arXiv:2303.02959v1 [cs.CV] 6 Mar 2023

Butterfly：神经视频压缩的多参考帧特征传播机制

Feng Wang*、Haihang Ruan†, ⋆、Fei Xiong*、Jiayu Yang*、Litian Li* 和Ronggang Wang*✉

*北京大学深圳研究生院，中国

†中国科学院上海微系统与信息技术研究所，上海，中国*中国科学院大学，北京，中国 wangfeng@stu.pku.edu.cn & rgwang@pkusz.edu.cn

摘要

使用更多参考帧可以显著提升神经视频压缩的压缩效率。然而，在低延迟场景中，大多数现有神经视频压缩框架通常使用前一帧作为参考帧。少数使用前面多帧作为参考帧的框架，也仅采用简单的多参考帧传播机制。本文提出了一种更合理的神经视频压缩多参考帧传播机制，称为Butterfly多参考帧传播机制（Butterfly），能够更有效地融合多参考帧的特征。借此，我们可以为上下文编码模块生成更准确的时序上下文条件先验。此外，当已解码帧的数量未达到所需参考帧数量时，我们复制最近的参考帧来满足要求，这比复制最远的参考帧效果更好。实验结果表明，我们的方法能够显著优于此前的最先进水平；在采用相同压缩配置的情况下，与我们的单参考帧模型相比，我们的神经编解码器在HEVC D类数据集上可实现7.6%的码率节省。

1 引言

如今，视频占据了互联网流量的大部分。互联网上的视频数据流量增长速度超过了网络带宽。因此，有必要研究和开发视频压缩技术，以便在带宽有限的情况下高效地通过互联网传输视频数据。在过去几十年中，人们开发了许多传统的视频压缩标准，例如 H.264/AVC [1],H.265/HEVC [2], H.266/VVC [3], 等。它们通常采用混合编码框架，包括帧内/帧间预测、变换器、量化、熵编码和环路滤波器。然而，这些传统的视频压缩标准依赖于手工设计模块。尽管每个模块都经过精心设计，但它们难以联合优化。因此，为了获得更高的压缩效率，有必要对视频压缩框架进行端到端的联合优化。

近年来，随着深度学习的兴起与发展，越来越多基于深度学习的神经视频压缩（NVC）方法 [4, 5, 6, 7, 8, 9, 10]

✉: 通讯作者邮箱：rgwang@pkusz.edu.cn。

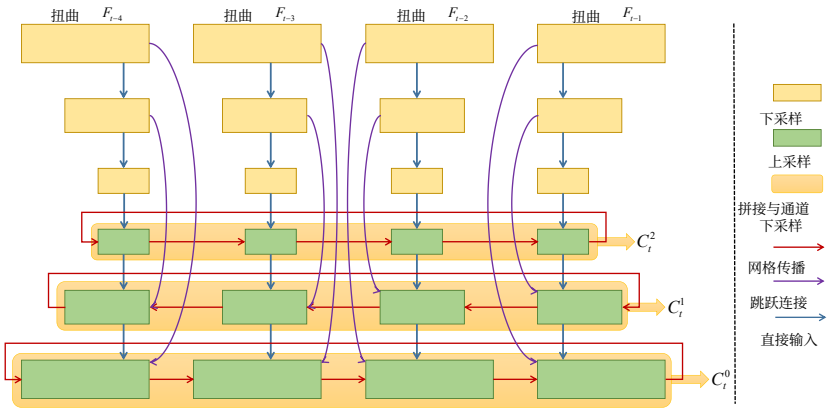

图1：我们提出的蝴蝶式多参考帧特征传播机制（Butterfly）模块的结构。

相继被提出。根据视频压缩的应用场景，现有的神经视频压缩研究可分为两类。[1] 第一类是低延迟（LD），另一类是随机访问（RA）。本文聚焦于低延迟模式，该模式通过单向预测来减少时间冗余

在低延迟模式下，已有一系列成功的神经视频编解码器 [4, 5, 6, 7, 8, 9, 10, 11]，并取得了非常出色的压缩效率。它们大多遵循相似的流程。首先，利用运动估计模块获取前一帧与当前帧之间的运动信息，然后通过运动矢量（MV）编码器-解码器对其进行压缩。获取运动信息后，运动补偿模块会对前一帧进行运动补偿。随后，以补偿结果为先验，对当前帧进行残差编码或条件编码。

然而，它们大多仅使用一帧前一帧作为参考来预测当前帧。有些工作 [8, 10] 使用多个前一帧作为参考，并取得了更好的压缩效率。但是，它们融合多参考帧的方法较为简单。例如，MLVC [10]只是直接拼接四个前一帧，并未进行更有针对性的设计。这会导致在融合这些多个前一帧时，帧中的一些关键信息被抹去。

本文针对低延迟（LD）模式下的神经视频压缩（NVC），设计了一种蝴蝶式多参考帧特征传播机制（Butterfly），以实现更高的压缩效率。如图 1 所示，Butterfly 能够更有效地融合来自不同先前参考帧的特征，并且不会导致融合过程中某些关键信息丢失。因此，我们可以为上下文编码模块和帧生成模块生成更准确的时序上下文条件先验。此外，在压缩前 **n-1** 个 P 帧时（假设先前参考帧的数量为 **n**），由于其参考帧少于 **n** 个，

[1]实际上，传统的 VVC 视频压缩标准包含三种编码结构：全帧内模式（AI）、低延迟模式（LD）和随机访问模式（RA）。然而，在全帧内模式下，视频序列的每一帧都通过帧内编码（图像压缩）进行编码，并未利用视频序列中的时间依赖性。因此，神经视频压缩方法不讨论全帧内模式。

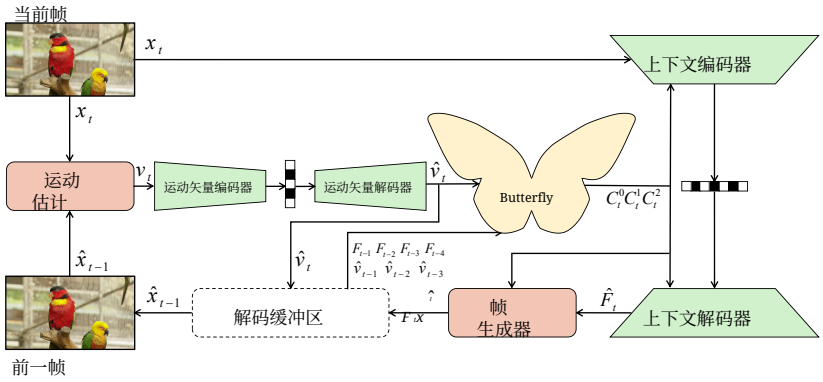

图2：我们提出的多参考帧神经视频压缩方案概览。运动矢量编码器-解码器和上下文编码器-解码器均采用超先验结构 [12]。

有必要复制某个参考帧，以满足所需的参考帧数量。我们通过不等式证明，在 [10] 我们的模型中，复制最近的参考帧优于复制较远的参考帧。实验结果表明，我们的方法能够显著优于此前的最先进水平；与采用相同压缩配置的基础单参考帧模型 [5] 相比，我们的方法在 HEVC D类数据集上可实现 -7.6% 的码率节省。本文的主要贡献总结如下：

- 我们提出了一种用于低延迟模式下神经视频压缩的蝴蝶式多参考帧特征传播机制（Butterfly）。借助该机制，网络不仅能够防止帧中的某些关键信息在融合多个参考帧时被抹除，还能聚合来自不同尺度和不同时间位置的信息。

- 我们通过不等式证明，当解码帧数量未达到所需的参考帧数量时，复制最近的参考帧（近帧复制）效果更好。我们还通过对比实验验证了这一点。

- 实验结果表明，我们的方法显著优于此前的最先进水平（SOTA），与作为基准的单参考帧模型相比，也能实现更高的压缩效率。

2 蝴蝶式多参考帧特征传播机制

我们提出利用蝴蝶式多参考帧特征传播机制，融合先前多个参考帧的特征。随后，将融合结果作为先验，对当前帧进行条件编码。本节首先在第2.1小节详细介绍我们的神经视频压缩框架，然后在第2.2小节描述蝴蝶式多参考帧特征传播机制。

2.1 概述

我们的网络采用基于条件编码的 TCM 框架 [5]，并重新设计了参考帧特征融合模块。图2概述了我们的

广义神经视频压缩模型，该模型包含以下五个主要子网络。

运动估计。 视频中相邻帧之间的相关性非常强。通过运动估计可以减少帧间的时间冗余，从而大幅减少视频传输所需的比特数。在我们提出的神经视频压缩方法中，综合考虑整个模型的计算复杂度和模型大小，我们使用 SPyNet [13] 来估计运动信息。

运动矢量编码器-解码器。 将估计出的运动信息直接通过信道传输会消耗大量比特，因此有必要对运动信息进行压缩。我们采用一种具有超先验结构的自动编码器式网络 [12] 作为 [5]。解码后的 $\hat{}v_t$ 将存储到解码缓冲区（DB）中。

蝴蝶式多参考帧特征传播。 现有大多数方法都采用前一帧作为参考帧来压缩当前帧。少数使用前多个帧作为参考帧的方法 [8, 10] 也仅采用简单的多参考帧传播。这些方法会限制神经视频压缩方法的压缩性能。为了充分利用多参考帧的信息，我们提出了一种蝴蝶式多参考帧特征传播机制，用于融合多参考帧的信息，并生成条件先验，以指导后续的上下文编码器-解码器和帧生成器对当前帧进行压缩与重建 x_t。具体而言，在我们的网络中，扭曲后的多参考帧被输入 Butterfly，以获取多尺度特征。在下采样阶段，多参考帧分别进行下采样，以获得不同尺度的特征。在上采样阶段，我们利用网格传播融合不同尺度的特征。通过这种方式，网络不仅能够防止帧中的一些关键信息在融合多参考帧的过程中被抹除，还能够通过聚合来自不同的时空位置和不同尺度的信息，生成更加准确的时序上下文条件先验。具体细节见第2.2小节。

上下文编码器-解码器。 遵循 DCVC [4], ，我们采用基于条件编码的方法来压缩当前帧 x_t。为了提取 x_t 更紧凑的潜在表示，我们将时序上下文条件先验拼接到上下文编码器-解码器中，以减少 x_t 的时间冗余和空间冗余$\hat{}$。上下文编码器-解码器的输出是当前帧的高分辨率特征 F_t 。我们将其输入帧生成器，以生成重建帧 $\hat{}x_t$。

帧生成器。 它用于生成高质量的重建帧 $\hat{}x_t$。我们使用两个普通残差块构建帧生成器。这样既能生成高质量的重建帧，又能使整个神经视频压缩模型的复杂度和参数量保持适中。帧生成器的输入是 $\hat{}$高分辨率特征 F_t 和最大尺度的时序上下文条件先验$\bar{C}_t^0$。输出是重建帧 $\hat{}x_t$ 和特征 F_t，它们会存储到解码缓冲区（DB）中，以帮助压缩后续帧。

2.2 Butterfly

使用更多参考帧可以显著提升压缩效率。然而，简单拼接不同参考帧的信息会丢失

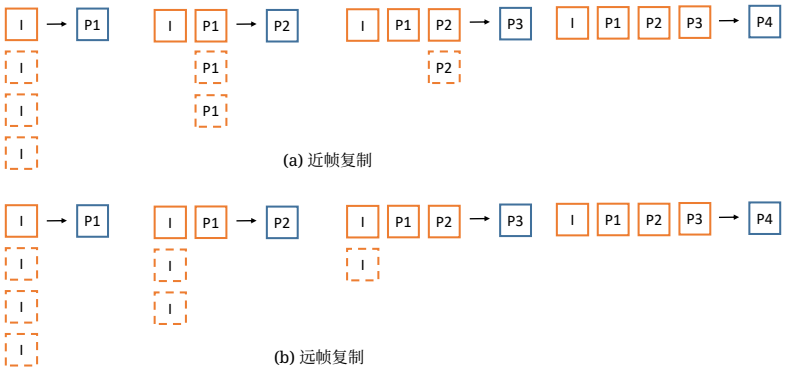

图 3：我们提出的近帧复制方法和 MLVC [10] 采用的远帧复制方法。橙色帧表示已解码的重建帧，蓝色帧表示待解码帧，虚线框表示复制帧。

一些关键信息。例如，由于遮挡，某个物体可能只出现在多参考帧中的一帧里。Concat 只是在通道维度上拼接不同帧的信息（也就是说，每帧的重要性都相同），因此在融合过程中，这些信息会被其他帧的信息消除。因此，探索一种更有效的多参考帧特征传播机制，可以充分利用多参考帧提供的时间相关性信息。

为了融合多参考帧的特征，同时避免丢失某一帧中的关键信息，我们设计了一种 Butterfly 多参考帧特征传播机制，如图 1 所示，该机制包含两个主要过程：上采样和下采样。在下采样阶段，对多个参考帧分别进行下采样，以获取不同尺度的特征，从而确保部分关键信息不会丢失。在上采样阶段，我们使用网格传播来融合不同尺度和不同帧的特征。这样可以聚合来自不同时空位置和不同尺度的信息，并提高网络对遮挡区域和细粒度区域的鲁棒性。

网格状设计常见于多种视觉任务中，例如目标检测、语义分割和视频超分辨率 [14]。将这种设计用于神经视频压缩后，模块可以适应不同分辨率，从不同参考帧中提取细节和粗略信息。具体而言，不同尺度的特征会在不同的先前参考帧之间交替传播。通过传播，可以“重新访问”不同帧中的信息，并将其用于特征融合，从而提升特征表达能力。

3 近帧复制

当解码帧数量未达到所需的参考帧数量时，需要复制一帧已解码的帧。为此，我们可以复制最近的参考帧，也可以复制最远的参考帧。

我们证明，当参考帧的数量无法满足所需数量时，复制最近的参考帧（近帧复制）会更好。

由于我们的神经视频压缩方法属于有损压缩，在一个帧内周期中会产生误差累积，即随着压缩过程的进行，时间维度上每帧丢失的信息量逐渐增加。为便于分析，我们假设每帧相对于前一帧的信息损失百分比增量为固定值 $\boldsymbol{\alpha}$。即

$$L_{t+1} = (1+\boldsymbol{\alpha}) * L_t, \tag{1}$$

其中， L_t 和 L_{t+1} 是在时间步 t 和（$t+1$）时帧中丢失的信息量。

视频是沿时间轴方向排列的帧图像序列，相邻帧之间的相关性很强。为便于分析，我们假设相邻帧之间的相关性也是一个固定值 $\boldsymbol{\beta}$。因此可得

$$I_{t+1} = \boldsymbol{\beta} * I_t, \quad I_{t+2} = \boldsymbol{\beta}^2 * I_t \tag{2}$$

其中， I_t、 I_{t+1} 和 I_{t+2} 是在时间步 t、（$t+1$）和（$t+2$）时帧中的信息量。

以四个参考帧为例。为便于分析，我们假设每个参考帧对当前帧的贡献相同。压缩前三个 P 帧时，我们复制距离最远的参考帧。可得

$$L_{1_f} = L_1 \tag{3}$$

$$L_{2_f} = \frac{4 * L_{1_f} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{4}$$

$$L_{3_f} = \frac{3 * L_{1_f} * (1-\boldsymbol{\beta}^2) + L_{2_f} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{5}$$

$$L_{4_f} = \frac{2 * L_{1_f} * (1-\boldsymbol{\beta}^3) + L_{2_f} * (1-\boldsymbol{\beta}^2) + L_{3_f} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{6}$$

信息总损失为

$$\begin{aligned} \boldsymbol{L_{total_f}} &= L_{1_f} + L_{2_f} + L_{3_f} + L_{4_f}, \\ &= L_1 * (1 + (1-\boldsymbol{\beta}) * (1+\boldsymbol{\alpha}) + \frac{3 * (1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})}{4} + \frac{(1-\boldsymbol{\beta})^2 * (1+\boldsymbol{\alpha})^2}{4} \\ &+ \frac{2 * (1-\boldsymbol{\beta}^3) * (1+\boldsymbol{\alpha})}{4} + \frac{(1-\boldsymbol{\beta}) * (1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})^2}{4} \\ &+ \frac{3 * (1-\boldsymbol{\beta}) * (1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})^2}{16} + \frac{(1-\boldsymbol{\beta})^3 * (1+\boldsymbol{\alpha})^3}{16}) \end{aligned} \tag{7}$$

类似地，当我们复制最近的参考帧时，我们有

$$L_{1_n} = L_1 \tag{8}$$

$$L_{2_n} = \frac{4 * L_{1_n} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{9}$$

$$L_{3_n} = \frac{L_{1_n} * (1-\boldsymbol{\beta}^2) + 3 * L_{2_n} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{10}$$

$$L_{4_n} = \frac{L_{1_n} * (1-\boldsymbol{\beta}^3) + L_{2_n} * (1-\boldsymbol{\beta}^2) + 2 * L_{3_n} * (1-\boldsymbol{\beta})}{4} * (1+\boldsymbol{\alpha}) \tag{11}$$

信息的总损失为

$$\begin{aligned}\boldsymbol{L_{total_n}} &= L_{1_n} + L_{2_n} + L_{3_n} + L_{4_n}, \\ &= L_1 * (1 + (1-\boldsymbol{\beta}) * (1+\boldsymbol{\alpha}) + \frac{(1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})}{4} + \frac{3 * (1-\boldsymbol{\beta})^2 * (1+\boldsymbol{\alpha})^2}{4} \\ &+ \frac{(1-\boldsymbol{\beta}^3) * (1+\boldsymbol{\alpha})}{4} + \frac{(1-\boldsymbol{\beta}) * (1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})^2}{4} \\ &+ \frac{(1-\boldsymbol{\beta}) * (1-\boldsymbol{\beta}^2) * (1+\boldsymbol{\alpha})^2}{8} + \frac{3 * (1-\boldsymbol{\beta})^3 * (1+\boldsymbol{\alpha})^3}{8})\end{aligned} \tag{12}$$

利用韦达定理和一元微分学，我们可以得到：

$$\boldsymbol{L_{total_n}} < \boldsymbol{L_{total_f}} \tag{13}$$

若 $0 < \boldsymbol{\alpha} < \frac{-b+\sqrt{b^2-4*a*c}}{2*a} < 1$，其中 $a = 5 * (1-\boldsymbol{\beta})^3$， $b = -9 * \boldsymbol{\beta}^3 + 17 * \boldsymbol{\beta}^2 - 27 * \boldsymbol{\beta} + 17$ 且 $c = 2 * \boldsymbol{\beta}^3 - 28 * \boldsymbol{\beta}^2 - 30 * \boldsymbol{\beta}$。

根据相关性的概念， $0 < \boldsymbol{\beta} < 1$，因此 $0 < \frac{-b+\sqrt{b^2-4*a*c}}{2*a} < 1$。~~这意味~~着，当某种 NVC 方法中信息损失的百分比增量 $\boldsymbol{\alpha}$ 低于临界值时，采用近帧复制可以减少信息损失，并实现更高的压缩效率。

4 实验

4.1 实验设置

按照 MLVC [10]，我们使用前四个解码帧作为参考帧。

数据集。 我们使用 Vimeo-90k [15] 进行训练，并使用 HEVC B 类、C、D、E [2],HEVC RGB类 [5], UVG [16]和 MCL-JCV [17]进行测试。训练时，我们将 Vimeo-90k 视频随机裁剪为 256x256 的图像块。

实现细节。 我们采用与 TCM[5]相同的图像压缩方法。我们使用质量指标均方误差（MSE）对模型进行优化，并且λ 从集合 (256, 512, 1024, 2048) 中选取。我们采用逐步渐进式训练策略来优化整个网络。对于前 3 个 P帧，我们复制最近的参考帧，以获得所需的 4 帧。我们使用峰值信噪比（PSNR）评估重建帧与原始帧之间的失真性能。我们还使用每像素比特数（bpp）衡量用于编码表示的比特数，包括运动矢量和上下文潜码。所有实验均在 2 块 NVIDIA V100 GPU 上进行，批量大小为 4，并使用 AdamW优化器。

4.2 实验结果

我们在 TCM [5], 上实现了 Butterfly，并采用与 TCM 相同的配置，以确保比较公平。在实际应用中，帧内周期设为 32。为展示所提方法的优势，我们将其与现有的传统视频编解码器进行比较，包括采用最高压缩比低延迟编码

[2]我们使用的 JM、HM 和 VTM 软件版本分别为 JM-19.0、HM-16.2 和 VTM-13.2，它们分别代表 H.264/AVC、H.265/HEVC 和 H.266/VVC 的最佳编码器。

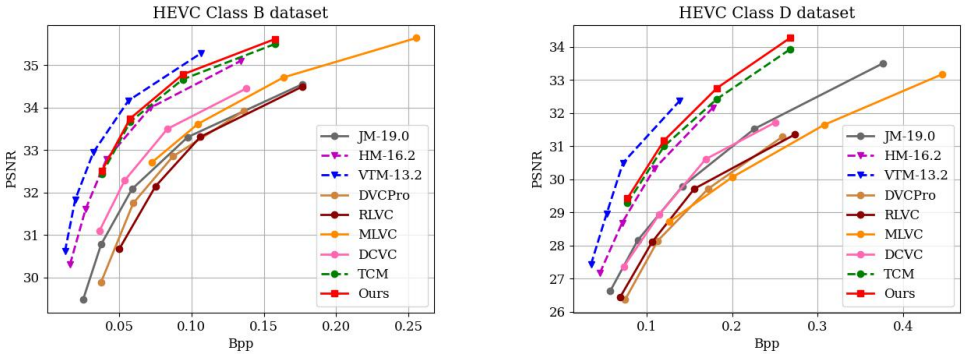

图 4：峰值信噪比和每像素比特数曲线。

表 1：峰值信噪比的 BD-Rate（%）比较。锚点为 HM-16.2。

	B	C	D	E	红绿蓝 UVG	MCL-JCV	
HM-16.2 [19]	0.0	0.0	0.0	0.0	0.0	0.0	0.0
JM-19.0 [18]	96.9	56.6	50.0	80.5	102.4	108.1	95.4
VTM-13.2 [20]	-28.8	-29.0	-26.5	-29.1	-29.7	-28.9	-31.2
DVCPro [11]	123.7	124.0	93.6	283.0	102.1	137.7	99.3
RLVC [8]	122.6	118.9	81.2	246.2	114.2	140.1	124.8
MLVC [10]	61.4	124.1	96.1	138.8	82.1	66.5	66.8
DCVC [4]	56.0	76.9	52.8	156.8	51.9	67.3	42.8
TCM [5]	-5.3	15.1	-5.4	18.5	-14.4	-9.0	-3.2
Ours	-10.4	7.5	-12.6	8.3	-17.5	-15.6	-6.2

设置的 JM [11], HM [10], 和 VTM [8], 。我们还将其与以往的 NVC 方法进行比较，包括 DVCPro [11], MLVC [10], RLVC [8], DCVC [4]和 TCM [5]。

表 1 展示了不同测试数据集上的 BD-Rate。我们选择 HM 作为基准。负值表示与 HM 相比节省了码率，正值表示码率增加。可以看出，我们的方法在所有测试数据集上均优于 TCM [5]，后者是我们的单参考帧基础模式。此外，我们的方法在部分数据集上也优于传统编解码器 HM。与 HM 相比，我们的方法在 HEVC D类数据集上实现了 -12.6% 的码率节省（这意味着与 TCM 相比实现了 -7.6% 的码率节省）。我们还绘制了 HEVC B 类和 D 类数据集上的 RD 曲线，以直观展示实验结果，如图 4 所示。

4.3 消融研究

我们开展了消融研究，以验证所提出的组件（Butterfly）和方法（近帧复制）的有效性。为简化实验，所有实验均使用 HEVC 测试集。比较结果以 BD-Rate（%）进行衡量。

Butterfly。在 Butterfly 中，下采样过程彼此独立，而上采样过程会融合来自不同帧的多尺度特征。因此，我们设计了两组对比实验。第一组实验中，上采样和下采样过程均采用联合方式，即在下采样前拼接特征。这是 MLVC [10] 采用的特征融合方法。第二组实验中，上采样和下采样过程均彼此独立，即在上采样后拼接特征。

表 2：上采样和下采样的方法。

	B	C	D	E	RGB 平均	
所提出的 Butterfly	0.0	0.0	0.0	0.0	0.0	0.0
上下采样联合	0.6	1.1	0.8	1.0	0.5	0.8
上下采样独立	0.8	1.2	0.7	1.0	0.8	0.9

表 2 对比了上述三种特征融合方法的性能。从表中可以看出，我们的方法性能最佳。这是因为我们充分考虑了各帧特征的独立性，并有针对性地设计了特征融合方法，而非简单地进行拼接。

近帧复制 压缩前三个 P 帧时，需要复制一帧已重建的帧，以满足使用四个参考帧的要求。在第 3 节中，我们证明，对于我们的多参考帧 NVC 方法，复制最近的参考帧优于复制最远的参考帧。在本小节中，我们通过对比实验对此进行验证。

表 3：复制帧的位置。

	B	C	D	E	RGB 平均	
近帧复制（**我们的**）	0.0	0.0	0.0	0.0	0.0	0.0
远帧复制	1.4	1.0	1.5	2.2	1.6	1.5

我们在 Butterfly 上对近帧复制和远帧复制进行了实验。对比实验结果如表 3 所示。从表中可以看出，近帧复制能够带来显著的性能提升。

5 结论

本文探索并设计了一种用于神经视频压缩的 Butterfly 多参考帧特征传播机制。该机制能够有效融合多个先前参考帧的特征，避免某一帧中的关键信息在融合过程中被消除，并为后续模块生成更准确的时序上下文条件先验。此外，我们证明，当参考帧数量不足以满足所需数量时，复制最近的参考帧效果更好。基于 Butterfly 和近帧复制，我们提出了一种面向低时延编码场景的多参考帧神经视频压缩方案。实验结果表明，与其他现有神经视频压缩方案相比，我们的方法能够实现更高的压缩效率。

致谢

我们感谢 Ren Yang（苏黎世联邦理工学院）和 Zongyu Guo（中国科学技术大学）提供的宝贵讨论。本研究得到国家自然科学基金 U21B2012 和 62072013、深圳市优秀科技创新人才培养项目 RCJC20200714114435057，以及深圳市科研项目 20180608092141929O 的资助。

参考资料

[1] Gary J Sullivan、Pankaj N Topiwala 和 Ajay Luthra，《H.264/AVC 高级视频编码标准：概述及保真度范围扩展简介》，《数字图像处理应用》第二十七卷，第 5558 卷，第 454–474 页，2004 年。[2] Gary J Sullivan、Jens-Rainer Ohm、Woo-Jin Han 和 Thomas Wiegand，《高效视频编码（HEVC）标准概述》，*IEEE*《视频技术电路与系统汇刊》，第 22 卷，第 12 期，第 1649–1668 页，2012 年。[3] Benjamin Bross、Ye-Kui Wang、Yan Ye、Shan Liu、Jianle Chen、Gary J Sullivan 和 Jens-Rainer Ohm，《通用视频编码（VVC）标准及其应用概述》，*IEEE TCSVT*，第 31 卷，第 10 期，第 3736–3764 页，2021 年。[4] Jiahao Li、Bin Li 和 Yan Lu，《深度上下文视频压缩》，《神经信息处理系统进展》，第 34 卷，第 18114–18125 页，2021 年。[5] Xihua Sheng、Jiahao Li、Bin Li、Li Li、Dong Liu 和 Yan Lu，《面向学习式视频压缩的时间上下文挖掘》，*arXiv* 预印本 *arXiv:2111.13850*，2021 年。[6] Jiahao Li、Bin Li 和 Yan Lu，《用于神经视频压缩的混合时空熵建模》，载于第 *30* 届 *ACM*国际多媒体会议论文集，2022 年，第 1503–1511 页。[7] Ren Yang、Fabian Mentzer、Luc Van Gool 和 Radu Timofte，《通过分层质量和循环增强进行视频压缩学习》，载于*IEEE/CVF* 计算机视觉与模式识别会议论文集，2020 年。[8] Ren Yang、Fabian Mentzer、Luc Van Gool 和 Radu Timofte，《基于循环自编码器和循环概率模型的视频压缩学习》，*IEEE*《信号处理精选主题期刊》，第 15 卷，第 2 期，第 388–401 页，2021 年。[9] Ren Yang、Radu Timofte 和 Luc Van Gool，《采用循环条件 GAN 的感知视频压缩》，载于*IJCAI* 论文集，2022 年。[10] Jianping Lin、Dong Liu、Houqiang Li 和 Feng Wu，《M-LVC：用于学习式视频压缩的多帧预测》，载于*IEEE/CVF CVPR* 论文集，2020 年。[11] Guo Lu、Xiaoyun Zhang、Wanli Ouyang、Li Chen、Zhiyong Gao 和 Dong Xu，《视频压缩端到端学习框架》，*IEEE TPAMI*，2020 年。[12] Johannes Ball´e、David Minnen、Saurabh Singh、Sung Jin Hwang 和 Nick Johnston，《基于尺度超先验的变分图像压缩》，*arXiv:1802.01436*，2018 年。[13] Anurag Ranjan 和 Michael J Black，《使用空间金字塔网络进行光流估计》，载于*IEEE CVPR* 论文集，2017 年，第 4161–4170 页。[14] Kelvin CK Chan、Shangchen Zhou、Xiangyu Xu 和 Chen Change Loy，《Basicvsr++：通过增强传播与对齐改进视频超分辨率》，载于*IEEE/CVF* 计算机视觉与模式识别会议论文集，2022 年，第 5972–5981 页。[15] Tianfan Xue、Baian Chen、Jiajun Wu、Donglai Wei 和 William T Freeman，《采用面向任务的光流进行视频增强》，*IJCV*，第 127 卷，第 8 期，第 1106–1125 页，2019 年。[16] Alexandre Mercat、Marko Viitanen 和 Jarno Vanne，《UVG 数据集：用于视频编解码器分析与开发的 50/120 fps 4K 序列》，载于第 *11* 届 *ACM* 多媒体系统会议论文集，2020 年，第 297–302 页。[17] Haiqiang Wang、Weihao Gan、Sudeng Hu、Joe Yuchieh Lin、Lina Jin、Longguang Song、Ping Wang、Ioannis Katsavounidis、Anne Aaron 和 C-C Jay Kuo，《MCL-JCV：基于 JND 的 H.264/AVC 视频质量评估数据集》，载于*2016* 年 *IEEE ICIP*。IEEE，2016 年。[18] “JM-19.0”，http://iphome.hhi.de/suehring/，访问日期：2021-11-16。[19] “HM-16.20”，https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-16.20，访问日期：2021-11-16。[20]“VTM-13.2”，https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/tree/VTM-13.2，访问日期：2021-11-16。