# 用于神经视频压缩的混合时空熵模型

李家豪
中国北京微软亚洲研究院
li.jiahao@microsoft.com

李斌
中国北京微软亚洲研究院
libin@microsoft.com

Yan Lu
中国北京微软亚洲研究院
yanlu@microsoft.com



## 摘要

对于神经视频编解码器来说，设计一个高效的熵模型来准确预测量化潜在表示的概率分布至关重要，但也极具挑战性。然而，现有的视频编解码器大多直接使用图像编解码器中现成的熵模型对残差或运动进行编码，不能充分利用视频中的时空特性。为此，本文提出了一种功能强大的熵模型，它能有效捕捉空间和时间依赖性。特别是，我们引入了潜先验，利用潜表征之间的相关性来挤压时间冗余。同时，我们还提出了双重空间先验，以并行友好的方式减少空间冗余。此外，我们的熵模型还具有多功能性。除了估计概率分布外，我们的熵模型还能生成空间通道量化步骤。这种内容自适应量化机制不仅能帮助我们的编解码器在单一模型中实现平滑的速率调整，还能通过动态比特分配改善最终的速率失真性能。实验结果表明，与 UVG 数据集相比，利用所提出的熵模型，我们的神经编解码器可以在 UVG 数据集上节省 18.2% 的码率。
H.266（VTM）的最高压缩比配置。它为神经视频编解码器的发展树立了新的里程碑。编码网址为 https://github.com/microsoft/DCVC。

## 综合传播战略概念

**• 计算方法 → 重建；计算机视觉问题。**

## 关键词

视频压缩、熵模型、量化

**ACM 参考格式：**

Jiahao Li, Bin Li, and Yan Lu.2022.用于神经视频压缩的混合时空熵建模。*第 30 届 ACM 国际多媒体会议（MM '22）论文集》，2022 年 10 月 10-14 日，葡萄牙里斯 本*。ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/ 3503161.3547845

*MM '22，2022 年 10 月 10-14 日，葡萄牙里斯本*


## 1 引言

近年来，神经图像编码技术蓬勃发展。在发展过程中，许多工作都集中在熵模型的设计上，以准确预测量化潜表示的概率分布，如因子化模型[7]、hy-per 先验[8]、自动回归先验[36]、混合高斯模型[12]、基于变换器的模型[23]等。得益于这些不断改进的熵模型，神经图像编解码器的压缩率已经超过了最好的传统编解码器 H.266 内部编码 [10]。受神经图像编解码器成功的启发，最近神经视频编解码器受到越来越多的关注。

现有的大多数神经视频编解码器工作可大致分为三类：基于残差编码的解决方案、基于条件编码的解决方案和基于三维自动编码器的解决方案。其中，许多方法 [6, 14, 19, 20, 28-30, 32, 34, 39, 40, 46, 49] 属于基于残差编码的解决方案。残差编码来自传统的混合视频编解码器。具体来说，首先生成运动补偿预测，然后对其与当前帧的残差进行编码。对于基于条件编码的解决方案 [24-27]，时间帧或特征是当前帧编码的条件。与残差编码相比，条件编码的熵值更低或相等[24]。至于基于 3D 自动编码器的解决方案 [15, 37, 42]，它是神经图像编解码器的 自然扩展，扩大了输入维度。但它会带来更大的编码延迟，并显著增加内存成本。总之，现有的这些研究大多侧重于如何通过探索不同的数据流或网络结构来生成优化的潜在表示。至于熵模型，它们通常直接使用神经图像编码解码器中的现成解决方案（如超先验[8]和自动回归先验 [36]）来编码潜在表示。在设计视频熵模型时，空间-时间相关性尚未得到充分探讨。因此，以前的 SOTA（最先进的）神经视频编解码器 [41] 的 RD（速率-失真）性能有限，仅略高于 2013 年发布的 H.265。

因此，本文提出了一种能有效利用空间和时间相关关系的综合熵模型，进而帮助神经视频编解码器超越最新的传统标准 H.266。我们特别引入了潜先验和双空间先验。潜在先验探索了各帧潜在表示的时间相关性。前一帧的量化潜在表示被用来预判当前帧中潜在表示的分布。通过级联训练策略，潜表征的传播链得以形成。这使我们能够在当前帧的潜表征与长距

参考帧。这种连接有助于神经编解码器进一步挤压潜在表征中的时间冗余。在我们的熵模型中，提出了双重空间先验来减少空间冗余。现有的神经编解码器大多依赖自动回归先验[36]来探索空间相关性。然而，自动回归先验是一种序列化解决方案，遵循严格的扫描顺序。这种解决方案不适合并行处理，推理速度非常慢。相比之下，我们的双空间先验是一种两步编码解决方案，遵循棋盘式上下文模型 [16]，更省时。在 [16] 中，所有通道都使用相同的编码顺序（偶数位置总是先编码，然后作为奇数位置的上下文）。由于有时先对偶数位置编码比先对奇数位置编码的 RD 性能更差，因此无法有效处理各种视频内容。因此，为了解决这个问题，我们的双空间先验引入了一种机制，即首先对奇数和偶数位置的半潜在表示进行编码，然后左潜在表示的编码可以从所有位置的上下文中获益。同时，在两步编码过程中，还可以利用整个信道的相关性。在不带来额外编码依赖性的情况下，双空间先验使空间上下文的范围扩大了一倍，并利用了信道上下文。这将导致更准确地预测分布情况。

对于神经视频编解码器来说，另一个挑战是如何在单一模型中实现平滑的速率调整。对于传统编解码器，可通过调整量化参数来实现。然而，大多数神经编解码器缺乏这种能力，而是使用固定的量化步长（QS）。为了实现不同的速率，编解码器需要重新训练。这会带来巨大的训练和模型存储负担。为了解决这个问题，我们引入了一种多粒度的自适应量化机制，该机制由我们的熵模型提供动力。在我们的设计中，整个 QS 由三个不同的粒度决定。首先，全局 QS 由用户为特定目标速率设定。然后再乘以信道 QS，因为不同的信道包含不同重要性的信息，这与信道注意机制类似[18]。最后，再乘以我们的熵模型生成的空间信道 QS。这有助于我们的编解码器应对各种视频内容，并在每个位置实现精确的速率调整。此外，我们还注意到，使用熵模型学习 QS 不仅有助于我们的编解码器在单一模型中获得平滑的速率调整能力，还能提高最终的 RD 性能。这是因为熵模型将学会为更重要的内容分配更多比特，而这些内容对当前帧和后续帧的重建至关重要。这种内容自适应量化机制可实现动态比特分配，从而提高最终压缩率。

在多功能熵模型的支持下，我们的神经编解码器只需一个模型，就能比以前的 SOTA 神经视频编解码器节省大量比特率。例如，在 UVG 数据集 [3] 上，比 DCVC [27] 节省了 57.1% 的比特率。更好的是，我们的神经编解码器的性能超过了使用最高压缩比设置的低延迟配置的最佳传统编解码器 H.266（VTM）[4, 10]。对于 UVG 数据集，如果以 PSNR 为导向，比 H.266 (VTM) 平均节省 18.2% 的比特率。如果以 MS-SSIM 为导向，则相应的比特率节省为 35.1%。这些实质性的改进表明，我们的模型为神经视频编解码器的发展树立了新的里程碑。

我们的贡献概述如下：

- 针对神经视频编解码器，我们设计了一个功能强大、并行友好的熵模型，以改进对概率的预测。

  bility 分布。所提出的潜先验和双空间先验可以分别有效地捕捉时间和空间上的延迟。这有助于我们进一步压缩视频中的冗余。
- 我们的熵模型还具有多功能性。除了分布参数，还能生成空间信道的 QS。

  通过多粒度量化机制，我们的神经编解码器能够在单一模型中实现平滑的速率调整。同时，空间信道的 QS 是内容自适应的，通过动态比特分配提高了最终的 RD 性能。
- 在所提出的熵模型的支持下，我们的神经视频编解码器将压缩率推向了一个新的高度。对据我们所知，我们的编解码器是首款采用最高压缩比配置的端到端神经视频编解码器，其压缩比超过了 H.266（VTM）。比特率比

  在 UVG 数据集上，H.266（VTM）的 PSNR 为 18.2%。如果面向 MS-SSIM，节省的比特率甚至更高。

## 2 相关工作

### 2.1 神经图像压缩

近年来，神经图像编码技术发展迅速。早期的工作 [43] 使用基于压缩自动编码器的框架，实现了与 JPEG 2000 相似的 RD 性能。最近，许多工作都集中在熵模型的设计上。Ballé *等人*[7] 提出了因式分解模型，并获得了比 JPEG 2000 更好的 RD 性能。超先验[8]引入了分层设计，并使用额外的比特来估计分布，其结果与 H.265 相当。随后，自动回归先验[36]被提出来探索空间相关性。与超先验相比，它能获得更高的压缩比。不过，自动回归先验以序列化的顺序对所有空间位置进行推理，因此速度相当慢。棋盘式上下文模型
[16] 通过引入两步编码解决了复杂性问题。此外，为了进一步提高压缩比，有人提出了混合高斯模型[12]，其 RD 结果与 H.266 相当。最近，视觉变换器备受关注，相应的熵模型[23]帮助神经图像编解码器超越了 H.266 内部编码。

### 2.2 神经视频压缩

神经图像编解码器的成功也推动了神经视频编解码器的 发展。开创性工作 DVC [33] 沿用了传统编解码器，采用基于残差编码的框架，首先生成运动补偿预测，然后使用超先验[8]对残差进行编码。在自动回归先验[36]的帮助下，其后续作品 DVCPro 实现了更高的压缩率。

最近的大多数研究都采用了这种基于运动估计 [22, 38] 和残差编码的框架。随后，又提出了更先进的网络结构，用于生成优化的残差或运动。例如，[50] 通过学习参数对残差进行自适应缩放。尺度空间中的光流估计[6]。

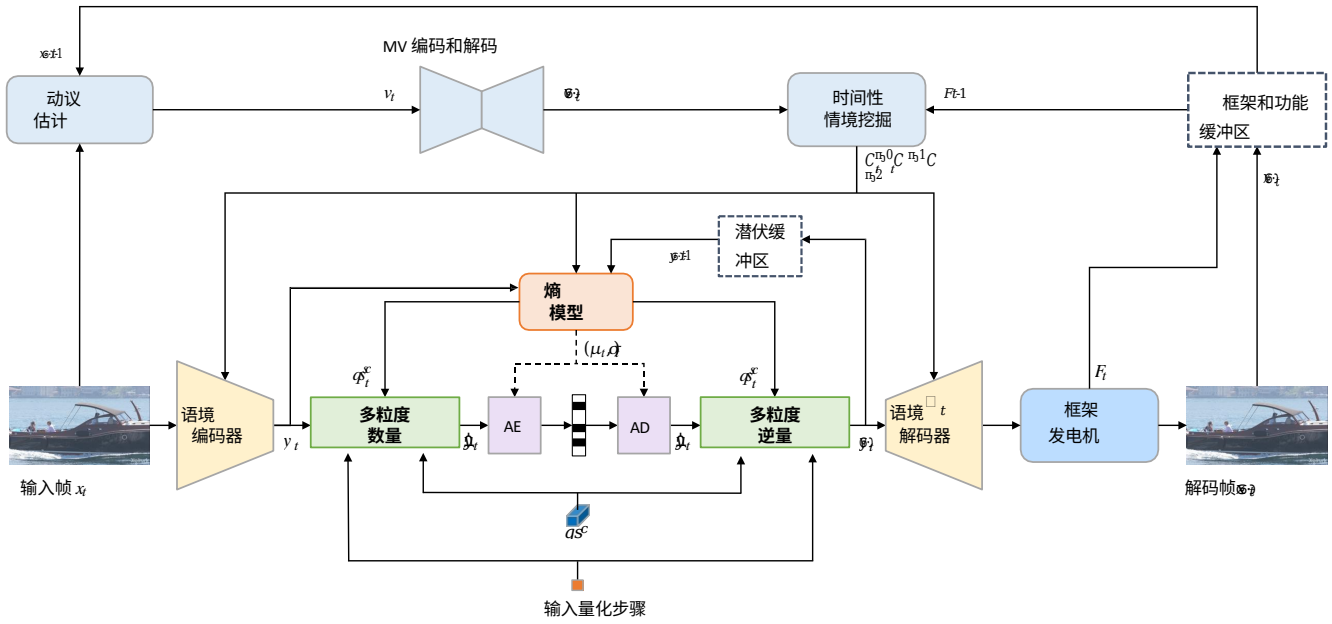



**图 1：我们方法的整体框架。Quant shorts 表示量化。AE 和 AD 是算术编码器和解码器。用于运动矢量编码的熵模型和量化机制与 $y_t$ 的设计相似，为简化起见，我们省略了它们。**

提出了减少快速运动区域残余能量的方法。在 [19] 中，应用了速率失真优化来改进运动编码。变形补偿[20]用于改善特征空间的预测。Lin 等人[28]提出使用多个参考帧来减少残余能量。在 [28, 39] 中，引入了运动预测来提高运动编码效率。

除了残差编码，还研究了其他编码框架。例如，3D 自动编码器 [15, 37, 42] 被用于同时对多个帧进行编码。它是神经图像编解码器的自然扩展，扩大了输入维度。不过，这种框架会带来明显的编码延迟，不适合实时场景。另一种新兴的编码框架是条件编码，其熵边界小于或等于残差编码 [24]。例如，Ladune 等人[24-26] 使用条件编码对前景内容进行编码。在 DCVC [27] 中，条件是可扩展的高维特征，而不是三维预测帧。接下来的工作 [41] 通过引入特征传播和多尺度时间上下文，进一步提高了压缩率。

然而，现有的神经视频编解码器大多侧重于如何生成优化的潜在表示和设计其中的网络结构。至于用于编码潜在表示的熵模型，则直接使用神经图像编解码器中的现成解决方案。在本文中，我们将重点放在有效利用空间和时间相关性的熵模型设计上。事实上，一些工作也已经开始对此进行研究。例如，[31] 提出了条件熵编码。从时间特征中提取的时间上下文先验在 [27, 41] 中被使用。Yang 等人[49] 提出了递归熵算法。

时间相关性。尽管 [27] 中的工作也研究了空间相关性，但使用的是自回归先验，导致编码速度非常慢。我们需要一种既能充分利用时空相关性，又能降低复杂度的熵模型。为了满足这一要求，我们专门设计了潜在先验和时间效率双空间先验来装备熵模型，从而将压缩率推向一个新的高度。此外，所有这些方法 [27, 31, 41, 49] 都需要为每个速率点训练单独的模型。相比之下，我们设计了一种由熵模型驱动的多粒度量化方法。这种内容自适应量化机制有助于我们的编解码器在单一模型中实现平滑的速率调整。通过增益单元[13]，神经图像编解码器对单一模型中的速率调整进行了研究。然而，据我们所知，我们的编解码器是第一个通过多粒度量化获得这种能力的神经视频编解码器。

## 3 建议方法

### 3.1 框架概述

为了提高输入帧 $x_t$ （$t$ 为帧索引）的压缩率，我们采用了基于条件编码的框架

而不是基于残差编码的框架。具体来说，我们遵循 DCVC [27] 及其改进工作 [41]，然后重新设计其中的核心模块。我们的编解码器框架如图 1 所示。如图所示，整个编码过程可大致分为三个步骤：时序上下文生成、上下文编码/解码和重构。

**时空背景生成。**为了充分探索时间相关性，我们生成多尺度上下文 $C_t^0, C_t^1, C_t^2$ at dif-

模型。然而，这些作品 [27, 31, 41, 49] 更侧重于通过时间上下文挖掘模块利用 时空上下文挖掘模块（更多

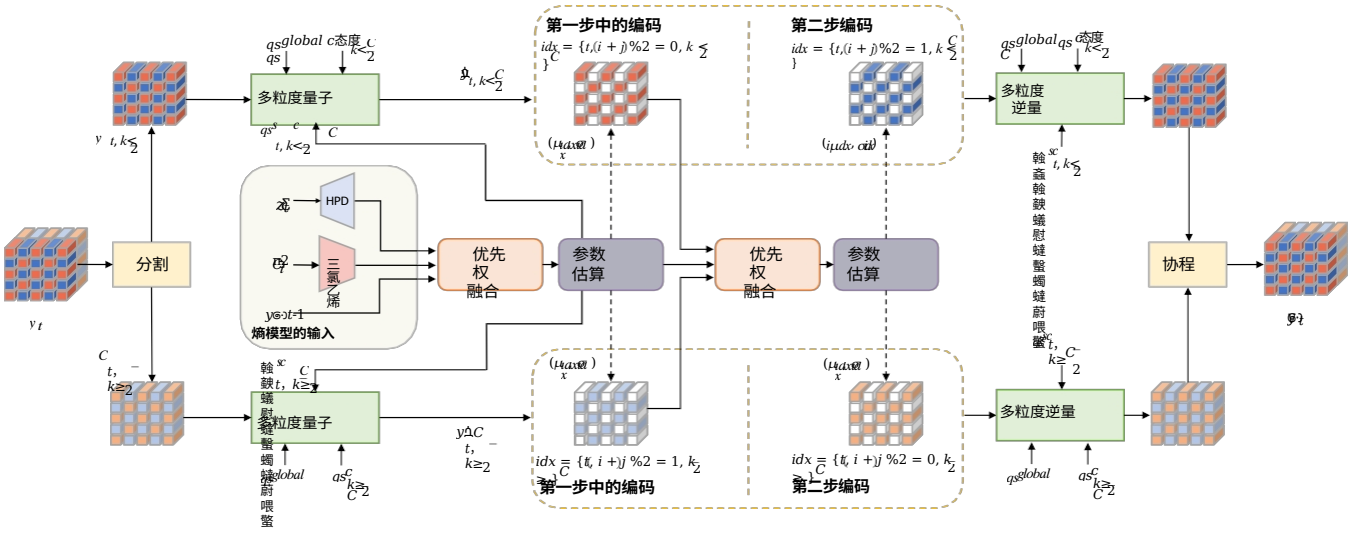



**图 2：我们的熵模型和量化潜在表示的编码说明。$i$、$j$ 和 $k$ 分别表示高度、宽度和通道索引。$C$ 是通道编号。括号$sg^{lobal}$，括号$s^{ch}$，和 括号$s^{sc}$ 分别是全局量化、信道量化和空间信道量化步骤。HPD 表示超先验解码器。TCE 指时间上下文编码器。对于为了简化，在两步编码中省略了参数估计之后的算术编码器和解码器。**

该模块的详细介绍见 [41]）。为了携带更丰富的信息
此外，我们还使用时间特征 $F_{t-1}$ 而非前一解码帧 $x_{t-1}$ 作为模块输入。至于运动估计，我们采用轻量级 SPyNet [38] 进行加速。

**语境编码/解码。**在多尺度语境的条件下，当前帧$x_t$ 被转换为潜在代表
在这种情况下，上下文编码器会将$y_t$ 。为了节省比特率，$y_t$ 被量化为$y^{-}_t$ ，然后被发送到生成比特流的算术编码器。在解码过程中，$y^{-}_t$ 将被解码
通过算术解码器从比特流中提取，并反向量化为
$_t$ 。同样以多尺度上下文为条件，上下文解码器从$y^{\wedge}_t$ 解码高分辨率特征$F^{\wedge}_t$ 。在这一编码/解码过程中，如何通过熵模型准确估计$y^{-}_t$ 的分布对降低比特率至关重要。为此，我们提出了时空混合熵模型
(第 3.2 节）。为了支持单一模型中的平滑速率调整，我们提出了由熵模型驱动的多粒度量化方法（第 3.3 节）。

**重建。**在获得高分辨率特征 $F^{\wedge}_t$ 之后，我们的目标是通过帧生成器生成高质量的重构帧 $x_t$ 。与 DCVC [27] 和 [41] 不同的只是
我们建议使用基于 W-Net [47] 的结构，将两个 U-Net 连接起来。这种网络设计可以在可接受的复杂度下有效扩大模型的感受野。这使得模型的生成能力更强。

## 3.2 混合时空熵模型

对于算术编码，需要知道$y^{-}_t$ 的概率质量函数（PMF）才能对其进行编码。然而，我们并不知道它真正的 PMF
$p(y^{-}_t)$），通常用估计的 PMF $\triangle(y^{-}_t)$来近似。阈值
交叉熵 $E_{y^{-}_t \sim p}\ [-log_2\ \triangle(y^{-}_t)]$捕捉到的是 "熵 "的平均数量。
算术编码所需的比特，而不考虑可忽略的开销。在本文中，我们沿用现有研究成果 [2]，假设 $\triangle(y^{-}_t)$ 遵循拉普拉斯分布。因此，我们的目标
是设计一个熵模型，该模型可以准确地估算出
$_t$的分布参数，以减少交叉熵。
为了改进估计，对于每个元素 $y^{-}_{t,i,j,k}$ $(i, j,$ 和 $k)$ 是高度、宽度和通道索引），其中我们需要充分挖掘其与已知信息（如之前解码的潜在表征、时间上下文特征等）之间的相关性。从理论上讲，$y^{-}_{t,i,j,k}$ 可能与所有这些信息相关。
之前所有解码位置的信息。对于传统的
由于空间巨大，传统编解码器无法明确利用这种相关性。因此，传统的编解码器通常使用简单的手工规则来利用几个邻近位置的上下文。相比之下，深度学习能够自动挖掘巨大空间中的相关性。

因此，我们建议向熵模型输入多维输入，让模型从丰富的高维输入中提取互补信息。图 2 展示了我们的熵模型。如图所示，输入信息中没有
$_t$虽然$C^{-2}$和$y^{\wedge}_{t-1}$都来自时间方向，但它们具有不同的特征。虽然 $C^{-2}$ 和 $y^{\wedge}_{t-1}$ 都来自时间方向，但它们具有不同的特征。例如，$C^{-2}$ 在 4 倍向下采样分辨率下使用
它还包含大量的运动信息（更多详情请参见
[41]）。相比之下，$y^{\wedge}_{t-1}$ 属于潜在表示域，分辨率为 16 倍下采样，与$y_t$ 具有更相似的特征。因此，$C^{-2}$ 和 $y^{\wedge}_{t-1}$ 可以为改进估计提供互补的辅助信息。此外，我们还注意到
我们采用级联训练策略 [11，41]，梯度将反向传播到多个帧。在这种训练策略下，潜表征的传播链就形成了。这意味着

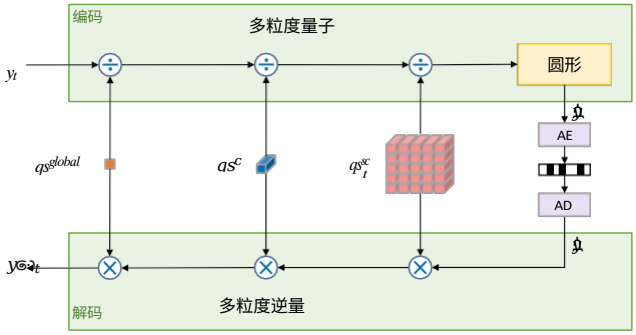



**图 3：多粒度量化和相应的反量化。**

此外，还建立了当前帧与远程参考帧之间的联系。这种连接非常有助于提取多个帧的潜在表征之间的相关性，从而对分布进行更准确的预测。

除了使用潜在先验$y\hat{}_{t-1}$ 来丰富输入外，我们的环境模型还采用了双重空间先验来利用空间先验。

我们采用棋盘模型[16]，而不是常用的自回归模型[36]。为了追求一种省时的机制，我们采用了棋盘模型[16]，而不是常用的自动回归模型[36]，因为后者会严重降低编码速度。然而，在最初的棋盘模型中，所有信道都使用相同的编码顺序，即偶数位置总是先编码，然后作为奇数位置编码的上下文。这样的编码顺序无法处理各种视频，因为有时先编码偶数位置比先编 码 奇数位置的 RD 性能更差。因此，我们设计了双空间先验，即同时先对偶数和奇数位置的半数通道进行编码。

如图 2 所示，潜在表示沿信道维度分成两块。在第一步编码过程中，图 2 中的上述分支将编码偶数位置$y_{t,(i+j)\%2=0,k<\frac{C}{2}}$在第一大块中。同时，下面的分支对奇 数 进行编码$y_{t,(i+j)\%2=1,k\geq\frac{C}{2}}$的位置。这两个块中的未编码点（即图 2 中的白色区域）被设置为为零。经过第一步编码后，编码后的$y_{t,(i+j)\%2=0,k<\frac{C}{2}}$和$y_{t,(i+j)\%2=1,k\geq\frac{C}{2}}$ 融合在一起，然后进一步生成 te 第二步编码的上下文。在第二步编码过程中，对每个语块的左侧位置进行编码。上述分支编码的奇数位置$y_{t,(i+j)\%2=1,k<\frac{C}{2}}$在第一大块中，下面是分支对第二块中的偶数位置$y_{t,(i+j)\%2=0,k\geq\frac{C}{2}}$进行编码。如图 2 所示，这种编码方式可使 nables 第二步编码可以从所有位置的上下文中获益。与最初的棋盘模型[16]相比，空间上下文的范围扩大了一倍，对分布的预测更加准确。值得注意的是，在这两个步骤中，第一块和第二块都将被添加，然后发送到算术编码器。因此，与 [16] 相比，我们的双空间先验不会带来任何额外的编码延迟。

此外，从渠道维度来看，我们的双空间先验还能挖掘各通道之间的相关性。空间先验

编码$y_{t,(i+j)\%2=0,k<\frac{C}{2}}$ 在第一步编码过程中，可以也可用作$y$编码的共 ndition $y_{t,(i+j)\%2=0,k\geq\frac{C}{2}}$

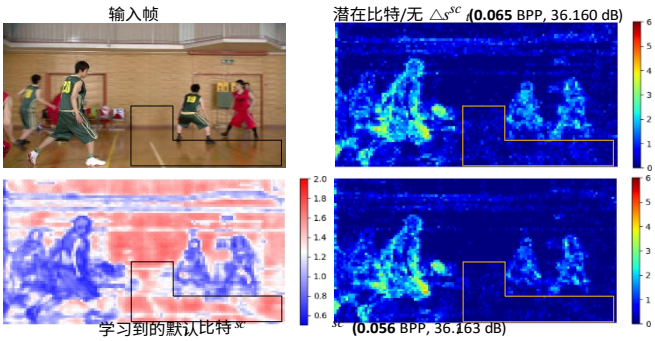



**图 4：$\triangle s^{sc}$ 的影响。BPP 指每像素比特数。在此示例中，在质量相似的情况下，$\Delta s^{sc}$ 使 BPP 降低了 13.8%。放大观看效果更佳。**

在第二步编码过程中。类似的情况还有$y_{t,(i+j)\%2=1,k\geq\frac{C}{2}}$和$y_{t,(i+j)\%2=1,k<\frac{C}{2}}$，其中信道的编码方向为 inv erse 。综上所述， r 提出的双空间先验通过更有效地利用空间和信道位置的相关性，进一步压缩了 $y_t$ 中的冗余。

## 3.3 单一模式的费率调整

现有的大多数神经编解码器都无法在单一模型中处理速率调整，这是一个很大的问题。为了实现不同的速率，需要通过调整 RD 损失中的权重来重新训练模型。这将带来巨大的训练成本和模型存储负担。这种缺陷要求建立一种机制，支持神经编解码器在单一模型中实现宽速率范围。为此，我们提出了一种自适应量化机制来实现这一功能。

如图 3 所示，我们的多粒度量化涉及三种不同的量化步骤（QP）：全局 QS$s\triangle sglo^{b}al$ 、信道维 QS$\triangle s^{ch}$ 和空间信道维 QS$\triangle s^{sc}$ 。幌$sgl^{obal}$ 仅为单一值，由用户输入设置，用于控制目标速率。由于所有位置的 QS 都是相同的，因此括号内的 会带来粗量化效果。因此，受信道注意机制[18]的启发，我们也设计了一个调制器 $\triangle s^{ch}$ 来扩展不同信道的 QS，因为不同信道携带的信息具有不同的重要性。然而、

不同的空间位置也有不同的特征，这是因为到各种视频内容。因此，我们按照 [21] 的方法学习了空间通道维度上的 $\Delta s^{sc}$ ，以实现对每一职位：

如图 2 所示，$^{sc}$ 由我们的熵模型生成。值得注意的是，对于每一帧图像，熵值都是动态变化的，以适应不同的图像。视频内容。这种设计不仅能帮助我们实现平滑的速率调整，还能通过内容自适应比特分配提高最终的 RD 性能。对重构至关重要的重要信息或后续帧编码所参考的重要信息将以较小的 QS 分配。我们展示了一个

图 4：$\triangle s^{sc}$ 的可视化示例。在这个例子中，模型了解到移动球员更重要，并产生这些区域的 QS 较小。相比之下，背景的 QS 较大，可节省大量比特率（如图 5 所示）。

黄线所示区域）。部分的消融研究 4.3 显示，$\triangle s^{sc}$ 可以在多个方面实现大幅改进。数据集。

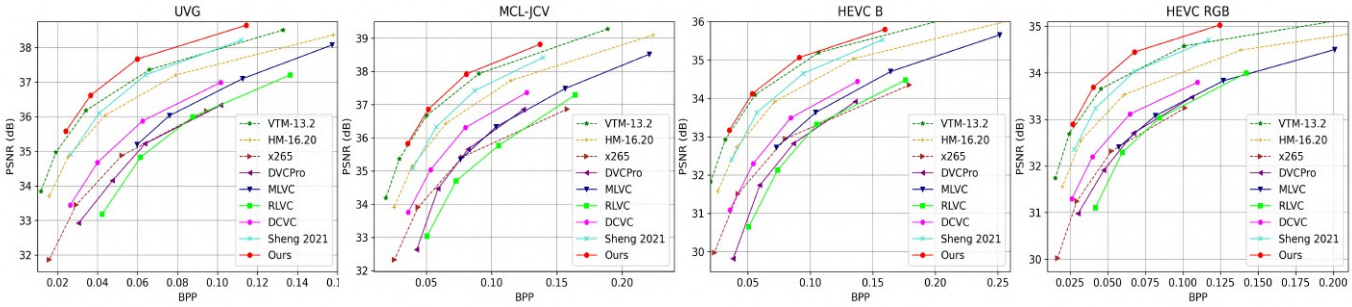



**图 5：PSNR 和 BPP 曲线。**

**表 1：PSNR 的 BD-Rate (%) 比较。锚点为 VTM-13.2。**

| | UVG | MCL-JCV | HEVC B | HEVC C | HEVC D | HEVC E | HEVC RGB | 平均 |
|---|---|---|---|---|---|---|---|---|
| VTM-13.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| HM-16.20 | 40.5 | 45.4 | 40.4 | 40.9 | 36.0 | 46.2 | 42.1 | 41.6 |
| x265 | 191.5 | 160.3 | 143.4 | 105.2 | 96.1 | 128.4 | 151.2 | 139.4 |
| DVCPro [34] | 227.0 | 180.8 | 209.8 | 220.6 | 166.4 | 446.2 | 178.5 | 232.8 |
| 区域联络中心 [49] | 224.2 | 214.3 | 207.0 | 212.4 | 149.2 | 392.8 | 195.5 | 227.9 |
| MLVC [28] | 113.6 | 124.0 | 118.0 | 213.7 | 166.5 | 237.6 | 151.2 | 160.7 |
| DCVC [27] | 126.1 | 98.2 | 115.0 | 150.8 | 109.6 | 266.2 | 109.6 | 139.4 |
| 盛 2021 [41] | 17.1 | 30.6 | 28.5 | 60.5 | 27.8 | 67.3 | 17.9 | 35.7 |
| 我们的 | -18.2 | -6.4 | -5.1 | 15.0 | -8.9 | 7.1 | -16.4 | -4.7 |

在解码过程中，会应用相应的反量化。需要注意的是，$\triangle g_b^{bal}$ 需要与比特流一起传送给解码器。不过，这种开销
由于每个帧或视频只传输一个数字（用户可灵活设置），因此可以忽略不计。调制器 $\triangle s^{ch}$ 属于我们的神经编解码器的一部分，在训练过程中学习。

## 4 实验结果

### 4.1 实验装置

**数据集。**我们使用 Vimeo-90k [48] 进行训练。这些视频被随机裁剪成 256x256 块。测试时，我们使用与 [41] 相同的测试视频。所有这些测试视频都广泛用于传统和神经视频编解码器的评估，包括 HEVC Class B、C、D、E 和 RGB。此外，我们还使用了 UVG 的 1080p 视频。
[35] 和 MCL-JCV [44] 数据集也进行了测试。

**测试条件。**我们对每段视频的 96 个帧进行测试。内部周期设置为 32，而不是 10 或 12。原因是 32 内周期更接近实际应用中的实际使用情况。例如，与 12 帧内相比，32 帧内平均可为 HM 节省 23.8% [41] 的比特率。我们沿用了大多数现有作品[27, 33, 34, 41]的低延迟编码设置。压缩率用 BD-Rate [9]来衡量，负数表示比特率节省，正数表示比特率增加。

除了 x265 [5]（使用 *veryslow* 预设值），我们的基准测试还包括 HM-16.20 [1] 和 VTM-13.2 [4]，它们分别代表了 H.265 和 H.266 的最佳编码器。对于 HM 和 VTM，我们遵循 [41]，使用压缩比最高的配置。我们还比较了以前的 SOTA 神经视频编解码器，包括 DVCPro [34]、MLVC [28]、RLVC [49]、DCVC [27]，以及
盛 2021 [41]。

**实施和训练细节。**运动矢量潜在表示的熵模型和量化方法如下 $_t$ 。唯一不同的是熵模型的输入。在运动矢量编码中，输入是相应的
超先验和潜先验，即前一帧运动矢量的量化潜代表。对运动矢量进行编码时没有时间上下文先验，因为时间上下文的生成取决于解码后的运动矢量。此外，由于我们的目标是用单一模型处理多种速率，因此我们还训练了一种神经图像编解码器，支持这种内部编码能力。

在训练过程中，损失函数包括失真和速率：$Los = \lambda \cdot D + R$。$D$指的是输入和输出之间的失真。
帧和重建帧。失真可以是 $L_2$ loss
或针对不同视觉目标的 MS-SSIM [45]。$R$ 表示用于编码$y^{\cdot\cdot}_t$ 的比特和运动矢量的量化潜在表示，两者都与用于编码其运动矢量的比特相关。
相应的超先验

我们采用与 [41] 相同的多阶段训练方法。此外，为了训练支持速率调整的单一模型，我们在不同的优化步骤中使用了不同的$\lambda$值。为了简化训练过程，我们使用了 4 个$\lambda$值（85、170、380、840）。在训练过程中
值将通过 RD 损失来学习，每一个相关值都将通过 RD 损失来学习。对应的$\lambda$值。我们分别训练了图像和视频模型

**表 2：MS-SSIM 的 BD 速率（%）比较。锚点为 VTM-13.2。**

| | UVG | MCL-JCV | HEVC B | HEVC C | HEVC D | HEVC E | HEVC RGB | 平均 |
|---|---|---|---|---|---|---|---|---|
| VTM-13.2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| HM-16.20 | 36.9 | 43.7 | 36.7 | 38.7 | 34.9 | 40.5 | 37.2 | 38.4 |
| x265 | 150.5 | 137.6 | 129.3 | 109.5 | 101.8 | 109.0 | 121.9 | 122.8 |
| DVCPro [34] | 68.1 | 37.8 | 61.7 | 59.1 | 23.9 | 212.5 | 57.3 | 74.3 |
| 区域联络中心 [49] | 83.9 | 72.1 | 66.6 | 76.5 | 34.1 | 268.4 | 60.5 | 94.6 |
| DCVC [27] | 33.6 | 4.7 | 31.0 | 22.8 | 1.2 | 124.7 | 36.5 | 36.4 |
| 盛 2021 [41] | -10.1 | -24.4 | -24.1 | -23.3 | -37.3 | -8.0 | -25.9 | -21.9 |
| 我们的 | -35.1 | -46.8 | -48.1 | -44.6 | -55.7 | -47.5 | -47.0 | -46.4 |

且它们的 $\Delta s_{global}$ 值各不相同。值得注意的是，虽然我们在训练过程中只使用了 4 个$\lambda$值，但在测试过程中，通过手动调整$s^{\triangle global}$ ，模型仍然可以实现平滑的速率调整，相应的研究将在第 4.4 节中介绍。

## 4.2 与以往 SOTA 方法的比较

表 1 和表 2 分别显示了 PSNR 和 MS-SSIM 的 BD 速率（%）比较。以最佳传统编解码器 VTM 作为锚点。从表 1 可以看出，在所有数据集上，我们的神经编解码器比 VTM 平均节省了 4.7% 的比特率。相比之下，第二好的方法 Sheng 2021 [41]远远落后于 VTM，比特率提高了 35.7%。据我们所知，这是首个采用最高压缩比的端到端神经视频编解码器，其性能优于 VTM。
配置的一个重要里程碑。
的神经视频编解码器。特别是，我们的神经编解码器在 1080p 视频（HEVC B、HEVC RGB、UVG、MCL-JCV）中表现更好。图 5 显示了这些数据集的 RD 曲线。我们可以发现，在相同质量下，我们的编解码器消耗的比特最少。这些结果验证了我们的熵模型在利用卷积视频数据之间的相关性方面的有效性。此外，高分辨率视频在未来将更加普及，我们的神经编解码器的优势将更加明显。面向 MS-SSIM 时，我们的神经视频编解码器有较大的改进。如表 2 所示，在所有数据集上，我们比 VTM 平均节省了 46.4% 的码率。

## 4.3 消融研究

为了验证每个组件的有效性，我们进行了全面的消融研究。我们分析了熵模型输入、双空间先验、多粒度量化以及帧生成器设计的效果。为简化起见，我们在消融研究中仅使用 HEVC 测试集。比较结果以 BD-Rate (%) 度量。

**熵模型的输入。**如图 2 所示，我们的熵模型包含三种不同的输入：超先验、时间上下文先验和潜先验。表 3 比较了这些输入的有效性。移除潜先验后，比特率提高了 10.1%。如果只启用一种先验输入，第一和第二重要的输入分别是超先验（15.6%）和潜先验（19.3%）。从这些比较中，我们可以发现超先验仍然是

**表 3：熵模型输入的影响。**

| 超级优先 | 时间性先验上下文 | 潜伏优先 | B | C | D | E | RGB 平均值 | |
|---|---|---|---|---|---|---|---|---|
| ✓ | ✓ | ✓ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ✓ | ✓ | ▪ | 9.5 | 5.6 | 6.4 | 18.4 | 10.7 | 10.1 |
| ✓ | ▪ | ✓ | 9.2 | 8.8 | 9.1 | 15.0 | 9.1 | 10.2 |
| ▪ | ✓ | ✓ | 19.5 | 9.1 | 11.9 | 27.2 | 20.9 | 17.7 |
| ✓ | ▪ | ▪ | 12.4 | 14.7 | 17.9 | 23.4 | 9.4 | 15.6 |
| ▪ | ✓ | ▪ | 33.8 | 29.3 | 31.6 | 54.3 | 36.2 | 37.0 |
| ▪ | ▪ | ✓ | 19.5 | 19.3 | 19.2 | 21.1 | 17.3 | 19.3 |

**表 4：不同空间先验的效果。**

| | B | C | D | | ERGB 平均值 拟 | |
|---|---|---|---|---|---|---|
| 议的双重空间先验0 | .0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 前棋盘格 | 5.6 | 4.1 | 6.0 | 11.0 | 5.5 | 6.2 |
| 无空间先验 | 11.9 | 11.1 | 11.6 | 29.7 | 6.4 | 14.1 |
| 自回归先验 | -14.8 | -11.4 | -11.2 | - 5 | .6 -16.9 | -12.0 |

最重要。但是，通过我们的潜在先验来丰富熵模型输入也能大大节省比特率。

**不同的空间先验**表 4 比较了不同的空间先验模块对探索《侏罗纪公园》中内部相关性的影响。

$t$ 。测试的空间语境模型包括我们提出的双重空间先验、棋盘先验 [16]，以及并行不友好的自回归先验[36]。从表中可以看出，我们的双空间先验可以节省 14.1%的比特率。比棋盘先验提高了 6.2%。这验证了扩大空间上下文范围和利用跨信道相关性的好处。此外，我们还发现自动回归先验可以进一步节省大量比特率（12.0%）。不过，考虑到编码和解码速度非常慢，我们仍然没有在神经编解码器中采用这种方法。

**多粒度量化。**如第 3.3 节所述，熵模型支持的多粒度量化不仅能帮助我们实现单一模型的速率调整，还能通过内容自适应比特分配提高最终的 RD 性能。表 5 显示了 BD 速率比较。从表中我们可以发现，如果禁用整个熵模型，则会有 11.8%的损失。

**表 5：多粒度量化的 RD 比较。**

| | B | C | D | E | RGB | 平均 |
|---|---|---|---|---|---|---|
| 多粒度量化 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 無/無 墜墜 墜墜 墜墜 | 9.7 | 10.1 | 10.6 | 8.3 | 5.6 | 8.9 |
| 无多粒度量化 | 9.9 | 15.1 | 14.6 | 11.7 | 7.8 | 11.8 |

**表 6：对不同框架发生器的研究。**

| | B | C | D | E | RGB | 平均 |
|---|---|---|---|---|---|---|
| W-Net | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| U-Net | 6.3 | 7.6 | 9.1 | 7.7 | 8.6 | 7.9 |
| 2 残块 [27, 41] | 11.7 | 9.9 | 11.0 | 13.4 | 9.1 | 11.0 |
| 1 个残块 | 12.5 | 12.5 | 13.4 | 14.5 | 10.9 | 12.8 |

多粒度量化。如果只去掉空间-信道-量化步骤 $\triangle s^{sc}$，比特率将显著增加。这表明，$\triangle s^{sc}$ 降低 8.9%。 $t$ 在多学科教育中发挥重要作用粒度量化，因此有必要设计一个动态的位分配，可适应各种视频内容。

**帧发生器设计。**为了提高神经编解码器的生成能力，我们的帧发生器采用了基于 W 网络的结构来扩大感受野。为验证其有效性，我们对使用不同网络的帧发生器进行了比较，如表所示 6.我们比较了 W-Net、U-Net 和基于残差块 [27, 41] 的结构。从表中可以看出，W-Net 比使用残差块的普通网络节省了 10%以上的比特率。绑定两个 U-Net 的 W-Net 也比单一 U-Net 好 7.9%。因此，使用更多的 U-Net 有可能节省更多的比特率。不过，考虑到复杂性，我们目前使用的是 W-网络。

## 4.4 单一模式下的平稳费率调整

表 5 显示了多粒度量化的 RD 改进情况。本小节将研究我们的多粒度量化能否实现平滑的速率调整。对于我们的编解码器 全局量化步长 $\Delta sglobal$ 可在计算过程中灵活调整。 测试。它的作用类似于量化参数。 传统的视频编解码器。图 6 显示了我们的编解码器使用学习到的 4 个括号$\$lobal$ 值的结果，这些值是在训练过程中使用 4 个$\lambda$值以 RD loss 为指导生成的。此外，我们还手动生成了 30 个括号$sgalo^{ba}$ 值，这些值是在所学括号$^{sgaloba}$ 值的最大值和最小值之间进行插值的。从 30 个比率点的结果可以发现，我们的单一模型可以实现 费率调整平稳，没有任何异常值。相比之下，DCVC [27] 和 Sheng 2021 [41]需要为每个速率点建立不同的模型。

## 4.5 模型复杂性

如表 7 所示，我们将模型复杂度的模型大小、MAC（多层累加操作）、编码时间和解码时间与之前的 SOTA 神经视频编解码器进行了比较。我们使用 1080p 帧作为输入来测量这些数字。对于编码/解码时间，我们测量了英伟达 V100 GPU 上的时间，包括写入比特流和读取比特流的时间，如表 7 所示。

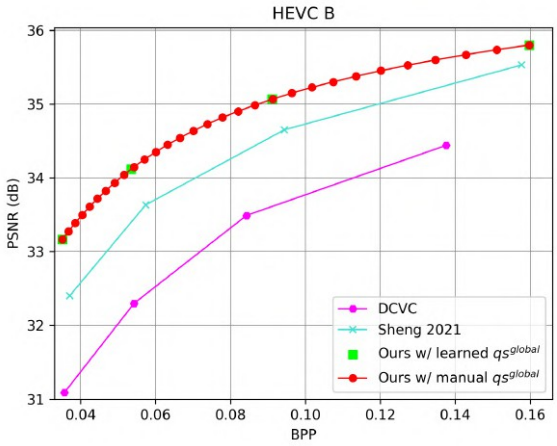



**图 6：单一模型中的费率调整。DCVC [27] 和 Sheng 2021 [41] 需要为每个费率点建立单独的模型。**

**表 7：复杂性比较**

| | 模型尺寸 | 互助会 | 编码时间 | 解码时间 |
|---|---|---|---|---|
| DCVC [27] | 35.2MB ×$N$ | 2.4T | 12,260ms | 35,590ms |
| 盛 2021 [41] | 40.9MB ×$N$ | 2.9T | 880ms | 470ms |
| 我们的 | 67.0MB ×1 | 3.3T | 986ms | 525ms |

注：$N$ 为速率点数。1080p 用于测试。

[41].如上所述，我们的模型支持在单一模型中进行速率调整。因此，我们大大减轻了模型训练和存储的负担。由于 DCVC [27] 使用的是并行不友好的自动回归先验模型，其编码/解码时间相当慢。相比之下，[41] 和我们的编解码器要快得多。与 [41] 相比，我们的编码/解码时间略有增加。然而，压缩比却被推向了一个新的高度（从盛 2021 超越 HM 到我们超越 VTM）。我们认为这是值得付出的代价。

## 5 结论

本文介绍了如何设计一种高效的熵模型，它不仅能帮助我们的神经编解码器实现比 VTM 更高的压缩率，还能在单一模型中平滑调整速率。其中，我们提出了潜在先验来丰富熵模型的输入。通过建立传播链，该模型可以利用多个帧的潜在表示之间的相关性。所提出的双空间先验不仅以并行友好的方式将空间上下文的范围扩大了一倍，还进一步挤压了信道维度上的冗余。此外，我们的熵模型还能生成空间信道量化步骤。这种内容自适应量化机制有助于编解码器很好地应对各种视频协议。作为多粒度量化的核心要素，它不仅有助于实现平滑的速率调整，还能通过动态比特分配提高最终的 RD 性能。与使用最高压缩比配置的最佳传统编解码器 VTM 相比，在 UVG 数据集上平均节省了 18.2% 的比特率，这是一个重要的里程碑。

## 参考文献


[1] 2022.HM-16.20. https://vcgit.hhi.fraunhofer.de/jvet/HM/.访问日期：2022-03-02。

[2] 2022. PyTorchVideoCompression. https://github.com/ZhihaoHu/PyTorchVideoCompression.访问日期：2022-03-02。

[3] 2022.超视频组测试序列。http://ultravideo.cs.tut.fi。访问： 2022-03-02。

[4] 2022.VTM-13.2. https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/。Accessed: 2022-03-02.

[5] 2022.x265。https://www.videolan.org/developers/x265.html。访问时间： 2022-03-02.

[6] Eirikur Agustsson、 David Minnen、 Nick Johnston、 Johannes Balle、 Sung Jin Hwang 和 George Toderici。2020.用于端到端优化视频压缩的规模空间流。*IEEE/CVF 计算机视觉 和模式识别会议论文集*》。8503-8512.

[7] Johannes Ballé、Valero Laparra 和 Eero P. Simoncelli。2017.端到端优化图像压缩。In *5th International Conference on Learning Represen- tations, ICLR 2017*.

[8] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston.2018.使用尺度超优先级的变异图像压缩。*6th Interna- tional Conference on Learning Representations, ICLR* (2018).

[9] Gisle Bjontegaard.2001.RD- 曲线之间平均 PSNR 差异的计算。*VCEG-M33* (2001)。

[10] Benjamin Bross、Ye-Kui Wang、Yan Ye、Shan Liu、Jianle Chen、Gary J Sullivan 和 Jens-Rainer Ohm。2021.多功能视频编码（VVC）标准及其应用概述》。*IEEE 视频电路与系统论文 Technology* 31, 10 (2021), 3736-3764.

[11] Kelvin CK Chan、 Xintao Wang、 Ke Yu、 Chao Dong 和 Chen Change Loy。2021.BasicVSR: The search for essential components in video super-resolution and beyond.In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.4947-4956.

[12] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto.2020.使用离散高斯混合似然和注意力模块的学习图像压缩。*IEEE/CVF 计算机视觉和 模式识别会议论文集*》。7939-7948.

[13] Ze Cui, Jing Wang, Shangyin Gao, Tiansheng Guo, Yihui Feng, and Bo Bai.2021.具有连续速率适应性的非对称增益深度图像压缩。*IEEE/CVF 计算机视觉与模式识别大会论文集*》。10532-10541.

[14] Abdelaziz Djelouah、 Joaquim Campos、 Simone Schaub-Meyer 和 Christopher Schroers。2019.用于视频编码的神经帧间压缩。 *IEEE/CVF 计算机视觉国际会议（ICCV）论文集*。

[15] Amirhossein Habibian、Ties van Rozendaal、Jakub M Tomczak 和 Taco S Cohen。2019.使用速率失真自动编码器的视频压缩。 *IEEE/CVF 计算机视觉国际会议论文集*》。7033-7042.

[16] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin.2021.用于高效学习图像压缩的棋盘式上下文模型。In *Pro- ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.14771-14780.

[17] 何开明、张翔宇、任绍清、孙健。2016.图像识别的深度残差学习。In *Proceedings of the IEEE conference on computer vision and pattern recognition*.770-778.

[18] Jie Hu, Li Shen, and Gang Sun.2018.挤压与激发网络。In *Proceed- ings of the IEEE conference on computer vision and pattern recognition*.7132-7141.

[19] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu.2020.通过分辨率自适应流编码改进深度视频压缩。 *欧洲计算机视觉会议*。Springer, 193-209.

[20] Zhihao Hu, Guo Lu, and Dong Xu.2021.FVC：实现特征空间深度视频压缩的新框架。*计算机视觉与模式识别（CVPR）IEEE/CVF 会议 论文集*。1502-1511.

[21] Cong Huang, Jiahao Li, Bin Li, Dong Liu, and Yan Lu.2022.基于神经压缩的视频还原特征学习》。*IEEE/CVF 计算机视觉与模式识别（CVPR）会议论文集*》。5872-5881.

[22] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy.2020.轻量级光流 CNN--重新审视数据保真度和正则化。*IEEE patterns analysis and machine intelligence* 43, 8 (2020), 2555-2569.

[23] A Burakhan Koyuncu, Han Gao, and Eckehard Steinbach.2022.Contextformer：用于学习图像压缩中上下文建模的空间通道注意变换器。 *arXiv 预印本 arXiv:2203.02452* (2022).

[24] Théo Ladune、 Pierrick Philippe、 Wassim Hamidouche、 Lu Zhang 和 Olivier Déforges。2020.基于学习的视频编码光流和模式选择》。*第 22 届 IEEE 多媒体信号处理国际研讨会*。

[25] Théo Ladune、Pierrick Philippe、Wassim Hamidouche、Lu Zhang 和 Olivier Déforges。2021.实用学习型 视频编码的条件编码和可变比特率。*CLIC 研讨会，CVPR*（2021 年）。

[26] Théo Ladune、Pierrick Philippe、Wassim Hamidouche、Lu Zhang 和 Olivier Déforges。2021.灵活学习视频压缩的条件编码。*神经压缩：从信息论到应用--工作坊@.....*

*ICLR*.

[27] Jiahao Li, Bin Li, and Yan Lu.2021.深度上下文视频压缩。*神经信息处理系统进展* 34 (2021).

[28] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu.2020.M-LVC：用于学习视频压缩的多帧预测。*IEEE/CVF 计算机视觉与模式识别会议论文集*》。

[29] Haojie Liu, Ming Lu, Zhan Ma, Fan Wang, Zhihuang Xie, Xun Cao, and Yao Wang.2020.使用多尺度运动补偿和时空上下文模型的神经视频编码。*电气和电子工程师学会电路与系统 视频技术论文集*（2020 年）。

[30] Haojie Liu, Han Shen, Lichao Huang, Ming Lu, Tong Chen, and Zhan Ma.2020.通过联合时空相关性探索学习视频压缩。*AAAI 人工智能大会论文集*，第 34 卷。11580-11587.

[31] Jerry Liu、Shenlong Wang、Wei-Chiu Ma、Meet Shah、Rui Hu、Pranaab Dhawan 和 Raquel Urtasun。2020.用于高效视频通信的条件熵编码 pression. *欧洲计算机视觉会议*。Springer, 453-468.

[32] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao.2020.内容自适应和误差传播感知深度视频 压缩。*欧洲计算机视觉会议*。Springer, 456-472.

[33] 郭璐、欧阳万里、董旭、张晓云、蔡春雷、高志勇。2019.DVC：端到端深度视频压缩框架。*IEEE/CVF 计算机视觉与模式识别大会论文集*》。11006-11015.

[34] Guo Lu、Xiaoyun Zhang、Wanli Ouyang、Li Chen、Zhiyong Gao 和 Dong Xu。2020.用于视频压缩的端到端学习框架。*IEEE transactions on pattern analysis and machine intelligence* (2020).

[35] Alexandre Mercat、Marko Viitanen 和 Jarno Vanne。2020.UVG 数据集：用于视频编解码器分析和开发的 50/120fps 4K 序列。*第 11 届 ACM 多媒体系统会议论文集*。297-302.

[36] David Minnen、Johannes Ballé 和 George D Toderici。2018.用于学习图像压缩的联合自回归和分层先验。*神经 信息处理系统进展* 31 (2018).

[37] Jorge Pessoa、Helena Aidos、Pedro Tomás 和 Mário AT Figueiredo。2020.使用时空自动编码器的视频压缩端到端学习。In *2020 IEEE Workshop on Signal Processing Systems (SiPS)*.IEEE, 1-6.

[38] Anurag Ranjan and Michael J Black.2017.使用空间金字塔网络进行光流估计。*电气和电子工程师学会计算机视觉和 模式识别会议论文集*》。4161-4170.

[39] Oren Rippel、Alexander G Anderson、Kedar Tatwawadi、Sanjay Nair、Craig Lytle 和 Lubomir Bourdev。2021.ELF-VC: Efficient Learned Flexible-Rate Video Coding.*IEEE/CVF 国际计算机 Vision (ICCV) 会议论文集*》。14479-14488.

[40] Oren Rippel、Sanjay Nair、Carissa Lew、Steve Branson、Alexander G Anderson 和 Lubomir Bourdev。2019.学习视频压缩。*IEEE/CVF 计算机视觉国际会议论文集*》。3454-3463.

[41] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu.2021.用于学习视频压缩的时态上下文挖掘》。*arXiv 预印本 arXiv:2111.13850* (2021).

[42] 孙文宇、唐晨、李维贵、袁竹青、杨华中、刘永攀。2020.采用帧Conv3D和多帧差分调制的高质量单模型深度视频压缩。*欧洲 计算机视觉会议（ECCV）*。Springer, 239-254.

[43] Lucas Theis、Wenzhe Shi、Andrew Cunningham 和 Ferenc Huszár.2017.使用压缩自动编码器进行有损图像压缩。*第五届学习表征国际会议，ICLR*（2017）。

[44] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo.2016.MCL-JCV：基于 JND 的 H. 264/AVC 视频质量评估数据集。In *2016 IEEE International Conference on Image Processing (ICIP)*.IEEE，1509-1513。

[45] Zhou Wang, Eero P Simoncelli, and Alan C Bovik.2003.用于图像质量评估的多尺度结构模拟相似性。In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398-1402.

[46] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl.2018.通过图像插值实现视频压缩。*欧洲 计算机视觉会议（ECCV）论文集*。416-431.

[47] Xide Xia 和 Brian Kulis.2017.W-net：用于完全无监督 图像分割的深度模型。*arXiv preprint arXiv:1711.08506* (2017)。

[48] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman.2019.面向任务流的视频增强。*国际计算机期刊 Vision (IJCV)* 127, 8 (2019), 1106-1125.

[49] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte.2021.利用递归自动编码器和递归概率 模型学习视频压缩》。*IEEE 信号处理选题期刊》* 15，2 (2021)，388-401。

[50] Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt.2021.神经视频压缩的分层自回归建模。*第九届国际学习表征会议，ICLR*（2021）。

# 附录

附录为我们提出的用于神经视频通信的时空混合熵模型提供了补充材料。

## A 网络架构

**上下文编码器和解码器。**上下文编码器和解码器的网络设计如图 7 所示。对于编码器
输入是当前原始帧$x_t$ 和多尺度
上下文$C_t^0, C_t^1, C_t^2$在不同的分辨率下（原始分辨率、2 倍缩放分辨率
-分辨率
质，4 倍降采样），通道 64。输出是潜在表示$y_t$ ，通道 96，16 倍降采样分辨率。对于解码器，输入包含解码后的潜表征$\hat{y}_t$。以$C_t^1$ 和$C_t^2$为条件，高分辨率特征$F_t$ 与
通道 32 被解码。对于图 7 中的卷积层和子像素卷积层，（$Cin, Cout, S$）表示内核大小、输入通道数、输出通道数和跨距、
分别为图 7 中的瓶颈残差块来自文献 [41]。其中卷积层的核大小和步长分别为 3 和 1。因此，我们只显示了瓶颈残差块的输入和输出通道编号。

**熵模型。**图 8 显示了熵模型的详细网络结构。输入包括通道 192 的超先验、通道 192 的时间上下文先验、通道 96 的潜先验（即上一帧的解码潜表示）。如图 9 所示，时间上下文先验由时间上下文编码器生成。超先验编码器和解码器的网络结构如图 10 所示。在第一步编码过程中，熵模型不仅能估算出帧数据概率分布的均值和标度值，还能估算出帧数据的熵值。$y_{t,(i+j)\%2=0,k<\frac{C}{2}}$和$y_{t,(i+j)\%2=1,k\geq\frac{C}{2}}$，同时也产生量化步骤 at 空间信道-e。
在第二步编码过程中，上一步编码的潜在表征与输入融合，然后熵模型计算出概率分布的均值和标度值。
for $y_{t,(i+j)\%2=1,k<\frac{C}{2}}$ and $y_{t,(i+j)\%2=0,k\geq\frac{C}{2}}$

**运动矢量编码器和解码器**编码器和解码器
图 11 展示了运动矢量编码器的工作原理。对于编码器来说，输入是通道 2 的原始运动矢量。输出是 16 倍降采样分辨率的潜在表示，通道为
64.解码器采用反向结构。$mv\_y_t$ 的多粒度量化和熵模型与 $y_t$ 的相同。唯一不同的是熵模型输入。对于运动矢量
熵模型的输入是相应的超先验和潜先验。图 11 中的下采样和上采样残差块来自 [41]。

**帧生成器。**与 DCVC [27] 和 [41] 仅使用普通残差块不同，我们建议使用基于 W-Net [47] 的结构来构建帧生成器。W-Net 可以有效扩大模型的感受野，从而提高模型的生成能力。图 14 展示了帧生成器的详细网络结构。输入包括高分辨率的
特征$F_t$ ，通道 32 和上下文$C_t^0$ ，原始分辨率
通道 64。输出是最终的重建帧和
下一帧使用的特征。在图 14 中，有两个相同的

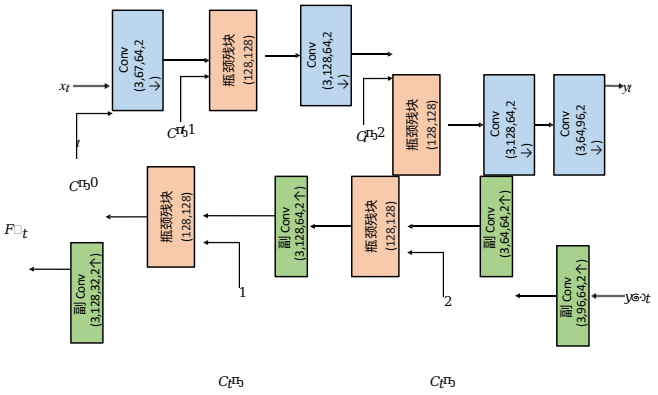



**图 7：上下文编码器和解码器的结构。**

通过 U-Net 建立 W-Net。图 12 展示了 U-Net 中受关注的残差块。

**神经图像编解码器。**由于我们的目标是用单一模型处理多种速率，因此我们也需要一个支持这种内部编码能力的神经图像编解码器。我们的神经图像编解码器结构如图 13 所示。相应的多粒度量化
zation和熵模型与$y_t$ 相似。唯一不同的是熵模型的输入。神经图像编解码器
熵模型的输入只包括超先验。在图 13 中，我们也使用了一个 U-Net 来提高神经图像编解码器的生成能力，其网络结构与图 14 类似。如表 8 所示，我们的图像编解码器的压缩率与 Cheng 2020 [12]相当。

## B 传统编解码器的设置

对于 x265 [5]，我们使用 *veryslow* 预设值。我们还比较了 HM-16.20[1]和 VTM-13.2 [4]，它们代表了最佳编码器的
分别为 H.265 和 H.266。对于 HM 和 VTM，使用压缩率最高的低延迟配置。使用 4 个参考帧和 10 位内部比特深度（HM 在 *encoder_lowdelay_main_rext.cfg* 中设置，VTM 在 *encoder_lowdelay_vtm.cfg* 中设置）。文献[41]中的比较表明，虽然最终失真度是在 RGB 域测量的，但使用 YUV 444 作为内部色彩空间可以提高压缩率。因此，我们也采用了这种设置。有关编解码器设置的更多研究，请参阅 [41]。
x265、HM 和 VTM 的详细设置如下：

- **x265**
  ffmpeg
  -像素_格式 yuv420p
  -s {宽度}x{高度}
  -帧频 {帧频}
  -*输入*文件名
  -vframes {帧号}
  -c:v libx265
  -预设速度很慢
  -调整零延迟
  -x265-params “qp={qp}:keyint=32:csv-

```
        log-level=1:
        csv={csv_path}:verbose=1:psnr=1"
        {输出视频文件名称}
```

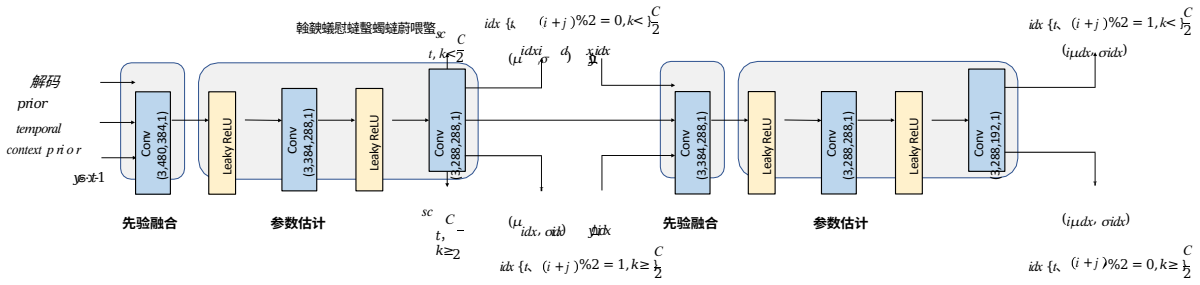



**图 8：熵模型的结构。**

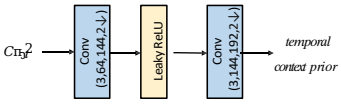



**图 9：时态上下文编码器的结构。**

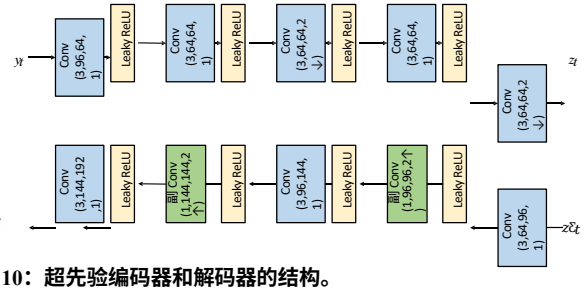



**图 10：超先验编码器和解码器的结构。**

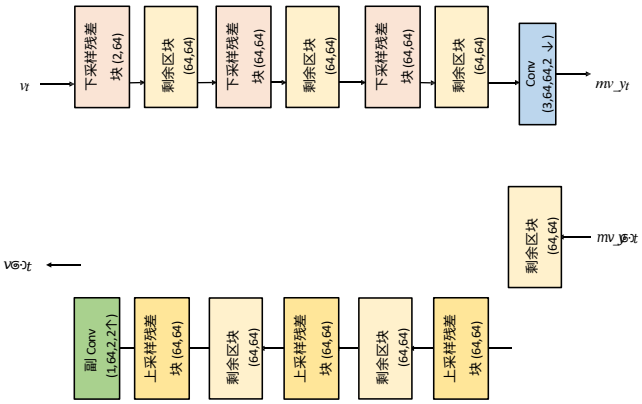



**图 11：运动矢量编码器和解码器的结构。**

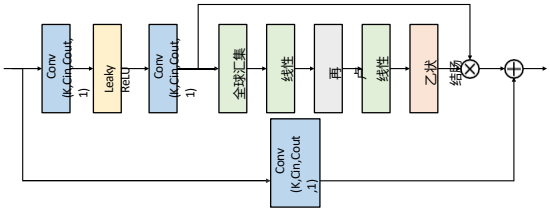



**图 12：关注残差区块的结构。**

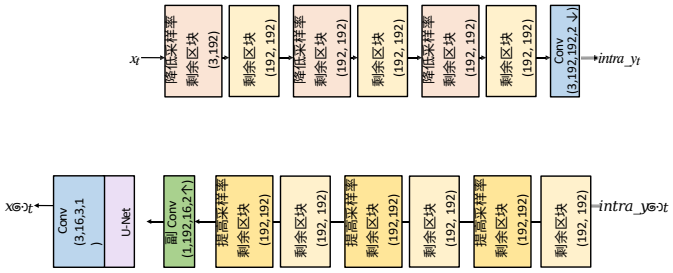



**图 13：神经图像编解码器的结构。**

- **HM**
  TAppEncoder
  -c encoder_lowdelay_main_rext.cfg
  --输入*文件*={*输入*文件名}.
  --输入位深度=8
  --输出位深度=8
  --输出位深度C=8
  --输入色度格式=444
  --*FrameRate*={帧频}
  --DecodingRefreshType=2解码刷新类型
  --*FramesToBeEncoded*={帧号}
  --SourceWidth={width}（源宽度
  --SourceHeight={height}源高度
  --内周期=32
  --QP={qp}。
  --级别=6.2
  --*BitstreamFile*={比特流文件名}.
- **VTM**
  编码器应用程序
  -c encoder_lowdelay_vtm.cfg
  --*输入文件*={输入文件名}.
  --*BitstreamFile*={比特流文件名}.
  --DecodingRefreshType=2解码刷新类型
  --输入位深度=8
  --输出位深度=8
  --输出位深度C=8
  --输入色度格式=444

--*FrameRate*={帧频}

--*FramesToBeEncoded*={帧号}

--SourceWidth={width}（源宽度

--SourceHeight={height}源高度

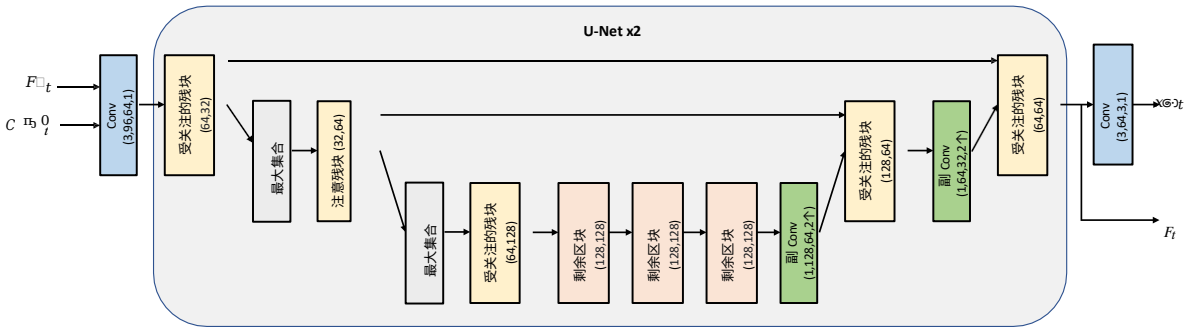



**图 14：帧发生器的结构。**

--内周期=32

--QP={qp}。

--级别=6.2

## C 速率-失真曲线

图 15 和图 16 显示了每个数据集的 RD（速率-失真）曲线，包括 PSNR 和 MS-SSIM 结果。从这些结果可以看出，我们的编解码器可以在很宽的速率范围内实现 SOTA 压缩比。

## D 不同内部时间设置下的比较

之前的工作 Sheng 2021 [41] 表明，周期内 12 对压缩率有害，在实际应用中很少使用。例 如，与周期内 12 相比，周期内 32 对 HM 平均可节省 23.8%的比特率（见 [41] 补充材料中的表 3）。因此，为了更贴近实际应用场景，我们效仿文献 [39]，在本文中也使用了周期内 32。但同样重要的是，要评估我们的编解码器在周期内 12.表 9 显示了 PSNR 的相应 BD 速率（%）比较。我们注意到，由于 VTM 的 GOP 大小为 *encoder_lowdelay_vtm* 配置为 8，不支持周期内 12。因此，我们对 VTM-13.2* 的配置如下 [41]，并将其作为锚点（VTM- 13.2* 的更多配置细节请参阅 [41]）。我们可以看到，在 12 期间内，我们的编解码器也明显优于之前所有的 SOTA 神经编解码器和传统编解码器。

## E 视觉比较

我们还与之前的 SOTA 神经编解码器和传统视频编解码器进行了直观比较。图 17 和图 18 显示了两个例子。从这些比较中我们可以发现，我们的编解码器可以在不增加比特率成本的情况下实现更高的重构质量。例如，在图 17 所示的例子中，我们可以发现之前的神经编解码器重建的帧有明显的色彩失真。相比之下，我们的编解码器能更准确地还原条纹色彩。在图 18 中，我们的编解码器可以生成更清晰的纹理细节。

## f 其他细节

我们的编解码器建立在之前开发的 DCVC [27] 及其改进作品 Sheng 2021 [41] 的基础上。与 Sheng 2021 [41]相比，我们重复使用了运动估计、运动矢量编码器/解码器、临时上下文挖掘和上下文编码器/解码器。熵模型、量化机制和帧生成器

进行了重新设计。此外，我们还分别增加了量化器和 MEMC（运动估计和运动补偿）过程的更多细节。

**量化器**量化器的工作原理如下

- 首先，潜在表示被量化步骤（QS）分割。
- 其次，减去平均值。
- 第三，将潜表征四舍五入为最接近的整数，并通过算术运算将其转换为比特流。

编码器。

在解码过程中，首先由算术解码器解码舍入的潜在表示，并将平均值加回来。最后，再乘以 QS。在这一过程中，现有的图像和视频编解码器大多使用固定 QS，即 1。首先，全局 QS 由用户为特定目标速率设定。然后，由于不同信道包含的信息的重要性不同，所以要根据信道的 Q S 进行缩放。最后，应用我们的熵模型生成的空间-信道-QS。这可以帮助我们的编解码器应对各种视频内容，并在每个位置实现精确的速率调整。

**运动估计和运动补偿**。MEMC 过程可分为以下几个步骤：

- 首先，当前帧和参考帧之间的原始运动矢量（MV）是通过光学矢量估算出来的。

  流量估算网络。MV 在全分辨率下非常密集。
- 然后通过 MV 编码器将原始 MV 转换为潜在表示。
- MV 的潜在表示被量化并转换成比特流。其中的量化步骤也去掉了

  在多粒度级别上确定。量化和熵模型与当前帧的潜在代表相似。
- 然后通过算术重建全分辨率的 MV
  解码器和 MV 解码器。

**表 8：图像编解码器 BD-Rate（%）与 PSNR 的比较。**

| | UVG | MCL-JCV | HEVC B | HEVC C | HEVC D | HEVC E | HEVC RGB | 平均 |
|---|---|---|---|---|---|---|---|---|
| 程 2020 图像编解码器 [12] | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 我们的图像编解码器 | -0.9 | 0.2 | -1.9 | -0.2 | 2.5 | 1.9 | -1.2 | 0.1 |

**表 9：12 期间内 PSNR 的 BD-Rate（%）比较。**

| | UVG | MCL-JCV | HEVC B | HEVC C | HEVC D | HEVC E | HEVC RGB | 平均 |
|---|---|---|---|---|---|---|---|---|
| VTM-13.2* | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| HM-16.20 | 8.3 | 15.3 | 20.1 | 13.3 | 11.6 | 22.4 | 13.2 | 14.9 |
| x265 | 97.4 | 99.2 | 89.1 | 49.5 | 44.2 | 79.5 | 94.5 | 79.1 |
| DVCPro [34] | 54.8 | 63.0 | 67.3 | 70.5 | 47.2 | 124.2 | 50.8 | 68.3 |
| 区域联络中心 [49] | 74.0 | 103.8 | 87.2 | 86.1 | 53.0 | 98.7 | 66.2 | 81.3 |
| MLVC [28] | 39.9 | 65.0 | 57.2 | 99.5 | 75.8 | 84.1 | 64.3 | 69.4 |
| DCVC [27] | 21.0 | 28.1 | 32.5 | 44.8 | 25.2 | 66.8 | 22.9 | 34.5 |
| 盛 2021 [41] | -20.4 | -1.6 | -1.0 | 15.7 | -3.4 | 7.8 | -13.7 | -2.4 |
| 我们的 | -38.4 | -24.2 | -19.9 | -10.5 | -23.9 | -18.7 | -34.2 | -24.3 |

- 重构后的 MV 可用于扭曲特征，并通过时态上下文挖掘模块
  提取上下文。

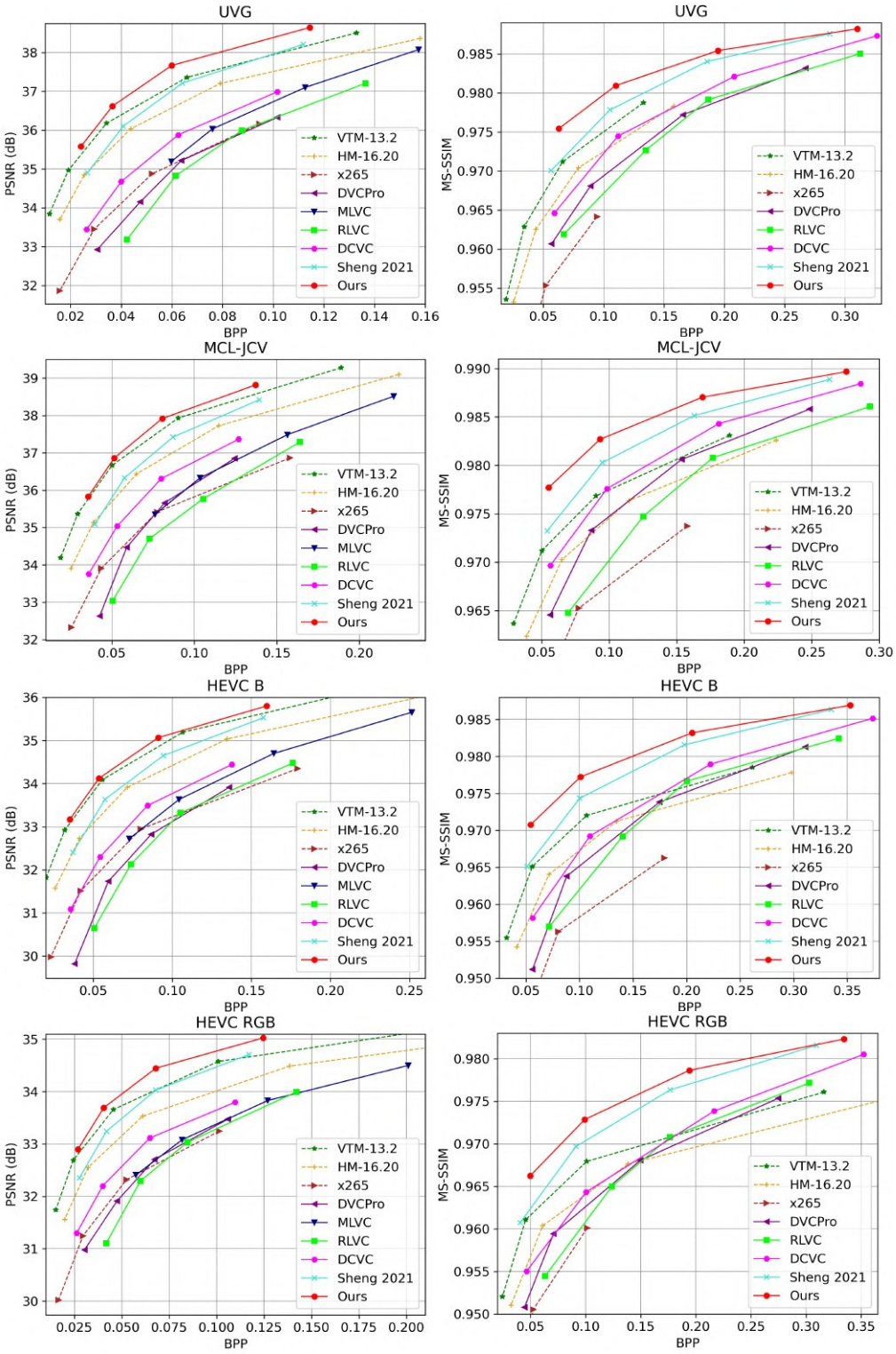



**图 15：UVG、MCL-JCV、HEVC B 和 RGB 的 RD 曲线。**

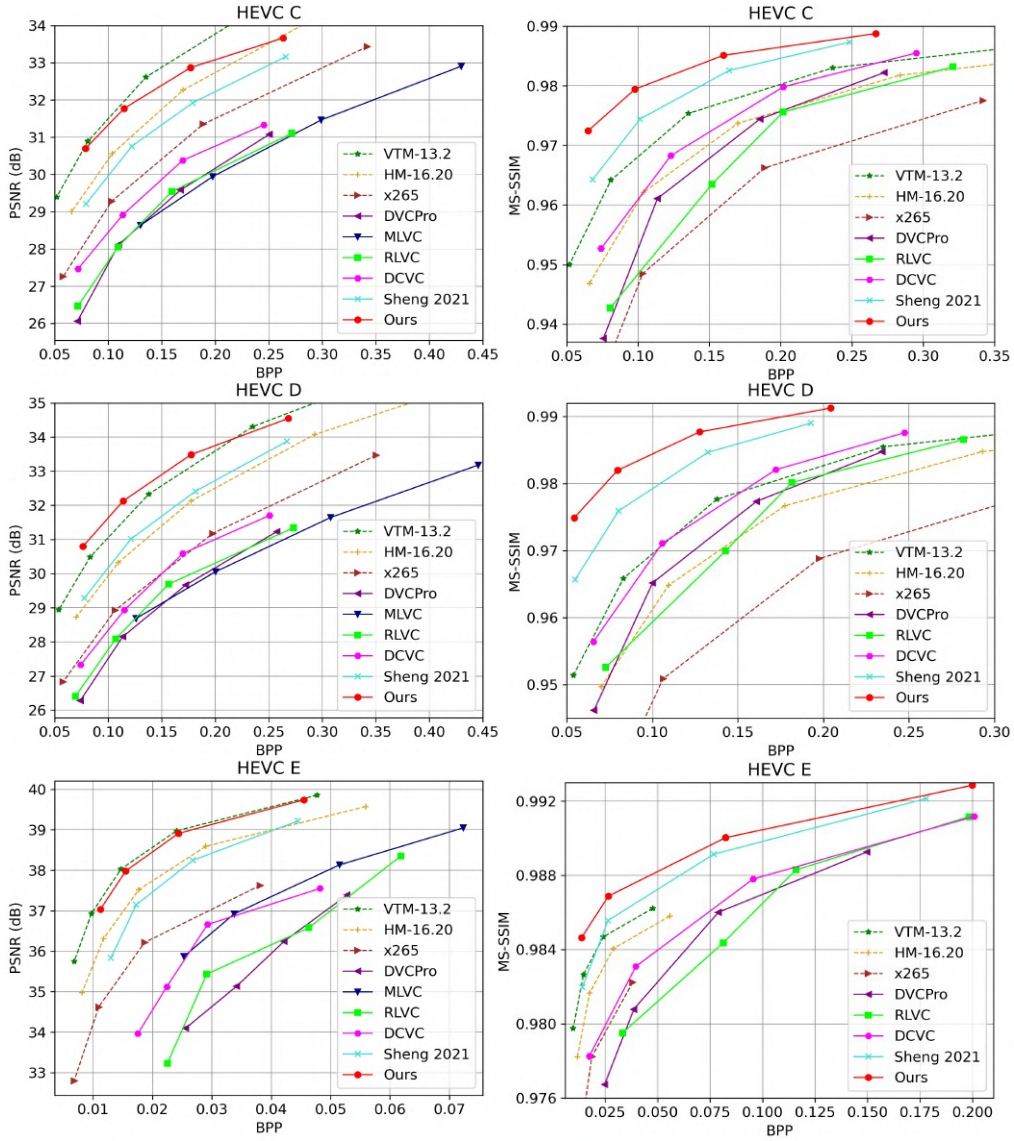
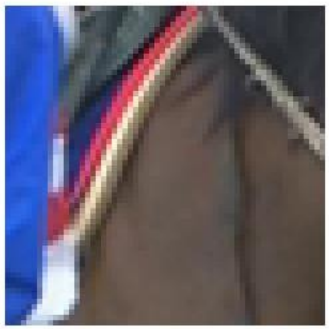
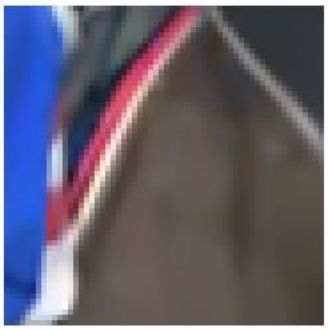



**图 16：HEVC C、D 和 E 的 RD 曲线。**

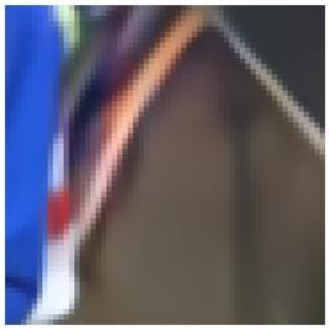


原件：(BPP/PSNR)

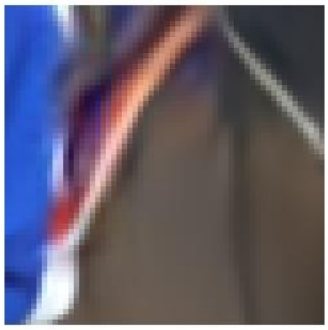
小时： 0.159/29.59



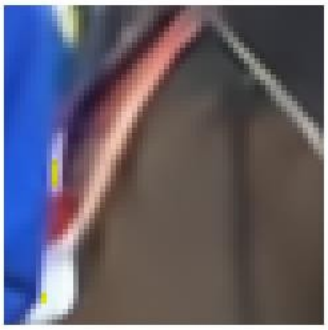
VTM: 0.115/29.49



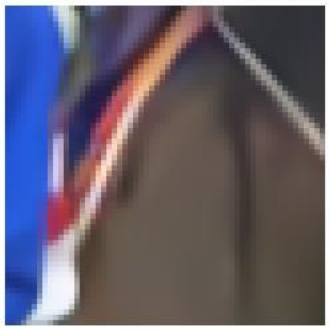
DVCPro: 0.150/28.50



RLVC: 0.151/28.21



多氯联苯： 0.210/28.98

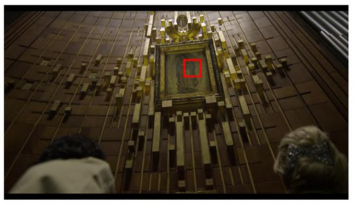


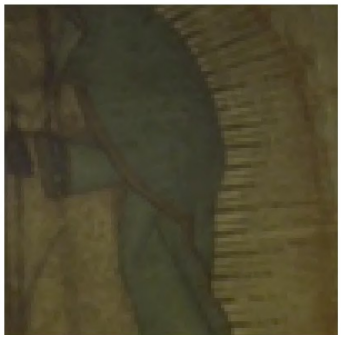
DCVC: 0.154/29.44

盛 2021: 0.115/29.57

我们的0.112/30.54

**图 17：HEVC D 数据集中的 *RaceHorses*。**

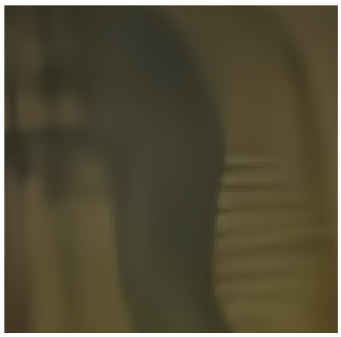




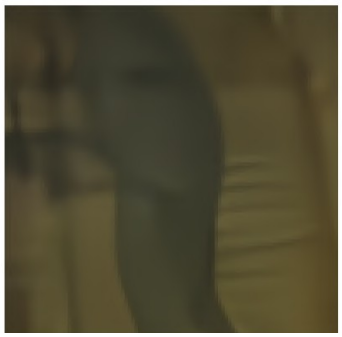
原件：(BPP/PSNR)

HM: 0.015/37.73

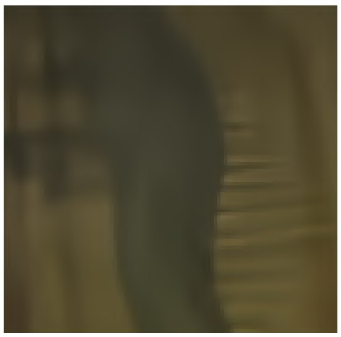
VTM: 0.013/38.34



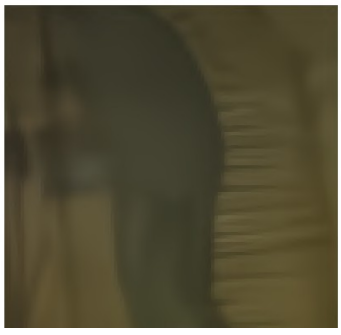
DVCPro: 0.016/34.86



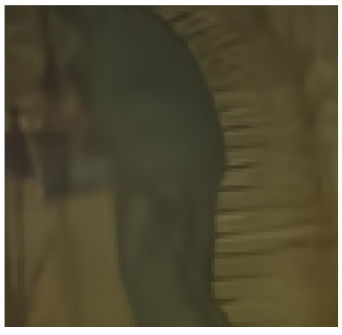
RLVC: 0.028/34.45



多氯联苯： 0.033/36.10



DCVC: 0.016/36.17



盛 2021: 0.015/37.83

我们的0.014/38.54

**图 18：MCL-JCV 数据集中的 *videoSRC03*。**