GRACE: Loss-Resilient Real-Time Video through Neural Codecs 的复现研究

王奕辉

December 5, 2024

摘要

在实时视频通信中，由于严格的延迟要求，在高延迟网络上重传丢失的数据包并不可行。为了在不进行重传的情况下应对数据包丢失，主要采用两种策略——基于编码器的前向纠错（FEC）和基于解码器的错误隐藏。前者在传输前对数据进行带冗余的编码，但预先确定最佳冗余级别颇具挑战性。后者从部分接收的帧中重构视频，不过将一帧分割成独立编码的分区本质上会损害压缩效率，而且如果不对编码器做出调整，解码器无法有效恢复丢失的信息。

GRACE 是一个抗丢包实时视频系统，它通过一种新型神经视频编解码器，在各种丢包情况下保障用户的体验质量（QoE）。GRACE 增强丢包恢复能力的核心在于，它在一系列模拟丢包情况下对神经编码器和解码器进行联合训练。在无损场景下，GRACE 实现的视频质量与传统编解码器（如 H.265）相当。随着丢包率上升，GRACE 的质量下降更为平缓、不那么明显，始终优于其他抗丢包方案。通过对各种视频和真实网络轨迹进行广泛评估，与 FEC 相比，GRACE 将无法解码的帧数减少了 95%，卡顿时长减少了 90%，同时相较于错误隐藏方法显著提升了视频质量。在一项有 240 名众包参与者、获得 960 个主观评分的用户研究中，GRACE 的平均意见得分（MOS）比其他基线高出 38%。

本文还介绍了复现的相关细节。包括搭建与实验环境匹配的测试平台、数据集的准备以及训练与调试，最终成功复现出接近论文所述性能表现的 GRACE 模型训练代码。

关键词：神经视频编解码；实时视频通信；丢包恢复

1 引言

实时视频通信已成为人们日常生活不可或缺的一部分 [3]，涵盖在线会议 [4, 54]、云游戏 [37, 50]、交互式虚拟现实 [13, 51] 以及物联网应用 [49, 52]。为确保用户获得高质量的体验（QoE），实时视频应用必须防范数据包丢失问题。然而，由于严格的实时延迟要求 [19]，在高延迟网络上重传丢失的数据包并不可行。

抗丢包技术通常分为两类。第一类是编码器端的前向纠错（FEC），例如里德-所罗门码 [55]、喷泉码 [32, 33]，以及近期出现的流码 [2, 40]。FEC 在传输前将冗余信息融入数据。冗余率 $R\%$ 表示冗余数据相对于总数据大小的百分比，也就是说，最多可恢复 $R\%$ 的丢失数据，超过这个比例，视频就无法解码，导致视频质量急剧下降（图 1）。提高 R 值能防范更高的

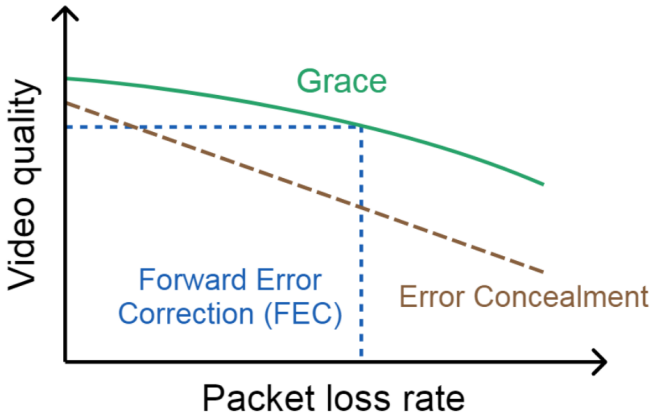

图 1. 不同抗丢包方案在不同丢包率情况下的视频质量示意图

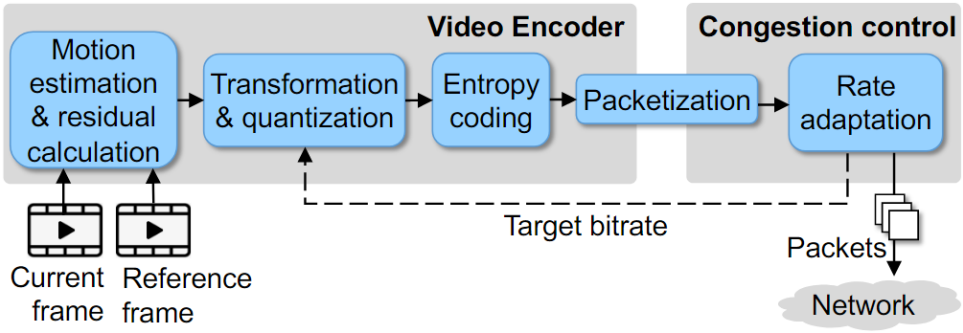

图 2. 视频帧编码的典型工作流程

丢包率，但也会带来更高的带宽开销，进而降低视频质量。因此，预先确定最优的 R 值在实际应用中面临挑战。第二类是解码器端的错误隐藏，它通过手工设计的启发式方法 [22,48,64] 或神经网络 [20,25,34,41,57]，重构受数据包丢失影响的视频帧部分内容。尽管如此，实现错误隐藏需要先将视频帧划分为可独立解码的单元（例如，条带 [53] 或瓦块 [23]），这会降低压缩效率。此外，由于编码器未针对抗丢包能力进行优化，仅靠解码器无法有效恢复丢失的信息。因此，如图 1所示，随着数据包丢失率的增加，视频质量往往会迅速恶化。

尽管现有技术在一定程度上缓解了丢包问题，但它们仍面临冗余带宽开销过大、恢复效果有限等挑战。为了解决这些问题，本研究提出了一种新型的抗丢包方法，通过联合优化神经编解码器，提高丢包恢复效果，从而为实时视频通信应用提供更高效的解决方案。

2 相关工作

图 2展示了视频帧编码的典型工作流程。由于丢包问题的出现，一直以来不少工作关注于将丢包问题对传输的影响降到最低。

2.1 前向纠错编码

前向纠错编码（FEC）在数据被发送到网络之前，于编码器处添加冗余信息。这也被称为抗错信道编码。示例包括里德-所罗门码、低密度奇偶校验码（LDPC）[33]、喷泉码和无速率码 [6,32]、流码 [2,40]，以及近期基于深度神经网络（DNN）的编码 [5,15]。还有分层和多级 FEC [46,47]，它们将 FEC 组织成多个层级，并用不同的冗余度保护每一层。FEC 还用于

保护可伸缩视频编码（SVC）中的帧元数据或基本层（也称为不等错误保护（UEP）[1,62]）。然而，为了选取合适的冗余率，需要预先估计将会丢失多少数据包。如果低估了丢包率，冗余将不足以恢复丢失的数据包。另一方面，添加过多的冗余会导致更高的带宽开销，进而降低视频质量。

2.2 后处理错误隐藏

后处理错误隐藏技术在解码器端重构丢失数据包中的缺失数据。这些方法通常由两部分组成。首先，当仅接收到数据包的一个子集时，编码数据包应能够被解码。这可以通过帧内模式宏块编码 [7]、条带交织 [18] 或灵活宏块排序 [24] 来实现。然而，这些方法往往会削弱编码器利用相邻宏块间冗余的能力，因为相邻宏块要么以帧内模式编码，要么被分割到不同数据包中（以棋盘格方式 [23,24]，或基于感兴趣区域检测 [45]）。因此，这些方法会损害压缩效率，导致编码后的帧大小膨胀 10%-50% [10,23,30,53]。

然后，解码器基于接收到的数据包重构丢失数据，使用经典的启发式方法（例如，H.264 中的运动矢量插值 [22,48,64] 和帧内块刷新 [23]），或是基于神经网络的图像修复方法 [20,25,34,41,57]。近期的工作 [25] 使用视觉 Transformer [11] 在帧解码前直接预测丢失数据包中的缺失位。然而，由于编码器对解码器的后处理缺乏了解，每个编码数据包包含的冗余和信息有限，难以辅助重构缺失的运动矢量或残差。结果，当数据包丢失时，重构过程只能被迫猜测缺失数据。即使是最新的技术，视频质量仍会出现显著下降（例如，在 20% 的丢包率下，峰值信噪比从 38dB 降至 25dB [36]）。

GRACE 采用了与前向纠错编码（FEC）和错误隐藏技术不同的方法。与仅依赖解码器端后处理的错误隐藏技术不同，GRACE（通过训练）联合优化（神经）编码器和解码器。与需要预先确定冗余率的 FEC 不同，GRACE 的编解码器在一系列丢包率下进行了优化。

2.3 其他方案

可伸缩视频编码（SVC）[8,42,43] 和精细粒度可伸缩性（FGS）[26,31] 旨在优化率失真（RD）权衡，即在不同接收比特率下单个编码比特流所实现的视频质量。SVC 将视频编码为多个质量层，并逐层发送数据。这对于点播视频 [8,28] 是可行的，但在实时视频中，一帧的所有数据包会一起发送以减少帧延迟。当基本层发生丢包时，它会阻碍任何更高层的解码。因此，SVC 很少用于提升单播实时视频的质量（尽管它在多播视频中用于为网络容量各异的用户提供服务 [42]）。

除了后处理错误隐藏技术，还有一些替代方案。例如，当发生丢包时，Salsify [14] 会回溯到一个更早但接收可靠的帧，而非用上一帧作为参考帧，这样解码器就可以安全跳过受丢包影响的帧，而不会影响后续帧。然而，与用上一帧作为参考帧相比，它编码相同质量的视频需要更多比特，例如，每隔一帧之间的 P 帧大小比连续两帧之间的 P 帧大 40%。长期参考帧（LTR）[61] 也有类似的局限性，只要接收到长期参考帧，每个 P 帧就可单独解码，无论其间是否有丢包。Voxel [38] 在编码器表明跳过某帧不会影响视频质量时，就会跳过受丢包影响的帧。这对于点播视频很有效，因为 B 帧可以被安全跳过，而且跳过一帧的影响会在几秒内止于下一个数据块。遗憾的是，这两种方法都不适用于实时视频。

最近，深度学习已被用于超分辨率 [21,44,58]、SVC [8,31] 以及基于卷积神经网络（CNNs）[20,34,41,57] 或 Transformers [12,25] 的后处理错误隐藏技术。超分辨率可以通过以较低比特

率发送视频，并在接收端提升视频质量来减少丢包。然而，它仍然需要重传机制来纠正因丢包而受损的帧。对于 SVC 和后处理错误隐藏技术，尽管使用了深度学习，这些方法固有的上述局限性依然存在。丢包恢复能力也在特定假设下被研究过，例如多路径的可用性 [9,35]、由路由器反馈驱动的早期重传 [65]、低延迟网络 [39] 以及视频游戏状态的可用性 [16,17,56]。

3 本文方法

3.1 本文方法概述

GRACE 是一个抗丢包的实时视频系统，在各种丢包情况下维持用户的体验质量（QoE）。其关键思路是，在一系列模拟丢包情况下联合优化编码器和解码器，能显著增强抗丢包能力。为实现这种联合优化，GRACE 巧妙地采用了神经视频编解码器（NVC）[29]，将神经网络融入传统视频编码器和解码器的核心组件。与 FEC 不同，GRACE 的 NVC 经过训练，可处理各种丢包情况，无需事先预测丢包率，也能避免在极高丢包率下出现无法解码的视频。与解码器端的错误隐藏不同，GRACE 对编码器和解码器进行联合训练，这样编码器就能学会在预估丢包的情况下，将每个像素的信息合理分配到多个输出元素中，便于解码器在实际丢包时重构帧。因此，GRACE 在不同丢包情况下质量下降更为平缓，且始终比以往解决方案提供更高的视频质量（图 1）。

GRACE 在设计时应对了三个系统挑战。首先，为确保容错能力，每个数据包必须可独立解码。现有解决方案通过将帧划分为可独立解码的单元来实现这一点。然而，这会引入大小开销，因为每个单元中的数据遵循不同分布，无法高效压缩。针对这一挑战，需要在训练 GRACE 的神经编码器时，使其输出值正则化，以符合相同分布，从而减少分包开销。在这种划分过程中，GRACE 还利用可逆随机映射 [27]，使其更适合 NVC。在丢包情况下训练 GRACE 时，模拟随机划分和丢包效率低下，且无法求导。因此，GRACE 直接对编码器输出应用随机归零，模拟丢包而无需实际丢弃数据包。

其次，数据包丢失可能导致编码器和解码器端参考帧之间出现差异，如果不保持同步，解码视频流的质量会持续下降。传统补救措施，如重传或发送新关键帧，无法无缝纠正这种不一致。GRACE 引入了一种创新协议，能在不妨碍视频解码的情况下，巧妙地“重新同步”编码器和解码器的状态。发生丢包时，解码器利用 GRACE 的抗丢包能力解码部分接收的数据包。同时，解码器将丢包细节告知编码器。这种反馈机制使编码器能够迅速调整其最近的参考帧，使其与解码器端的参考帧匹配，无需额外的数据传输。

第三，GRACE 必须能在从笔记本电脑到移动电话等各种设备上高效地实时编码和解码视频。然而，现有的 NVC 通常采用昂贵的神经网络，尤其是在运动估计和后处理方面。研究发现，通过缩小用于运动估计的图像输入尺寸并简化后处理，GRACE 能将编码和解码速度提高 4 倍，且对丢包恢复能力没有明显影响。

GRACE 的贡献可以总结为两点：第一，GRACE 是目前首次尝试在一系列丢包情况下联合训练神经视频编码器和解码器，旨在提升实时视频的抗丢包能力。与其他近期基于机器学习的实时视频系统 [59,60,63] 不同，这些系统使用基于机器学习的速率自适应来最小化丢包，而 GRACE 利用机器学习使视频编解码器本身具备抗丢包能力；第二，GRACE 构建了端到端视频系统，以应对与集成新 NVC 相关的实际挑战，开发了与分包、编解码器状态同步以及运行时效率相关的优化技术。

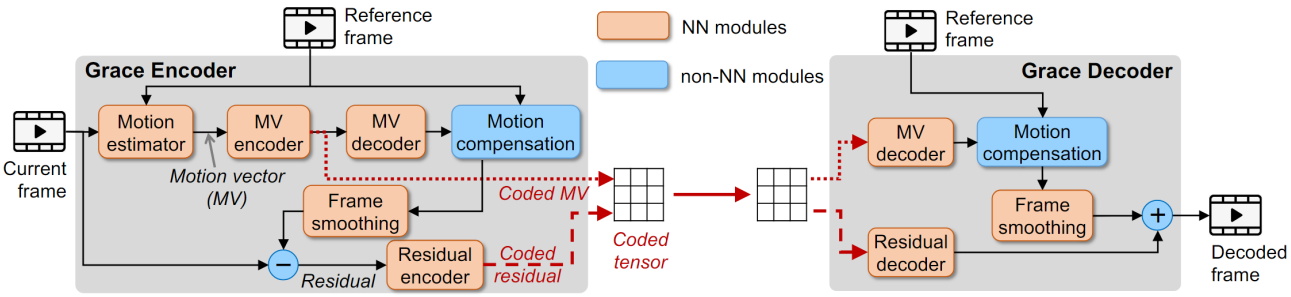

图 3. GRACE 的神经视频编解码器的工作流程

3.2 GRACE 的视频神经编解码及其训练

图 3描绘了 GRACE 编码器和解码器的工作流程（不包括熵编码和分包）。该编码器遵循与传统视频编码器类似的逻辑流程（图 2）。它首先使用一个神经网络（NN）来估计运动矢量（MV），并使用基于 NN 的 MV 编码器将其编码为一个量化张量。随后，该张量被解码回 MV，以匹配解码器接收到的 MV。接下来，编码器将这些 MV 应用于参考帧，生成一个运动补偿帧，并在计算它们之间的残差之前，使用一个帧平滑 NN 来提高其与当前帧的相似度。最后，基于 NN 的残差编码器将残差编码到另一个量化张量中。当解码器接收到编码后的 MV 张量和编码后的残差张量时，它们会经过与各自编码器联合训练的基于 NN 的 MV 解码器和残差解码器。

尽管 GRACE 的编码器和解码器包含多个步骤，但它们可被视为两个可微模型。我们用 f_ϕ（其中 ϕ 是其 NN 权重）表示编码器，用 g_θ（其中 θ 是其 NN 权重）表示解码器。编码器将帧 $\mathbf{x}$ 编码为编码张量 $\mathbf{y} = f_\phi(\mathbf{x})$，解码器将解码 $\mathbf{y}$ 为重构帧 $\hat{\mathbf{x}} = g_\theta(\mathbf{y})$。传统上，NVC 试图最小化以下损失函数：

$$\mathbb{E}_{\mathbf{x}}[D(g_\theta(\mathbf{y}), \mathbf{x}) + \alpha \cdot S(f_\phi(\mathbf{x}))], \text{where } \mathbf{y} = f_\phi(\mathbf{x}) \tag{1}$$

这里，$D(\hat{\mathbf{x}}, \mathbf{x})$ 是解码帧 $\hat{\mathbf{x}}$ 的像素级重构误差（例如 L2 范数），$S(\mathbf{y})$ 是 $\mathbf{y}$ 的熵编码数据大小，单位是每像素比特数（BPP）。参数 α 控制大小-质量的权衡：较高的会导致较小的帧大小，但重构帧 $\hat{\mathbf{x}}$ 的失真更大（即质量更差）。由于所有函数都是可微的，因此可以通过梯度下降联合训练 NN 权重和，以最小化损失函数1。

3.2.1 训练时模拟丢包

我们首先使用公式1预训练一个神经视频编解码器，然后通过以下方式引入模拟丢包来对其进行微调。GRACE 通过随机“屏蔽”——将编码器输出中的选定元素归零，来模拟丢包的影响，如图 4所示。形式上，GRACE 联合训练编码器和解码器神经网络，以最小化：

$$\mathbb{E}_{\mathbf{x}}[D(g_\theta(\mathbf{y}), \mathbf{x}) + \alpha \cdot S(f_\phi(\mathbf{x}))], \text{where } \mathbf{y} \sim P(\mathbf{y}|f_\phi(\mathbf{x})) \tag{2}$$

为了在的随机扰动下训练和的权重，这里使用强化学习技巧，通过蒙特卡罗采样来近似梯度。

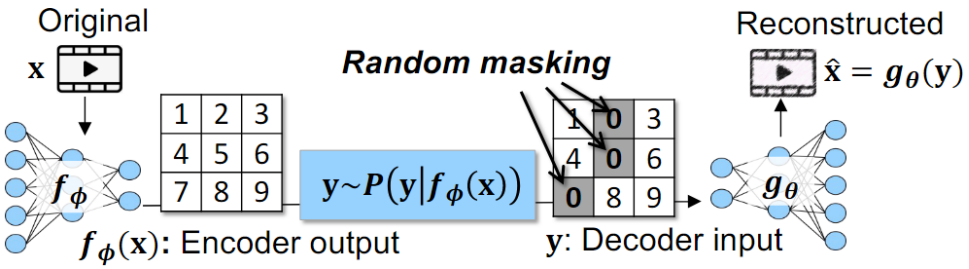

图 4. 随机掩码训练示意图

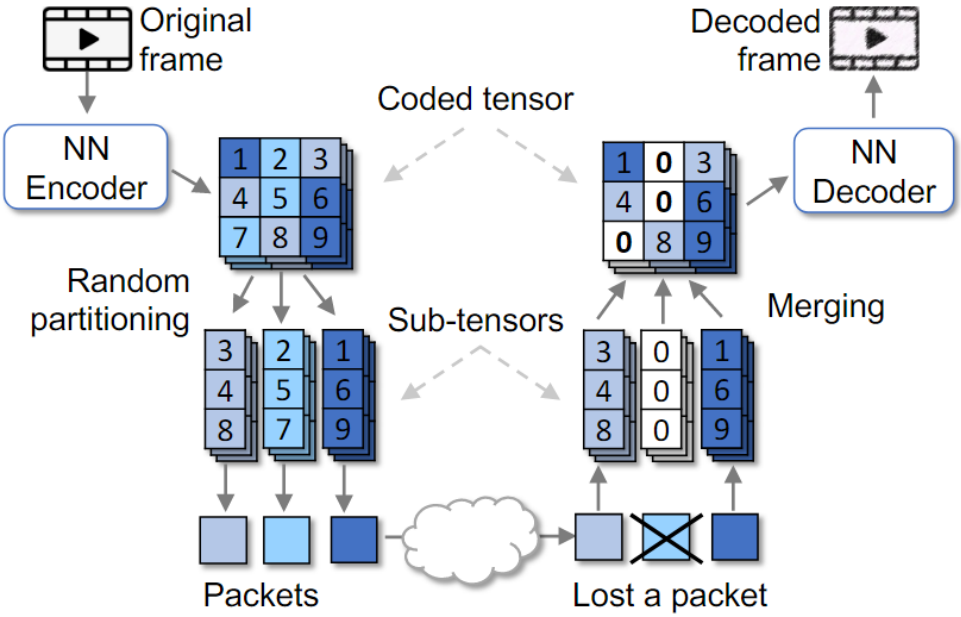

图 5. GRACE 的可逆随机化分组示意图

3.2.2 选择模拟丢包率

为了让 GRACE 的神经视频编解码器（NVC）能够应对各种各样的丢包率，在训练过程中模拟这类丢包情况至关重要。值得注意的是，即便只有一小部分训练样本引入了高丢包率（例如，超过 80%），通过实验观察到，在低丢包率下视频质量会大幅下降，而高丢包率下的质量提升却微乎其微。这种现象可能是因为编码器倾向于加入更多冗余信息来应对高丢包率，这对低丢包率下的视频质量产生了负面影响。因此，一个实际有效的丢包率分布既要涵盖低丢包率，也要涵盖高丢包率，并且要稍微侧重于低丢包率。

3.2.3 推理时分包

为了确保运行时实际数据包丢失的影响与随机屏蔽的效果一致，GRACE 采用了如图 5所示的可逆随机分包技术。GRACE 的发送方首先使用均匀随机映射将一帧的编码张量（包括编码后的运动矢量和编码后的残差）分割成多个子张量。我们使用可逆伪随机函数来生成该映射，以便接收方能够使用相同的随机种子正确恢复原始张量。

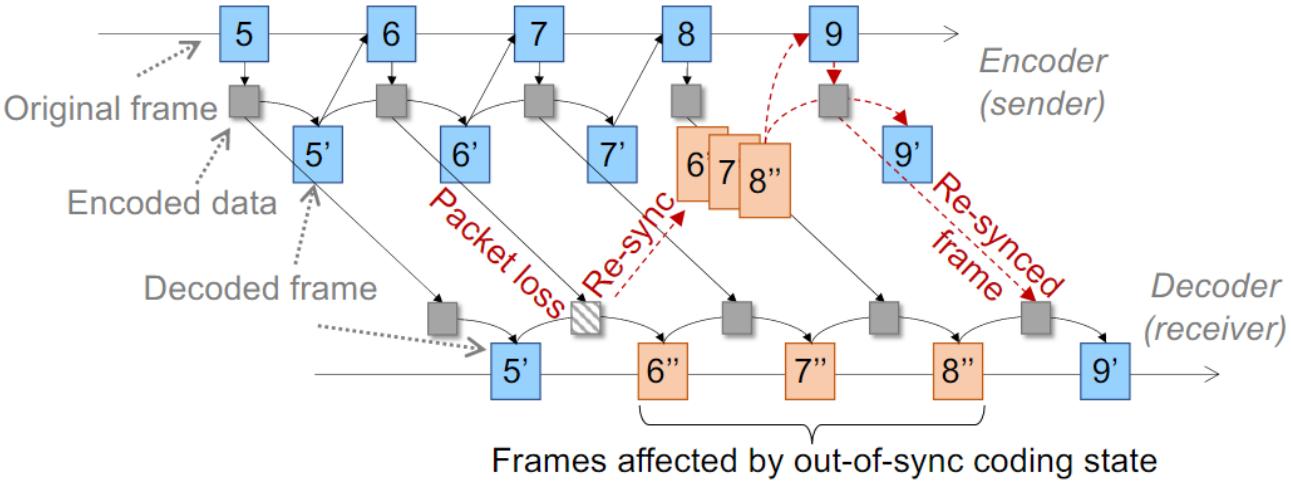

图 6. “重新同步”示意图

3.3 实时视频传输框架

3.3.1 编码器输出的熵编码

GRACE 使用可逆随机函数将编码器的输出分割为子张量，每个子张量对应一个单独的数据包。与 H.265 和 VP9 等经典编解码器类似，每个子张量都要通过算术（熵）编码进行无损压缩，形成比特流。算术编码器利用底层的符号分布来压缩张量中的值。我们没有依赖手工调整的启发式方法（例如 H.265 中的 CABAC），而是采用了 [29] 中描述的方法，与神经编码器和解码器一起训练一个分布估计器，以便更好地估计每个编码器输出的符号分布。由于 GRACE 独立解码单个数据包，数据包的符号分布必须作为数据包的一部分发送给解码器，这意味着随着数据包数量的增加，符号分布的大小开销也会增加。GRACE 通过采用更简单的符号分布来减少这种开销，该分布在每个数据包中只需占用更少的比特来存储。

3.3.2 实时视频流协议

GRACE 的编码器以固定帧率对新帧进行编码。当下一帧的任意数据包到达时，解码器会立即尝试解码当前帧。除非当前帧的所有数据包都丢失（这会触发重发该帧的请求），否则解码器将使用已接收的数据包来解码当前帧。我们把使用部分接收数据包解码出来的帧称为不完全帧。然而，尽管 GRACE 能以不错的质量解码不完全帧，但将这些不完全帧用作解码后续帧的参考图像，会导致编码器和解码器的状态“不同步”，也就是说，下一帧的解码所依据的参考图像，与编码时使用的参考图像不同。这种不一致会导致错误传播到后续帧，即便这些后续帧的所有数据包都无损到达。解决错误传播的一个初步方案是，在每一帧上都同步编码器和解码器。不过，每一帧的编码都会被阻塞，直至编码器知晓用于解码上一帧的是哪些数据包。这种同步延迟会让流水线式的编码、传输和实时解码变得不可行。

动态状态重同步的乐观编码：GRACE 采用两种策略来防止不同步状态阻塞编码器或解码器。首先，编码器乐观地假定所有数据包都将被接收，并据此对帧进行编码，利用 GRACE 解码器对少量帧丢包的容忍度。其次，当接收到不完全帧时，解码器在不停止解码新帧的情况下，以下列方式请求编码器动态重同步状态。收到重同步请求后，编码器从该不完全帧开始，仅使用解码器接收到的数据包子集（如重同步请求中所示），对最近的帧重新解码，以计

算出解码器所使用的最新参考帧。如图 6所示，如果编码器正要编码第 9 帧，并且得知第 6 帧是使用部分接收的数据包解码的，那么它会迅速重新解码从第 6 帧到第 8 帧。现在，第 8 帧与接收方的观测结果对齐，因此被用作编码第 9 帧的参考帧。状态重同步期间的帧重新解码（例如，图 6中的第 6 帧到第 8 帧）可能是一个潜在的速度瓶颈。幸运的是，编码器通过仅运行运动解码器和残差解码器，可以比常规解码过程更快地重新解码这些帧。原因有两点。第一，运动估计、运动编码和残差编码可以跳过，因为这些帧在编码器端已经被解码过一次，所以重新解码只需要估计由丢失数据包导致的增量变化。第二，虽然跳过帧平滑神经网络可能会影响最后一帧（例如，图 6中的第 9 帧）的压缩效率，但由于下一帧仍将被乐观编码，所以它只影响单帧。

GRACE 的乐观编码和动态状态重同步方法利用了 GRACE 神经视频编解码器（NVC）的一个关键优势——它不需要跳过或阻塞受丢包影响帧的解码处理；相反，在编码器和解码器的状态在几帧内不同步的情况下，它仍能以不错的质量解码这些帧，从而减少帧延迟。

4 复现细节

4.1 与已有开源代码对比

原文作者复现了神经编解码部分工作[1]，使用代码原本的设置并不能达到原文中理想的效果。此次复现工作将重点聚焦于神经编解码器的训练环节，通过精心设计实验流程、优化训练参数以及采用严谨的数据处理方法等一系列精细操作，力求使复现后的神经编解码器性能能够精准达到原文所呈现的水准。这不仅有助于深入理解原文核心技术的实现机制，而且为后续基于该技术的拓展研究筑牢根基。

原文开源的代码为达成提升编解码速度与效率的目标，选用了 BPG 编码方式。然而相关共享库文件出现损坏情况，致后续工作受阻。因此尝试自主编写代码来生成可用的库文件。经过尝试，所生成的库文件在压缩效率方面未能达到理想预期。最终改用更为稳定、编码效率也足够高的 jpeg 编解码技术。

4.2 实验环境搭建

本研究选用 Ubuntu 22.04 作为操作系统，搭配 PyTorch 1.13.1 深度学习框架，依托 NVIDIA Geforce RTX 3090 GPU 为神经编解码器训练提供算力支持。

至于数据集选取，从 Kinetics、Gaming、UVG、FVC 四个公共数据集中随机抽取 61 个视频，总时长 770 秒，单个视频时长介于 10 到 30 秒。这些视频来源独立于训练集，涵盖丰富多样的空间、时间内容复杂度，且囊括多种分辨率，这为精准评估 GRACE 在不同内容下的平均性能以及探究内容对其性能的影响创造了条件。

为实现与常用视频编解码标准 H.264、H.265 的对比，使用 FFmpeg v4.2.7 对这两种非神经编解码方法进行设置，以此作为实验基线，确保实验结果具备广泛参考性。

[1]https://github.com/UChi-JCL/Grace

4.3 实验设置

预训练 DVC 模型：用 90k Vimeo 数据集微调 DVC 模型来获取 GRACE 的抗丢包模型，训练时，批次大小设为 4，采用 Adam 优化器，学习率为 10^{-4}，学习率衰减为 0.1。

丢包率模拟：有 80% 的概率将丢包率设为 0%；有 20% 的概率从10%, 20%, 30%, 40%, 50%, 60% 中随机选取丢包率。

光流估计网络：使用 ME SPYNet，一种用于光流估计的深度学习模型。其核心组件是 MEBasic 模块，每个模块包含五个卷积层，逐步提取输入图像的特征。在前向传播过程中，输入的两张图像首先经过平均池化降采样，逐层处理以提取多尺度特征。然后，模型通过逐层上采样和残差学习，逐步生成精细的光流场。最终，输出的光流场用于描述两张图像之间的运动信息。

运动向量编码网络：使用 Analysis mv Net，一种用于运动压缩的深度学习模型。该网络由八个卷积层组成，逐步提取输入的运动信息。在前向传播过程中，输入的运动数据首先经过一系列卷积层和 LeakyReLU 激活函数，逐层提取特征。每个卷积层的权重初始化采用 Xavier 正态分布，偏置初始化为 0.01。最终，输出的特征图用于表示压缩后的运动信息。

运动向量解码网络：使用 Synthesis mv Net，一种用于运动合成的深度学习模型。该网络由八个卷积层和转置卷积层组成，逐步重建输入的运动信息。在前向传播过程中，输入的运动特征首先经过一系列卷积层和 LeakyReLU 激活函数，逐层提取特征。然后，模型通过转置卷积层逐步上采样，恢复空间分辨率。每个卷积层的权重初始化采用 Xavier 正态分布，偏置初始化为 0.01。最终，输出的特征图用于表示合成后的运动信息。

残差编码网络：使用 Analysis Net，一个用于压缩残差的神经网络模型。该网络由多个卷积层（和相应的 GDN（Generalized Divisive Normalization）层组成。GDN 层是一种归一化层，用于对特征图进行归一化处理，以提高模型的表达能力和训练稳定性。在每个卷积层后，GDN 层对卷积结果进行归一化处理。最后，conv4 层输出压缩后的特征图。该网络的设计旨在有效地压缩输入图像的残差信息。

残差解码网络：使用 Synthesis Net，一个用于解码残差的神经网络模块。将经过编码和压缩的残差信息还原为原始图像的残差部分。该网络由多个转置卷积层和逆 GDN（Inverse GDN）层组成。在每个反卷积层的权重初始化中，使用了 Xavier 正态分布初始化方法，以促进训练过程的稳定性。每个反卷积层的偏置项初始化为 0.01，以避免在训练初期出现偏置项为零的情况。逆 GDN 层用于对特征图进行逆 GDN 变换，恢复特征的非线性关系。

残差先验编码网络：使用 Analysis Prior Net，一个用于压缩残差先验的神经网络。该网络包含三个卷积层。权重初始化采用 Xavier 正态分布，偏置初始化为 0.01。卷积层之间使用 ReLU 激活函数。

残差先验解码网络：使用 Synthesis Prior Net，一个用于压缩残差先验的神经网络。该网络包含三个转置卷积层。权重初始化采用 Xavier 正态分布，偏置初始化为 0.01。转置卷积层之间使用 ReLU 激活函数。

5 实验结果分析

图 7展示了复现得出的 SSIM 和丢失率的变化关系。容易看出在同一丢失率下，GRACE 在绝大部分情况都优于传统编解码方法。

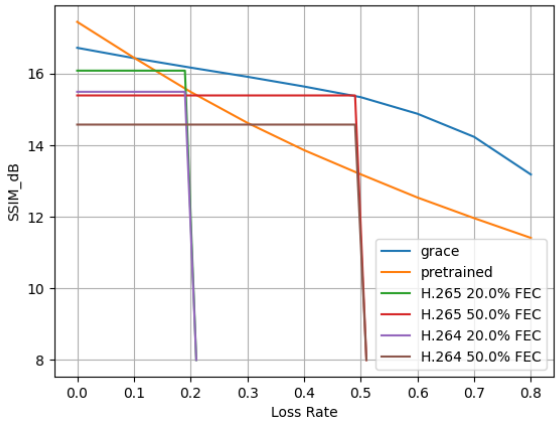

图 7. SSIM-丢失率对应变化 复现结果

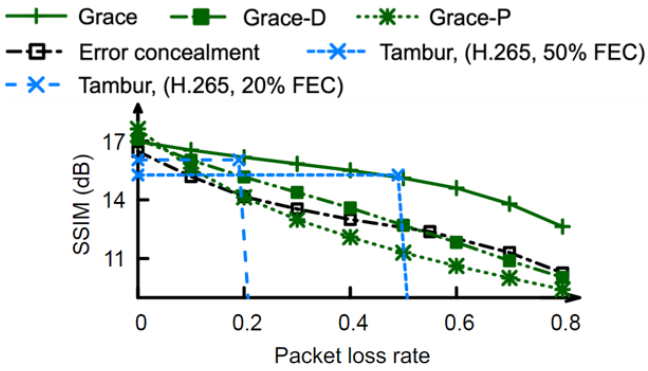

图 8. SSIM-丢失率对应变化 原文结果

图 8展示了原文中 SSIM 和丢失率的变化关系，其中 Grace-D 表示训练时冻结编码器权重的变体；Grace-P 表示不模拟丢失的变体。容易看出复现工作达到了论文展示的效果。

图 9展示了复现得出的 SSIM 和帧大小的变化关系。容易看出在帧大小较大时 GRACE 的帧质量更好。

图 10展示了原文中的 SSIM 和帧大小的变化关系。容易看出复现工作基本达到了论文展示的效果，对于帧大小较小时的差异，推测可能原因为受编码方式改变的影响。

6 总结与展望

本次 GRACE 复现工作取得了显著进展。GRACE 创新性地在多样的丢包情境下联动训练神经编码器与解码器，强化了抗丢包能力，为实时视频通信的稳定传输提供了有力保障。

复现过程带来了很多优点，一方面提升代码调试与优化技能；另一方面，在神经编解码领域的精研迈出了第一步，通过反复调试参数、优化算法结构，成功使复现性能逼近原文水准，不仅带来了学习的正反馈，也为后续的进一步学习打下基础。

然而，不可忽视的是，工作仍存在短板，尚未复现实时视频流部分，这使得复现成果暂时无法完整覆盖实时视频通信全流程，限制了其应用的广泛性。

展望未来，前进方向清晰明了。深入剖析其技术架构，结合当下人工智能算法革新趋势，探索更高效的联合训练模式，力求让复现成果全面赋能实时视频通信，切实优化用户的使用

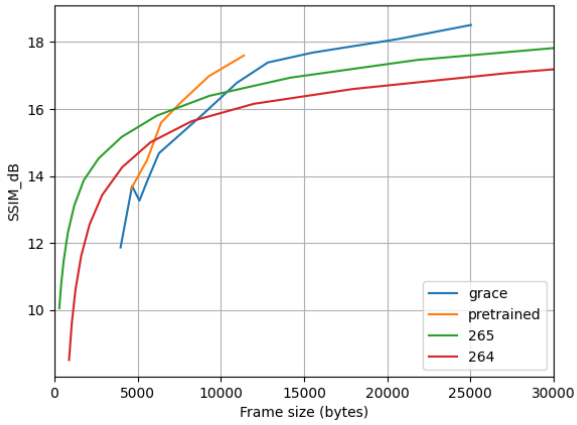

图 9. SSIM-帧大小对应变化 复现结果

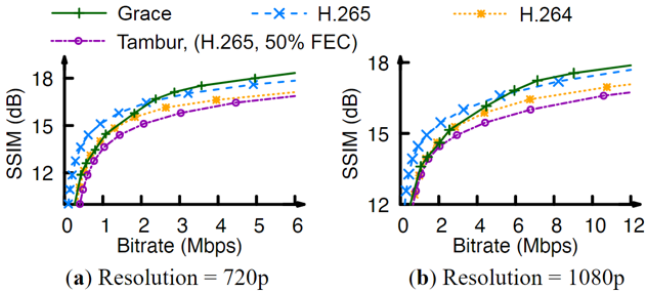

图 10. SSIM-帧大小对应变化 原文结果

体验。另外，GRACE 更侧重于实时视频传输的应用场景，或许在点播视频传输的应用场景，也会有更友好、更广阔的钻研体验。

参考文献

[1] Asma Ben Abdallah, Amin Zribi, Ali Dziri, Fethi Tlili, and Michel Terré. H. 264/avc video transmission over uwb av phy ieee 802.15. 3c using uep and adaptive modulation techniques. In *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–6. IEEE, 2019.

[2] Ahmed Badr, Ashish Khisti, Wai-tian Tan, Xiaoqing Zhu, and John Apostolopoulos. Fec for voip using dual-delay streaming codes. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[3] Niklas Blum, Serge Lachapelle, and Harald Alvestrand. Webrtc: Real-time communication for the open web platform. *Communications of the ACM*, 64(8):50–54, 2021.

[4] Bringing Zoom's end-to-end optimizations to WebRTC. https://blog.livekit.io/livekit-one-dot-zero/.

[5] Fabrizio Carpi, Christian Häger, Marco Martalò, Riccardo Raheli, and Henry D Pfister. Reinforcement learning for channel coding: Learned bit-flipping decoding. In *2019 57th*

Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 922–929. IEEE, 2019.

[6] Jeff Castura and Yongyi Mao. Rateless coding over fading channels. *IEEE communications letters*, 10(1):46–48, 2006.

[7] Wen-Jeng Chu and Jin-Jang Leou. Detection and concealment of transmission errors in h. 261 images. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(1):74–84, 1998.

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

[9] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. Converge: Qoe-driven multipath video conferencing over webrtc. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 637–653, 2023.

[10] Yves Dhondt and Peter Lambert. Flexible macroblock ordering: an error resilience tool in h. 264/avc. In *5th FTW PhD Symposium*. Ghent University. Faculty of Engineering, 2004.

[11] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[12] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[13] Features of WebRTC VR Streaming. https://flashphoner.com/features-of-webrtc-vr-streaming/.

[14] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S Wahby, and Keith Winstein. 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)*, pages 267–282, 2018.

[15] Tobias Gruber, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. On deep learning-based channel decoding. In *2017 51st annual conference on information sciences and systems (CISS)*, pages 1–6. IEEE, 2017.

[16] Zhaoyuan He, Yifan Yang, Shuozhe Li, Diyuan Dai, and Lili Qiu. Neural video recovery for cloud gaming. *arXiv preprint arXiv:2307.07847*, 2023.

[17] Zhaoyuan He, Yifan Yang, Lili Qiu, and Kyoungjun Park. Real-time neural video recovery and enhancement on mobile devices. *arXiv preprint arXiv:2307.12152*, 2023.

[18] Ismaeil Ismaeil, Shahram Shirani, Faouzi Kossentini, and Rabab Ward. An efficient, similarity-based error concealment method for block-based coded images. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 388–391. IEEE, 2000.

[19] ITU-T. Recommendation g.114, one-way transmission time. Technical report, Telecommunication Standardization Sector of ITU, 2003.

[20] Jaeyeon Kang, Seoung Wug Oh, and Seon Joo Kim. Error compensation framework for flow-guided video inpainting. In *European conference on computer vision*, pages 375–390. Springer, 2022.

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

[22] Vineeth Shetty Kolkeri. Error concealment techniques in h. 264/avc, for video transmission over wireless networks. Master's thesis, The University of Texas at Arlington, 2009.

[23] Sunil Kumar, Liyang Xu, Mrinal K Mandal, and Sethuraman Panchanathan. Error resiliency schemes in h. 264/avc standard. *Journal of Visual Communication and Image Representation*, 17(2):425–450, 2006.

[24] Peter Lambert, Wesley De Neve, Yves Dhondt, and Rik Van de Walle. Flexible macroblock ordering in h. 264/avc. *Journal of Visual Communication and Image Representation*, 17(2):358–375, 2006.

[25] Tianhong Li, Vibhaalakshmi Sivaraman, Pantea Karimi, Lijie Fan, Mohammad Alizadeh, and Dina Katabi. Reparo: Loss-resilient generative codec for video conferencing. *arXiv preprint arXiv:2305.14135*, 2023.

[26] Weiping Li. Overview of fine granularity scalability in mpeg-4 video standard. *IEEE Transactions on circuits and systems for video technology*, 11(3):301–317, 2001.

[27] Linear Congruential Generator. https://en.wikipedia.org/wiki/Linear_congruential_generator.

[28] Yunzhuo Liu, Bo Jiang, Tian Guo, Ramesh K Sitaraman, Don Towsley, and Xinbing Wang. Grad: Learning for overhead-aware adaptive video streaming with scalable video coding. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 349–357, 2020.

[29] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11006–11015, 2019.

[30] Rong Luo and Bin Chen. A hierarchical scheme of flexible macroblock ordering for roi based h. 264/avc video coding. In *2008 10th International Conference on Advanced Communication Technology*, volume 3, pages 1579–1582. IEEE, 2008.

[31] Yi Ma, Yongqi Zhai, and Ronggang Wang. Deepfgs: Fine-grained scalable coding for learned image compression. *arXiv preprint arXiv:2201.01173*, 2022.

[32] David JC MacKay. Fountain codes. *IEE Proceedings-Communications*, 152(6):1062–1068, 2005.

[33] David JC MacKay and Radford M Neal. Near shannon limit performance of low density parity check codes. *Electronics letters*, 33(6):457–458, 1997.

[34] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[35] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 193–206, 2022.

[36] Cholman Nam, Changgon Chu, Taeguk Kim, and Sokmin Han. A novel motion recovery using temporal and spatial correlation for a fast temporal error concealment over h. 264 video sequences. *Multimedia Tools and Applications*, 79(1):1221–1240, 2020.

[37] Open Source Cloud Gaming with WebRTC. https://webrtchacks.com/open-source-cloud-gaming-with-webrtc/.

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

[39] Devdeep Ray, Connor Smith, Teng Wei, David Chu, and Srinivasan Seshan. Sqp: Congestion control for low-latency interactive video streaming. *arXiv preprint arXiv:2207.11857*, 2022.

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

[41] Arun Sankisa, Arjun Punjabi, and Aggelos K Katsaggelos. Video error concealment using deep neural networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 380–384. IEEE, 2018.

[42] Thomas Schierl, Thomas Stockhammer, and Thomas Wiegand. Mobile video transmission using scalable video coding. *IEEE transactions on circuits and systems for video technology*, 17(9):1204–1217, 2007.

[43] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007.

[44] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. Gemino: Practical and robust neural compression for video conferencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 569–590, 2024.

[45] Keyu Tan and Alan Pearmain. A new error resilience scheme based on fmo and error concealment in h. 264/avc. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1057–1060. IEEE, 2011.

[46] Wai-tian Tan and Avideh Zakhor. Multicast transmission of scalable video using receiver-driven hierarchical fec. In *Packet Video Workshop*, volume 99, 1999.

[47] Wai-Tian Tan and Avideh Zakhor. Video multicast using layered fec and scalable compression. *IEEE Transactions on circuits and systems for video technology*, 11(3):373–386, 2001.

[48] Yi Wang, Xiaoqiang Guo, Feng Ye, Aidong Men, and Bo Yang. A novel temporal error concealment framework in h. 264/avc. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2013.

[49] WebRTC and IoT Applications. https://rtcweb.in/webrtc-and-iot-applications/.

[50] WebRTC Cloud Gaming: Unboxing Stadia. https://webrtc.ventures/2021/02/webrtc-cloud-gaming-unboxing-stadia/.

[51] WebRTC: Enabling Collaboration Augmented Reality App. https://arvrjourney.com/webrtc-enabling-collaboration-cebdd4c9ce06?gi=e19b1c0f65c0.

[52] WebRTC in IoT: What is the Intersection Point? https://mobidev.biz/blog/webrtc-real-time-communication-for-the-internet-of-things.

[53] Stephan Wenger and Michael Horowitz. Scattered slices: a new error resilience tool for h. 26l. *JVT-B027*, 2, 2002.

[54] What powers Google Meet and Microsoft Teams? WebRTC Demystified. https://levelup.gitconnected.com/what-powers-google-meet-and-microsoft-teams-webrtc-demystified-step-by-step-tutorial-e0cb422010f7.

[55] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[56] Jiangkai Wu, Yu Guan, Qi Mao, Yong Cui, Zongming Guo, and Xinggong Zhang. Zgaming: Zero-latency 3d cloud gaming by image prediction. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 710–723, 2023.

[57] Chongyang Xiang, Jiajun Xu, Chuan Yan, Qiang Peng, and Xiao Wu. Generative adversarial networks based error concealment for low resolution video. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1827–1831. IEEE, 2019.

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

[59] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. Loki: improving long tail performance of learning-based real-time video adaptation by fusing rule-based models. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 775–788, 2021.

[60] Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, and Xiaojiang Chen. Onrl: improving mobile video telephony via online reinforcement learning. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[61] Qing Zhang and Guizhong Liu. Error resilient coding of h. 264 using intact long-term reference frames. 2008.

[62] Zenghua Zhao and Shubing Long. Rd-based adaptive uep for h. 264 video transmission in wireless networks. In *2010 International Conference on Multimedia Information Networking and Security*, pages 72–76. IEEE, 2010.

[63] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–16, 2019.

[64] Jie Zhou, Bo Yan, and Hamid Gharavi. Efficient motion vector interpolation for error concealment of h. 264/avc. *IEEE Transactions on Broadcasting*, 57(1):75–80, 2010.

[65] Xutong Zuo, Yong Cui, Xin Wang, and Jiayu Yang. Deadline-aware multipath transmission for streaming blocks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 2178–2187. IEEE, 2022.