches” **vs.** *“smooth playback but minor quality drop”*

GRACE keeps smoother playback with minor quality drop in real network scenarios

What does this tradeoff visually look like?

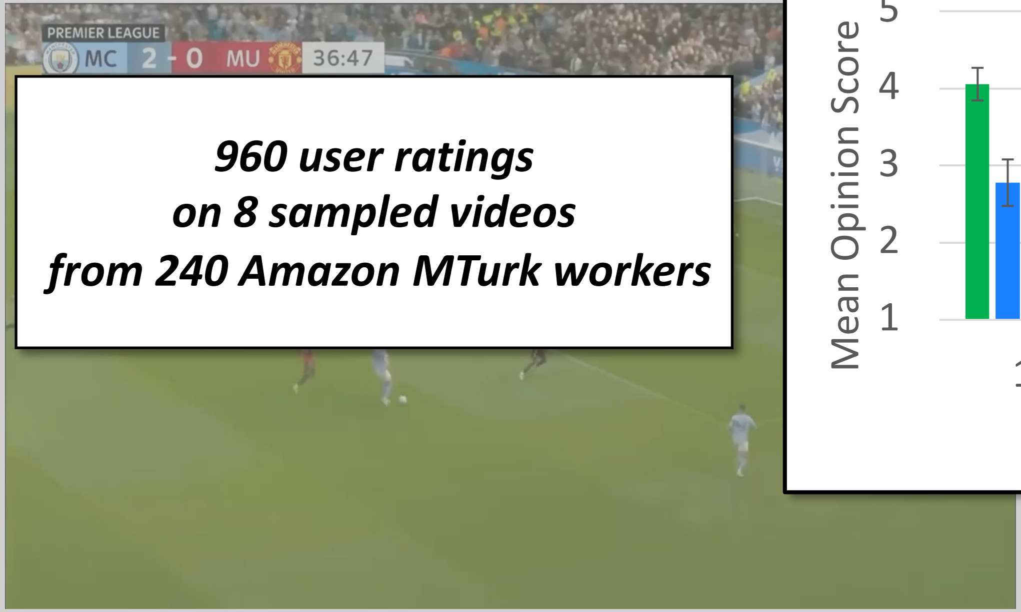
WebRTC w/ FEC
(high quality but minor glitch)

GRACE
(smoother playback yet minor quality drop)

What does this tradeoff visually look like?

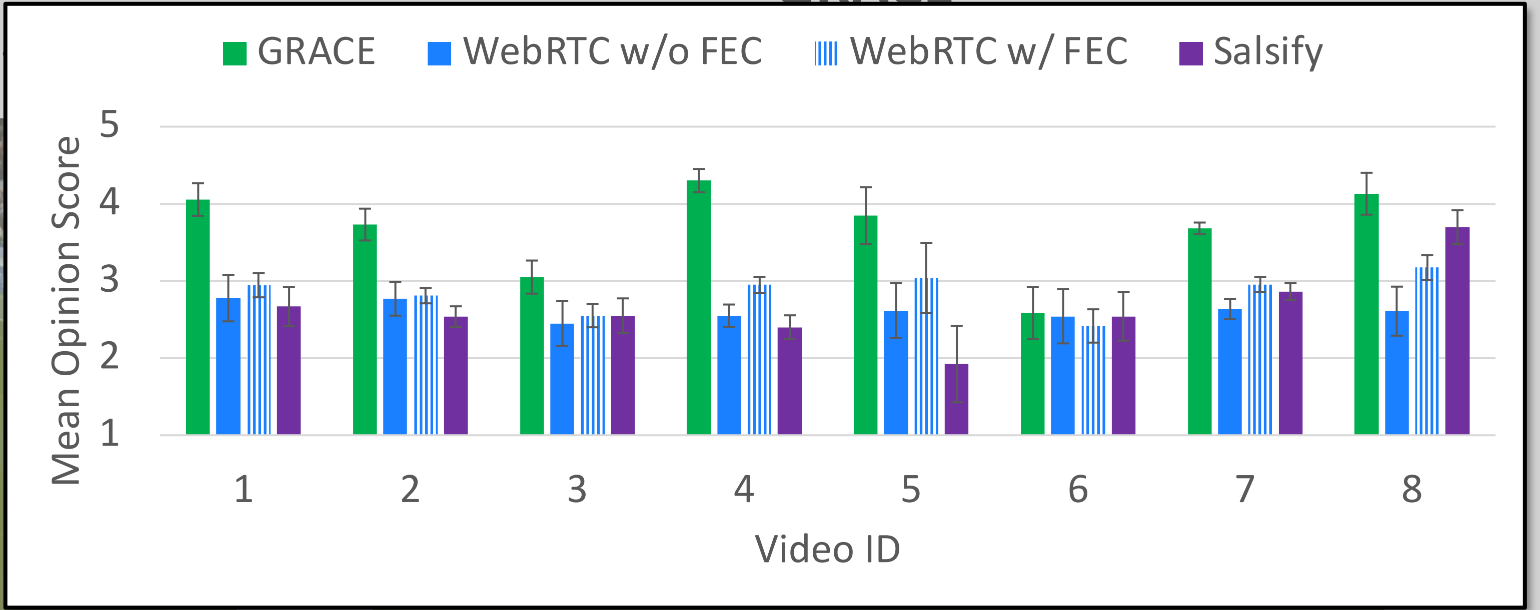
WebRTC w/ FEC

(high quality but minor gli



960 user ratings
on 8 sampled videos
from 240 Amazon MTurk workers

GRACE



Most users favor GRACE (smoother playback) over smooth quality with glitch

Conclusion

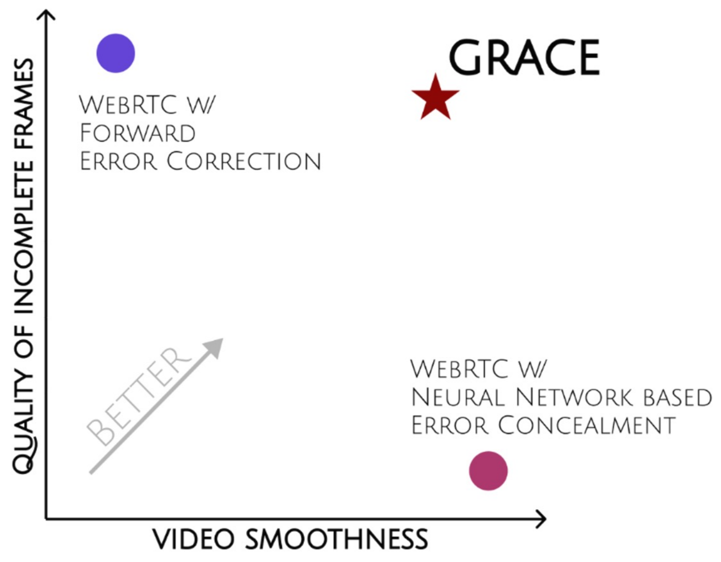
Objective	High quality, low latency playback under unpredictable network conditions
Insight	Improve loss resiliency by jointly training neural video encoder and decoder under different losses
GRACE	When packet loss happens, GRACE still streams video without freezing or significant quality drop

GRACE: The AI-Driven Future of Smooth Video Communication

A pioneering real-time video system that jointly optimizes a neural video codec's encoder and decoder to withstand diverse packet loss scenarios.

Paper

Code



Visit us at:

https://uchi-jcl.github.io/grace.html