EVC：通过 MaskDecay 实现实时神经图像压缩

王国华[1]*, **Jiahao Li**[2], **Bin Li**[2],**YanLu**[21]南京大学新软件技术国家重点实验室[2]微软亚洲研究院 wangguohua@lamda.nju.edu.cn, {li.jiahao,libin,yanlu}@microsoft.com

arXiv:2302.05071v1 [eess.IV] 10 Feb 2023

摘要

神经图像压缩在率失真（RD）性能方面已超越最先进的传统编解码器（H.266/VVC），但其复杂度较高，且针对不同率失真权衡需要使用不同模型。本文提出一种高效单模型可变码率编解码器（EVC），在输入图像为 768x512 时可达到 30 帧/秒，且率失真性能仍优于 VVC。通过进一步降低编码器和解码器的复杂度，我们的小型模型甚至能在输入图像为 1920x1080 时达到 30 帧/秒。为弥合不同容量模型之间的性能差距，我们精心设计了掩码衰减机制，可自动将大型模型的参数转换为小型模型的参数。我们还提出一种新颖的稀疏正则化损失，以缓解 L_p 正则化的不足。我们的算法显著缩小了性能差距：中型模型和小型模型分别缩小了 50% 和 30%。最后，我们倡导采用可扩展编码器进行神经图像压缩。编码复杂度可动态调整，以满足不同的延迟需求。我们提出多次衰减大型编码器，以逐步缩减残差表示。掩码衰减和残差表示学习都大幅提升了可扩展编码器的率失真性能。代码见 https://github.com/microsoft/DCVC。

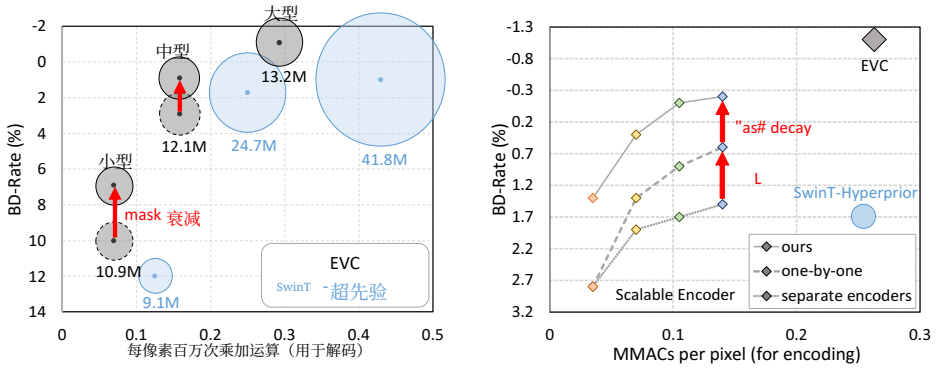
1 引言

基于深度学习的图像压缩相比传统编解码器（H.266/VVC）展现出了卓越的率失真（RD）性能（Bross 等人，2021）。然而，两个主要问题限制了其在实际应用中的可行性。其一是复杂度高。大多数最先进的神经图像编解码器都依赖复杂的模型，例如大容量骨干网络（Zhu 等人，2022年；Zou 等人，2022年）、复杂的概率模型（Cheng 等人，2020年）以及不利于并行化的自回归模型（Minnen 等人，2018年）。高复杂度很容易导致实际应用中的延迟无法令人满意。第二个问题是码率控制效率低下。针对不同的率失真权衡，需要训练并存储多个模型。在推理过程中，还需要根据目标质量加载特定模型。出于实际应用的考虑，人们希望单个模型能够处理可变的率失真权衡。

本文尝试同时解决这两个问题，构建单一的实时神经图像压缩模型，同时其率失真性能仍可与其他最先进模型媲美。受近期进展的启发，我们设计了一个高效框架，配备深度卷积块（Liu 等人，2022年）和空间先验（He 等人，2021；Li 等人，2022a；Qian 等人，2022年）。为实现可变的率失真权衡，我们针对表示引入了可调量化步长（Chen & Ma，2020年；Li 等人，2022a）。我们框架中的所有模块都高度高效且对图形处理器友好，这不同于近期基于 Transformer 的模型（Zou 等人，2022年）。编码和解码（包括算术编码）对于 768×512 输入可实现 30 帧/秒。与其他最先进模型相比，我们的方法具备相当的率失真性能、低延迟，并且单一模型即可适用于所有率失真权衡。

为了进一步加速模型，我们降低了编码器和解码器的复杂度。我们提出了三个序列模型：大型、中型和小型（参见附录表3）。值得注意的是，我们的小型

*本研究是在王国华于微软亚洲研究院担任全职实习生期间完成的。



(a) (b)

图1：Kodak 上 BD-Rate 与复杂度之间的权衡。锚点为 VTM。(a) 和 (b) 展示了我们的方法通过掩码衰减和残差表示学习（RRL）带来的性能提升。我们引用了 SwinT-Hyperprior（Zhu 等人，2022）的结果进行比较。

模型甚至能以 30 帧/秒处理 1920×1080 输入。近期研究（Zhu 等人，2022年）指出，单纯降低模型容量会导致性能显著下降。然而，这一关键问题尚未得到有效解决。本文主张利用教师模型训练小型神经压缩模型，以缓解这一问题。具体而言，我们提出了**掩码衰减**，以自动将大型模型的参数转化为小型模型。具体来说，首先在预训练教师模型中插入掩码层。随后逐渐稀疏化这些掩码，直至所有层都转变为学生结构。最后，将掩码合并到原始层中，同时保持网络功能不变。对于稀疏正则化，大多数先前工作（Li 等人，2016年；Ding 等人，2018年；Zhang 等人，2021）采用基于 L_p-范数的损失。但这类方法在神经图像压缩中效果不佳（参见图 7）。本文提出一种新颖的稀疏正则化损失，以缓解 L_p 的缺点，从而更有效地优化掩码。在大型模型的帮助下，我们的中型和小型模型分别提升了 50% 和 30%（参见图 1a）。这证明了掩码衰减的有效性，也表明复用大型模型的参数有助于训练小型神经图像压缩模型。

此外，考虑到实际编解码器应用中设备能力各不相同，仅使用一个解码器来支持不同编码复杂度的编码器可伸缩性也是一项关键需求。为此，我们提出对繁杂的编码器进行多次压缩，以逐步弥合性能差距。残差表示学习（RRL）和掩码衰减都隐式地将繁杂的编码器视为参考，并鼓励不同小型编码器之间的多样性。因此，我们的方法比单独训练多个编码器取得了更优的性能（参见图1b）。与受可伸缩网络（Yu 等人，2019）启发的 SlimCAE（Yang 等人，2021）相比，我们的方法框架更简单，率失真性能也更好。

我们的贡献如下。

- 我们提出了一种用于图像压缩的高效可变码率编解码器（EVC）。它仅需一个模型即可实现不同的率失真权衡。我们的模型处理 768×512输入时可达到 30 帧/秒，同时率失真性能可与其他最先进模型相媲美。我们的小型模型处理 1920×1080 输入时甚至也能达到 30 帧/秒。
- 我们提出了掩码衰减，这是一种借助教师模型改进学生图像压缩模型的有效方法。此外，我们还引入了一种新颖的稀疏正则化损失，以缓解 L_p 正则化的不足。得益于掩码衰减，我们的中型和小型模型分别提升了 50% 和 30%。
- 我们实现了神经图像压缩的编码可扩展性。通过残差表示学习和掩码衰减，我们的可扩展编码器显著缩小了与教师模型之间的性能差距，并取得了优于先前 SlimCAE 的率失真性能。

2 相关工作

神经图像压缩 正处于蓬勃发展的阶段。Ball´e 等人（2017年）提出用加性独立同分布均匀噪声替代量化器，使神经图像压缩模型能够进行端到端训练。随后，Ball´e 等人（2018年）提出了超先验结构。

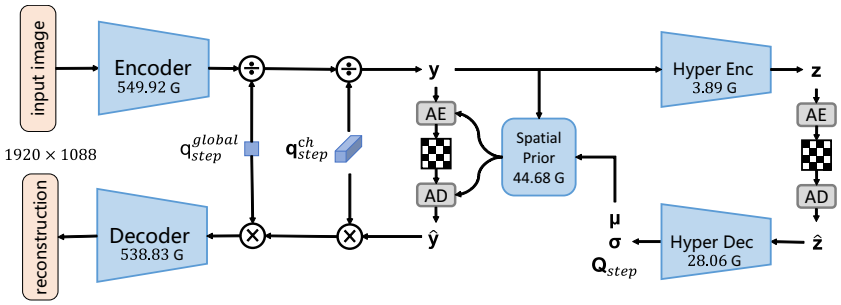


图 2：EVC 的整体框架。

此后，研究者提出了大量方法来改进熵模型和主干网络（包括编码器和解码器）。Minnen 等人（2018年）提出了自回归组件，Cheng 等人（2020年）提出了高斯混合模型，以改进熵模型中的概率估计。Informer（Kim 等人，2022年）提出利用注意力机制来挖掘全局和局部信息。近年来，研究者引入了更强大的主干网络（例如，INNs（Xie 等人，2021）、Transformer（Qian 等人，2022年）和 Swin（Zhu 等人，2022）），以替代编码器和解码器中的 ResNet（He 等人，2016年）。然而，近期最先进的神经图像编解码模型存在延迟高，以及针对每种率失真权衡都需要单独模型的问题。本文提出了一种高效的神经图像压缩框架，旨在实现实时编码和解码。特别是，我们的框架还可以通过可调量化步长处理可变率失真权衡（Chen & Ma，2020年；Cui 等人，2021；Li 等人，2022a）。我们模型的率失真性能与其他最先进的神经图像压缩模型相当，并且超过了最先进的传统图像编解码器（H.266/VVC）。

模型压缩和知识蒸馏（KD）是加速神经网络的两种常用技术。模型压缩，尤其是滤波器级剪枝（He 等人，2017年），旨在从庞大的网络中获得高效的网络结构。一些研究（Li 等人，2016年；Luo 和 Wu，2020年；He 等人，2019年；Molchanov 等人，2019年）会计算每个滤波器的重要性，并剪除不重要的滤波器。其他方法则在训练过程中直接对滤波器施加惩罚，将某些参数隐式地视为重要性指标（Liu 等人，2017年；Ding 等人，2021；Wang 和 Wu，2022年）。这些方法使用基于L_p-范数的稀疏损失（Ding 等人，2018年；Wen 等人，2016年；Zhang 等人，2021）。对于剪枝网络，知识蒸馏（KD）（Hinton 等人，2015年）通过模仿教师知识（例如逻辑值（Zhao 等人，2022年）或特征（Wang 等人，2022年））来提升性能。所有这些技术主要在语义任务中得到研究，却很少用于神经图像压缩。本文首次尝试借助大型模型改进小型神经图像压缩模型。我们认为，对于这项任务，复用教师滤波器是提升学生模型效果的有效方法。为此，我们提出了带有新型稀疏正则化损失的掩码衰减。借助所有提出的技术，我们的学生模型得到了显著改进。

3 方法

我们首先简要介绍高效框架。然后，我们提出三种复杂度各异的模型：大型、中型和小型。接下来，我们介绍如何借助大型模型（教师模型）提升小型模型（学生模型）。我们提出掩码衰减，使教师模型参数能够自动转换为学生模型参数。最后，我们提出可扩展编码器。通过残差表示学习和掩码衰减，逐步训练多个编码器，以弥合教师模型与其他模型之间的性能差距。

3.1 我们的 EVC 图像压缩方法

EVC 的框架如图2所示。给定一张图像，编码器会生成其表示$\boldsymbol{y}$。遵循 Chen & Ma（2020）；Li 等人（2022a），为了在单个模型中实现可变比特率，我们引入可调量化步长。 q^{global} 步长和 $\boldsymbol{q}_{\text{step}}^{\text{ch}}$ 是两个可学习参数，分别用于全局和按通道量化 $\boldsymbol{y}$ 。通过调整 q^{global} 步长 的大小，可以实现不同的率失真权衡。更多细节见附录A节。

图2还展示了每个模块处理 1920×1088 输入图像时的乘加运算次数。显然，大部分计算开销来自编码器和解码器。因此，我们专注于加速这

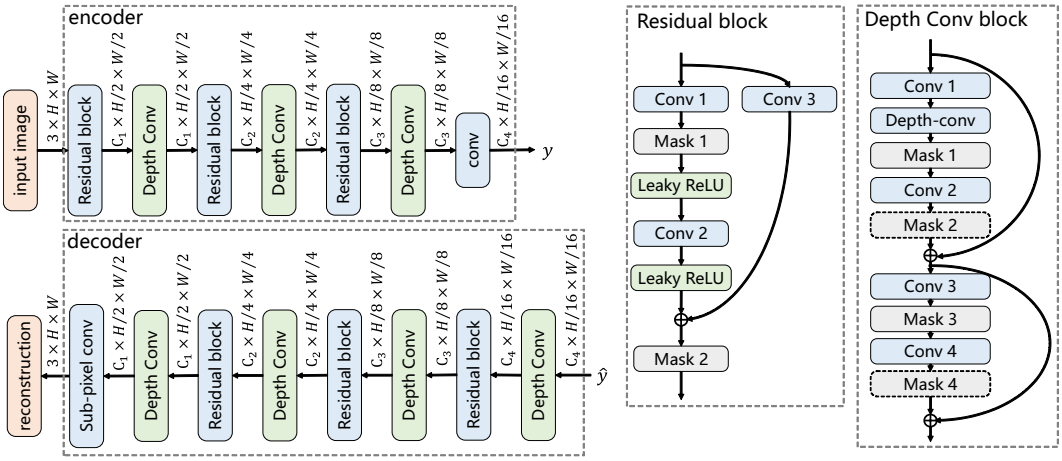
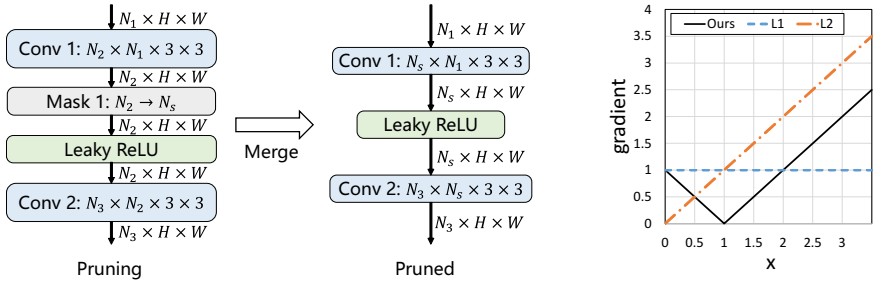


图3：我们的编码器和解码器架构。训练后，所有掩码层都会合并到卷积层中。为简化说明，我们省略了深度卷积块中的带泄露的ReLU。



(a) (b)

图4：（a）展示了我们的剪枝方法。掩码层插入在两个卷积层之间。通过将掩码层合并到这两个复杂的卷积层中，可将它们转换为高效的卷积层。（b）比较了不同稀疏损失函数的梯度 $\frac{\partial \mathcal{L}_{sparse}}{\partial x}$。

两个模块，同时保持其他部分的架构不变。图3展示了这两个模块的详细结构。与近期最先进模型（Zhu 等人，2022；Qian 等人，2022；Zou 等人，2022）不同，我们采用残差块和逐通道卷积块（Liu 等人，2022），而非 Transformer。因此，我们的方法更适合图形处理器，且延迟更低。 C_1、 C_2、 C_3和 C_4 是针对不同计算预算预先设定的数值。我们提出了三种模型，其通道数分别为 $[64, 64, 128, 192]$（小型）、$[128, 128, 192, 192]$（中型）和 $[192, 192, 192, 192]$（大型）。附录中的表3总结了各模型的参数数量和乘加运算次数。请注意，大型模型是我们计算开销较大的教师模型，而中型和小型模型是高效的学生模型。

3.2 通过掩码衰减改进学生模型

由于容量有限，学生模型从头开始训练时表现不佳（Zhu 等人，2022）。如何改进学生模型是神经图像压缩领域一个至关重要却鲜少研究的问题。本文认为，复用教师模型参数有所助益。为此，我们提出掩码衰减，以端到端方式转换参数。

通过掩码衰减压缩一层的表示。 如图 4a 所示，在两层之间插入一个掩码层，以减少表示的通道数（从 N_2 到 N_s）。该掩码层包含 N_2 个参数，用于剪除相应的通道。在前向计算中，表示与这些参数按通道相乘。训练期间，通过优化我们的稀疏损失，这些参数将变得稀疏。训练结束后，根据掩码剪除卷积层 #1 和卷积层 #2 中无用的参数。然后将掩码自身并入卷积层 #2，从而消除掩码。

为了使掩码层中的参数 $\boldsymbol{m}$ 稀疏化，我们设计了一种新颖的稀疏正则化损失。大多数以往的工作（Li 等人，2016年；Wen 等人，2016年；Zhang 等人，2021）采用基于 L1范数（$|\boldsymbol{m}|$）或 L2范数（$\frac{1}{2}\|\boldsymbol{m}\|_2$）的函数。请注意，在梯度下降过程中， $\boldsymbol{m}$通过 $sign(\boldsymbol{m})$ 更新，并且

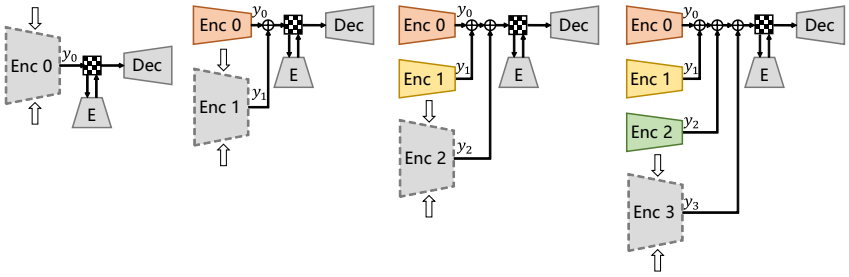


图 5：多次使用压缩编码器，逐步学习残差表示的示意图。“Enc”、“Dec”和“E”分别表示编码器、解码器和熵模块。

$\boldsymbol{m}$，分别。遗憾的是，这两种方法都不适用于神经图像压缩模型（参见图 7）。我们认为，对于取正值的参数，L1范数得到的梯度是一个常数，不考虑参数自身的大小。而对于 L2范数，当参数趋近于零时，梯度过小，无法使掩码稀疏化。此外，当参数为正时，这两种方法都不存在驻点。为了解决这些问题，我们从梯度的角度设计了一种新颖的稀疏正则化损失。如图 4b 所示，梯度定义为

$$\frac{\partial \mathcal{L}_{sparse}(x)}{\partial x} = |x-1|\,, \tag{1}$$

其中 $x \geq 0$。对于 $x > 1$，幅值较大的参数会受到较大的衰减梯度，这与 L2 范数类似。而对于 $0 \leq x < 1$，较小的参数则会受到较大的梯度，从而快速趋近于零。此外，在 $x = 1$ 处存在一个驻点，这意味着 $\frac{\partial \mathcal{L}_{sparse}(x)}{\partial x}\Big|_{x=1} = 0$。对于 L1 和 L2，$x > 0$ 始终会有一个正梯度。为了使模型稳定收敛，需要任务损失提供负梯度来平衡这个正梯度。得益于我们的稀疏损失，当 x 接近 1 时，只需要较小的负梯度。$\mathcal{L}_{sparse}$ 可以通过对公式 1 积分计算得出，即

$$\mathcal{L}_{sparse}(x) = \begin{cases} -\frac{1}{2}x^2 + x, & \text{if } 0 \leq x \leq 1\,, \\ \frac{1}{2}x^2 - x + 1, & \text{if } x > 1\,. \end{cases} \tag{2}$$

在训练中，受 AdamW（Loshchilov 与 Hutter，2018年）的启发，我们将 $\mathcal{L}_{sparse}$ 与任务损失 $\mathcal{L}_{RD}$ 解耦，并将其作为权重衰减。因此，任务损失不会显式受到我们的稀疏损失影响。所以我们将该算法命名为掩码衰减。对于掩码 $\boldsymbol{m}$，其更新公式为

$$\boldsymbol{m}_{t+1} = \boldsymbol{m}_t - \eta\frac{\partial \mathcal{L}_{sparse}}{\partial \boldsymbol{m}_t} - \gamma\frac{\partial \mathcal{L}_{RD}}{\partial \boldsymbol{m}_t} = \boldsymbol{m}_t - \eta|\boldsymbol{m}_t - 1| - \gamma\frac{\partial \mathcal{L}_{RD}}{\partial \boldsymbol{m}_t}\,, \tag{3}$$

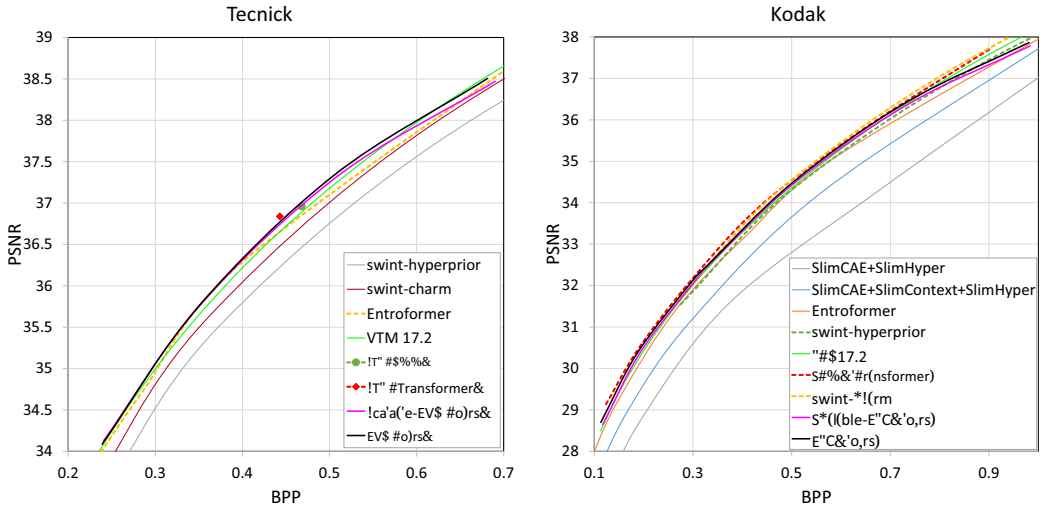
其中，γ 是学习率，η 是掩码衰减的码率。

编码器和解码器的掩码衰减。如图 3 所示，为了将教师模型转换为学生模型，我们会在必要时于某些位置插入掩码层。我们的原则是尽可能少地插入掩码层。在一个残差块中，分别插入掩码 #1 和 #2，以剪除残差连接内部和外部的通道。在深度卷积块中，掩码 #1 决定深度卷积层的通道，而掩码 #3 则剪除 Conv #3 和 #4 之间的表示。请注意，这两个块的输出通道是共享的，因此深度卷积块中的掩码 #2 和 #4 与残差块中的掩码 #2 共享权重。并且，我们将掩码 #2 和 #4 放置在残差相加之后，以确保恒等梯度反向传播。

使用掩码衰减进行训练。训练从一个预训练教师模型开始。随后，我们插入掩码层，并将其初始化为 1。通过最小化$\mathcal{L}_{RD} = R + \lambda D$来优化整个网络，其中 λ 用于调节码率 R 与失真 D之间的权衡。每次迭代前，我们都会对掩码进行衰减，这可能类似于权重衰减（Loshchilov 与 Hutter，2018年）。训练期间，如果某个掩码的稀疏度足够高，我们就停止对其进行衰减。当所有掩码都满足稀疏度要求后，可以通过合并掩码，将整个稀疏网络转换为学生模型，同时保持网络功能不变。最后，学生模型将再进行少量训练轮次的微调。

3.3 可扩展编码器

最后，我们倡导在神经图像编解码器中采用可扩展编码器。实现可扩展编码器的一种直接方法，是直接训练多个独立的小型编码器，并配备一个统一解码器。对于特定的编码



(a) (b)

图 6：Tecnick 和 Kodak 的率失真曲线。我们的模型与 SwinT（Zhu 等人，2022）、Entroformer（Qian 等人，2022）和 STF（Zou 等人，2022）不相上下，并且优于传统方法 VTM。请注意，我们的模型运行速度远快于这些方法（参见表 1）。

复杂度，选择相应数量的编码器共同进行编码，然后将最佳比特流发送给解码器。然而，这些独立编码器会受到同质化问题的影响。我们提出残差表示学习（RRL）来缓解这一问题。如图5所示，我们多次压缩规模庞大的编码器，以逐步学习残差表示。在每一步中，仅优化压缩编码器中的参数。这样可以确保先前的编码器生成可解码的比特流。对于特定的编码复杂度，采用前 k 个编码器。我们的 RRL 鼓励编码器实现多样化。RRL 和掩码衰减都将规模庞大的教师模型作为参考，从而使训练更加有效。SlimCAE（Yang 等人，2021）使用可伸缩层（Yu & Huang，2019）获得具有动态复杂度的图像压缩模型。我们的框架实现更加简单，并且取得了更优的率失真性能。CBANet（Guo 等，2021）提出了一种用于动态复杂度的多分支结构，可将其视为我们的基线。更多细节请参见附录 C。

4 实验

训练与测试。 训练数据集采用 Vimeo-90K（Xue 等人，2019）七元组数据集中的训练部分。为进行公平比较，所有模型均训练 200 个训练轮次。对于我们的方法，教师模型经过 60 个训练轮次，通过掩码衰减转换为学生模型，然后学生模型再进行 140 个训练轮次的微调。测试数据集包括 Kodak（Franzen，1999）、HEVC 测试序列（Sharman & Suehring，2017）和 Tecnick（Asuni & Giachetti，2014）。我们采用峰值信噪比（PSNR）与每像素比特数（BPP）对应的 BD-Rate（Bjontegaard，2001）作为比较不同模型的主要指标。有关实验设置的更多细节请参见附录 D 节。

4.1 与最先进水平的比较

图 6 比较了我们的方法与其他最先进模型在 Tecnick 和 Kodak 上的表现。我们的大型模型（“EVC（我们的方法）”）优于最先进的传统图像编解码器（“VTM 17.2”），并且与其他最先进的神经图像编解码模型性能相当。需要注意的是，EVC 仅通过一个 模型即可处理不同的率失真权衡，而其他最先进方法则需要针对每种权衡分别训练模型。表 1 比较了不同模型的延迟。对于 Entroformer（Qian 等人，2022）和 STF（Zou 等人，2022），我们通过运行其官方代码来测量延迟。需要注意的是，延迟包含算术编码，但忽略了磁盘输入/输出时间。SwinT（Zhu 等人，2022）也在附录表 6 中进行了比较。我们的模型以显著优势超过了这些最先进模型。对于 768×512 输入，我们的大型模型在 A100 上可达到 30 帧/秒，而我们的小型模型甚至可在 1080P 输入下达到 30 帧/秒。对于更多较早但影响力较大的神经图像编解码器（例如 Cheng2020（Cheng 等人，2020）和 Xie2021（Xie 等人，2021）），我们在附录 F 中进行了比较。

表 1：延迟（毫秒）比较。数值越低越好。Entroformer（Qian 等人，2022年）和 STF（Zou 等人，2022年）是最先进方法。'OM'表示内存溢出。请注意，2080Ti 和 A100 的内存分别为 12GB 和 80GB。**粗体**表示运行速度超过 30 帧/秒。

分辨率	GPU	Type	Entroformer	STF		EVC		
				Transformer	CNN	大型	中型	小型
768×512	2080Ti	编码	OM	176.3	158.5	63.0	44.7	**28.4**
		解码	OM	202.3	210.2	41.1	**32.4**	**24.4**
	A100	编码	816.8	115.9	96.4	**21.1**	**19.8**	**17.7**
		解码	4361.9	143.2	118.0	**19.1**	**17.1**	**15.6**
1920×1080	2080Ti	编码	OM	576.0	456.0	305.3	181.5	90.9
		解码	OM	531.7	652.0	179.2	118.1	73.2
	A100	编码	7757.4	355.6	278.1	84.2	56.3	**31.4**
		解码	OM	354.8	281.7	60.2	46.5	**29.7**

表2：峰值信噪比的 BD-Rate（%）比较。L、M 和 S 分别表示大型、中型和小型。锚点为 VTM。BD-Rate 越低，效果越好。MACs 是针对 1920×1088 输入计算的。

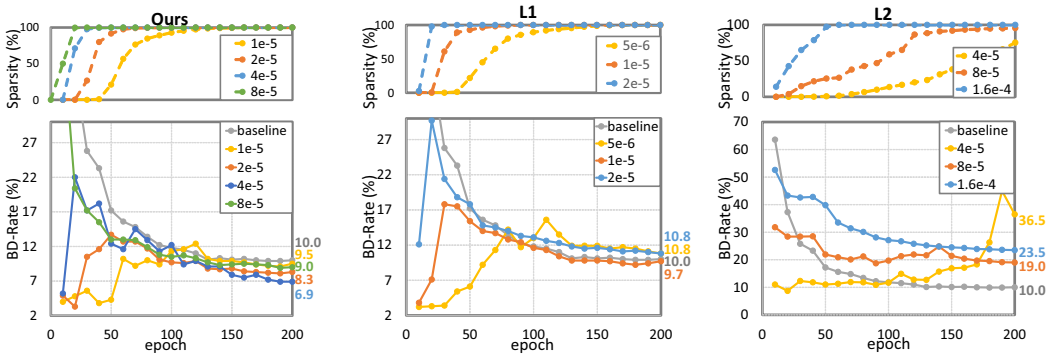
Enc	Dec	MACs（G）	方法	HEVC					Kodak
				B	C	D	E	Avg.	
L	L	1165.39	基线	-5.2	-0.9	-0.9	-3.5	-2.63	-1.1
M	L	878.37	基线	-2.8	2.2	2.5	0.0	0.48	1.1
			掩码衰减	**-4.5**	**-0.2**	**0.2**	**-3.0**	**-1.88 (76%↑)**	**-0.4 (68%↑)**
S	L	688.64	基线	0.3	7.1	7.0	2.8	4.30	4.0
			掩码衰减	**-1.9**	**2.9**	**3.5**	**0.3**	**1.20 (45%↑)**	**1.6 (47%↑)**
L	M	885.12	基线	-2.1	3.1	3.6	1.1	1.43	2.1
			掩码衰减	**-4.0**	**0.7**	**0.8**	**-1.6**	**-1.03 (60%↑)**	**0.2 (59%↑)**
L	S	700.38	基线	2.1	9.5	10.7	7.7	7.50	7.0
			掩码衰减	**-0.4**	**5.6**	**6.9**	**3.3**	**3.85 (36%↑)**	**4.0 (37%↑)**
M	M	598.10	基线	-0.8	4.7	5.0	1.6	2.63	2.9
			掩码衰减	**-3.1**	**1.6**	**1.5**	**-0.8**	**-0.20 (54%↑)**	**0.9 (50%↑)**
S	S	223.63	基线	6.3	14.7	15.6	10.6	11.80	10.0
			掩码衰减	**2.6**	**9.5**	**10.6**	**6.5**	**7.30 (31%↑)**	**6.9 (28%↑)**

4.2 掩码衰减

先前的工作（Zhu 等人，2022）发现，小型模型的率失真性能较差。我们提出的掩码衰减缓解了这一问题。我们分别加速编码器和解码器，以研究更多情况。需要注意的是，大型模型是教师模型，衰减率为 $4e-5$。图 1a 和表 2 总结了实验结果。在表 2 中，我们还计算了相对提升百分比。例如，如果基线学生模型、我们的学生模型和基线教师模型的 BD-Rate 分别为 1.1%、-0.4% 和 -1.1%，则相对提升为 $\frac{1.1-(-0.4)}{1.1-(-1.1)} = 68$%。该指标表示我们的方法缩小了学生模型与教师模型之间多少性能差距。率失真曲线见附录图 17。根据实验结果，我们得出以下结论：

• 通过掩码衰减，学生模型的性能得到了显著提升。对于中型和小型模型，相对提升幅度分别约为 50% 和 30%。这证明了我们方法的有效性，且学生模型受益于复用教师模型参数。与图 1a 中的 SwinT-Hyperprior 相比，我们的 EVC 在率失真性能与复杂度之间实现了更优的权衡。此外，我们的模型尺寸较小。

• 编码器比解码器冗余更多。请注意，我们的编码器和解码器是对称的。令人惊讶的是，我们发现，与压缩解码器相比，压缩编码器造成的率失真性能损失更小。使用大型解码器时，将编码器从大型压缩为小型，在 Kodak 上会损失 2.7% 的 BD-Rate。另一方面，仅压缩解码器则会损失 5.1% 的 BD-Rate。



(a) (b) (c)

图7：不同稀疏损失的消融研究。我们针对每种准则调整衰减率 η。建议以彩色并放大查看此图。

附录图20展示了我们模型的重建结果。不同容量模型的重建结果一致。掩码衰减不会产生明显的额外伪影。而且，我们的编码和解码时间比 VTM 短得多。

4.3 可扩展编码器

图1b展示了可扩展编码器的结果。首先，独立编码器的训练效果不佳，这是由同质化问题导致的。随后，使用我们的框架逐一训练这些编码器，以弥合残差表示之间的差异，显著提升了性能。我们的 RRL 鼓励这些编码器形成多样性。最后，掩码衰减将复杂的编码器作为参考，使训练更加高效。更多详情请参见附录 C 节。

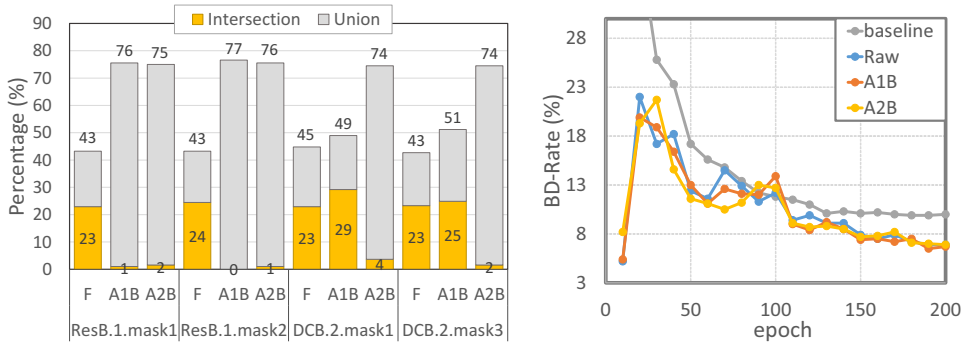
图6展示了我们提出的 Scalable-EVC 的率失真曲线。我们的方法显著优于 SlimCAE（Yang 等人，2021），BD-Rate 降低了 15.5%。值得注意的是，SlimCAE 是一种强大的模型，配备了超先验（Ball´e 等人，2018）、条件卷积（Choi 等人，2019）和自回归上下文模型（Minnen 等人，2018）。总体而言，Scalable-EVC 的表现与其他最先进模型相当。

4.4 消融实验

稀疏损失与衰减率。 图 7 汇总了采用不同稀疏性正则化损失和衰减率进行掩码衰减的实验结果。对于稀疏度，0% 和 100% 分别表示教师模型和学生模型的结构。我们针对每种稀疏损失调整衰减率 η，以获得合理的稀疏度衰减曲线。BD-Rate 是基于 Kodak 上的 VTM 计算的。对于我们的方法的稀疏损失，所有衰减率相比基线都能取得更好的率失真性能。合适的衰减率（$4e-5$）可实现最佳 BD-Rate。较大的衰减率（$8e-5$）会导致快速衰减，从而在训练轮次早期损害模型性能。该模型的衰减过程不充分，性能很容易饱和。较小的衰减率（$1e-5$ 和 $2e-5$）会在掩码衰减上浪费过多训练轮次，导致模型缺乏微调。对于 L1，衰减率过大或过小时，模型的性能都较差。即使采用合适的衰减率（$1e-5$），其性能仍明显逊于我们的方法。对于 L2，我们发现它几乎不起作用，因为当掩码接近零时梯度消失。

关于重要通道的讨论。在模型剪枝领域，许多工作（Li 等人，2016年；He 等人，2019年；Liebenwein 等人，2019年；Molchanov 等人，2019年）研究如何识别不重要通道并进行剪枝。然而，近期工作（Liu 等人，2018年；Li 等人，2022b）表明，随机剪枝通道也能取得相当甚至更优的性能，这说明所有通道之间可能并不存在重要性差异。以往工作主要研究语义任务。我们希望将这一结论推广到低级任务，即图像压缩。

首先，我们注意到，图 7a 中的掩码衰减会以不同的衰减率选择相似的通道。我们用 $\mathbb{C}_i$ 表示所选通道的集合，其中 $i \in F = \{1e-5, 2e-5, 4e-5, 8e-5\}$表示衰减率 η。我们检查了前两个块中的掩码，教师模型和学生模型的通道数分别为 192 和 64。因此，剪枝后约有 33% 的通道需要保留。然而，我们发现 $\cap_{i\in F}\mathbb{C}_i$ 和 $\cup_{i\in F}\mathbb{C}_i$ 分别包含 23% 和 43% 的通道。这些



(a) (b)

图 8：(a) 展示了不同层中，采用不同掩码衰减方式时通道所占的百分比。F、A1B 和 A2B 分别表示所有通道均可自由选择、避开第一个块，以及避开前两个块。AkB ($k = 1,\ 2$) 与 F 的交集数量较少，这意味着我们的 AMD 确实避开了 F 中的通道。(b) 展示了 Kodak 上相对于 VTM 的 BD-Rate 随训练轮次的变化。Raw 表示图 7a 中 $\eta = 4e - 5$ 对应的曲线。

所选通道重要吗？我们的答案是否定的。为验证这一观点，我们提出了避免特定通道的掩码衰减。我们将所有通道分为两组：自由通道 $\mathbb{F}$ 和规避通道$\mathbb{A}$。为避免选择 $\mathbb{A}$中的通道，我们为它们设置较大的衰减率。因此，我们的 可规避掩码衰减(AMD) 的损失为

$$\mathcal{L}_{AMD}(x) = \eta_f \mathbf{1}_{\mathrm{x}\in\mathbb{F}} \mathcal{L}_{sparse}(x) + \eta_a \mathbf{1}_{\mathrm{x}\in\mathbb{A}} \mathcal{L}_{sparse}(x) \tag{4}$$

当条件为真时，$\mathbf{1}_{\text{condition}}$ 为 1，否则为 0。默认情况下，我们设置 $\eta_a = 10\eta_f$。因此，可避开的通道具有较大的衰减率，能够快速趋近于零。我们进行了两项实验，分别在前一个或前两个块中避免选择 $\cup_{i\in F}\mathbb{C}_i$ 。图 8a 表明，我们的 AMD 确实避开了这些通道。具体而言，我们计算了 $\mathbb{C}_{AkB} \cap \mathbb{C}_F$ 和 $\mathbb{C}_{AkB} \cup \mathbb{C}_F$，其中 $k = 1,\ 2$ 表示避开前 k 个块中的通道，$\mathbb{C}_F = \cup_{i\in F}\mathbb{C}_i$。在第一个块中，A1D 和 F 共同选中的通道最多占 1%；而 A2B 在两个块中都避开了 F 中的通道。图 8b 展示了每个训练轮次对应的 BD-Rate。从中可以看出，与原始掩码衰减相比，AMD 的性能相当或略胜一筹。

总的来说，如果某些通道很重要，我们可以避开这些通道，同时实现相当的性能。这表明不存在重要通道。对于未来的工作，这一结论指导我们更加关注优化过程，而不是每个通道的重要性。

5 结论与未来工作

我们提出了用于图像压缩的 EVC。对于 768×512 输入，EVC 能够以 30 帧/秒运行，同时其率失真性能与最先进的神经编解码器相当，并优于最先进的传统编解码器（H.266/VVC）。借助可调量化步长，EVC 仅使用一个模型即可处理可变的率失真权衡。这三项进展为图像编解码器领域树立了新的里程碑。通过降低编码器和解码器的复杂度，我们提出了一种小型模型，能够针对 1080P 输入实现 30 帧/秒。我们借助繁重模型提出了掩码衰减，用于训练该小型模型。此外，我们设计了一种新颖的稀疏正则化损失，以缓解 L1 和 L2 损失的缺点。得益于我们的算法，中型模型和小型模型的性能分别显著提升了 50% 和 30%。最后，我们提出了可扩展编码器。仅使用一个解码器，便可针对不同复杂度配置多个编码器。借助残差表示学习和掩码衰减，可扩展编码器缩小了与教师模型之间的性能差距。我们的可扩展 EVC 也以显著优势超越了此前的 SlimCAE。

我们相信这篇论文将启发许多后续研究。首先，我们指出，编码器比解码器冗余更多。因此，改进解码器大有可为。其次，本文主要关注预定义的学生结构。如何确定图像压缩模型各层的复杂度，仍是一个悬而未决的问题。第三，我们也鼓励寻找更强大的剪枝算法。不过，根据我们的 AMD，我们建议更多关注优化过程，而非每个滤波器的重要性准则。

参考文献

Nicola Asuni 和 Andrea Giachetti。TESTIMAGES：用于测试视觉设备和基础图像处理算法的大型档案库。载于图形智能工具与应用会议论文集，第 63–70 页，2014 年。

Johannes Ball´e、Valero Laparra 和 Eero P Simoncelli。端到端优化的图像压缩。载于国际学习表征会议，第 1–27 页，2017年。Johannes Ball´e、David Minnen、Saurabh Singh、Sung Jin Hwang 和 Nick Johnston。具有尺度超先验的变分图像压缩。载于国际学习表征会议，第 1–23 页，2018年。Johannes Ball´e、Nick Johnston 和 David Minnen。采用潜变量模型进行数据压缩的整数网络。载于国际学习表征会议，第 1–10 页，2019年。Gisle Bjontegaard。率失真曲线之间平均峰值信噪比差异的计算。*VCEG-M33*，2001年。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*电路与系统视频技术汇刊，31(10):3736–3764，2021。Tong Chen 和 Zhan Ma。采用质量缩放因子的可变比特率图像压缩。载于*IEEE* 声学、语音与信号处理国际会议，第 2163–2167 页，2020年。Zhengxue Cheng、Heming Sun、Masaru Takeuchi 和 Jiro Katto。采用离散化高斯混合似然和注意力模块的学习式图像压缩。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 7939–7948 页，2020年。Yoojin Choi、Mostafa El-Khamy 和 Jungwon Lee。采用条件自编码器的可变码率深度图像压缩。载于*IEEE/CVF* 国际计算机视觉会议论文集，第 3146–3154 页，2019年。Ze Cui、Jing Wang、Shangyin Gao、Tiansheng Guo、Yihui Feng 和 Bo Bai。支持连续码率自适应的非对称增益深度图像压缩。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 10532–10541 页，2021。Xiaohan Ding、Guiguang Ding、Jungong Han 和 Sheng Tang。面向高效卷积神经网络的自动平衡滤波器剪枝。载于*AAAI* 人工智能会议论文集，第 32 卷，第 6797–6804 页，2018年。Xiaohan Ding、Tianxiang Hao、Jianchao Tan、Ji Liu、Jungong Han、Yuchen Guo 和 Guiguang Ding。ResRep：通过解耦记忆与遗忘实现无损 CNN 剪枝。载于*IEEE/CVF* 国际计算机视觉会议论文集，第 4510–4520 页，2021。Rich Franzen。柯达无损真彩色图像套件。来源：*http://r0k.us/graphics/kodak*，4(2)，1999年。Jinyang Guo、Dong Xu 和 Guo Lu。CBANet：使用单一网络实现复杂度与码率自适应深度图像压缩。*arXiv*预印本 *arXiv:2105.12386*，2021年。Dailan He、Yaoyan Zheng、Baocheng Sun、Yan Wang 和 Hongwei Qin。用于高效学习式图像压缩的棋盘格上下文模型。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 14771–14780 页，2021。Dailan He、Ziming Yang、Weikun Peng、Rui Ma、Hongwei Qin 和 Yan Wang。ELIC：采用不均匀分组空域-通道上下文自适应编码的高效学习式图像压缩。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 5718–5727 页，2022年。Kaiming He、Xiangyu Zhang、Shaoqing Ren 和 Jian Sun。用于图像识别的深度残差学习。载于*IEEE*国际计算机视觉会议论文集，第 770–778 页，2016年。

Yang He、Ping Liu、Ziwei Wang、Zhilan Hu 和 Yi Yang。通过几何中位数进行滤波器剪枝，加速深度卷积神经网络。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 4340–4349 页，2019年。Yihui He、Xiangyu Zhang 和 Jian Sun。用于加速超深神经网络的通道剪枝。载于*IEEE*国际计算机视觉会议论文集，第 1389–1397 页，2017年。Geoffrey Hinton、Oriol Vinyals、Jeff Dean 等。提炼神经网络中的知识。*arXiv*预印本 *arXiv:1503.02531*，2015年。Jun-Hyuk Kim、Byeongho Heo 和 Jong-Seok Lee。用于学习式图像压缩的联合全局和局部层次先验。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 5992–6001 页，2022年。Hao Li、Asim Kadav、Igor Durdanovic、Hanan Samet 和 Hans Peter Graf。用于高效卷积神经网络的滤波器剪枝。*arXiv*预印本 *arXiv:1608.08710*，2016年。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，第 1503–1511 页，2022a。Yawei Li、Kamil Adamczewski、Wen Li、Shuhang Gu、Radu Timofte 和 Luc Van Gool。重新审视神经网络压缩中的随机通道剪枝。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 191–201 页，2022b。Lucas Liebenwein、Cenk Baykal、Harry Lang、Dan Feldman 和 Daniela Rus。用于高效神经网络的可证明滤波器剪枝。*arXiv*预印本 *arXiv:1911.07412*，2019年。Zhuang Liu、Jianguo Li、Zhiqiang Shen、Gao Huang、Shoumeng Yan 和 Changshui Zhang。通过网络瘦身学习高效卷积网络。载于*IEEE*国际计算机视觉会议论文集，第 2736–2744 页，2017年。Zhuang Liu、Mingjie Sun、Tinghui Zhou、Gao Huang 和 Trevor Darrell。重新思考网络剪枝的价值。载于国际学习表征会议，第 1–21 页，2018年。Zhuang Liu、Hanzi Mao、Chao-Yuan Wu、Christoph Feichtenhofer、Trevor Darrell 和 Saining Xie。面向2020年代的卷积神经网络。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 11976–11986 页，2022年。Ilya Loshchilov 和 Frank Hutter。解耦权重衰减正则化。载于国际学习表征会议，第 1–19 页，2018年。Jian-Hao Luo 和 Jianxin Wu。采用残差连接和有限数据的神经网络剪枝。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 1458–1467 页，2020年。David Minnen 和 Saurabh Singh。用于学习式图像压缩的逐通道自回归熵模型。载于*2020*年*IEEE*国际图像处理会议 *(ICIP)*，第 3339–3343 页，2020年。David Minnen、Johannes Ball´e 和 George D Toderici。用于学习式图像压缩的联合自回归与层次先验。载于神经信息处理系统进展，第 31 卷，第 1–10 页，2018年。Pavlo Molchanov、Arun Mallya、Stephen Tyree、Iuri Frosio 和 Jan Kautz。神经网络剪枝的重要性估计。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 11264–11272 页，2019年。Yichen Qian、Ming Lin、Xiuyu Sun、Zhiyu Tan 和 Rong Jin。Entroformer：用于学习式图像压缩的基于Transformer的熵模型。载于国际学习表征会议，第 1–14 页，2022年。Karl Sharman 和 Karsten Suehring。HM 通用测试条件。*JCTVC-Z1100*，2017年。

Wenzhe Shi、Jose Caballero、Ferenc Husz´ar、Johannes Totz、Andrew P Aitken、Rob Bishop、Daniel Rueckert 和 Zehan Wang。使用高效亚像素卷积神经网络实现实时单图像和视频超分辨率。载于*IEEE*计算机视觉与模式识别会议论文集，第 1874–1883 页，2016年。王国华和 Jianxin Wu。使用小型集合实现实用网络加速。*arXiv*预印本 *arXiv:2202.07861*，2022年。王国华、Yifan Ge 和 Jianxin Wu。通过模仿特征蒸馏知识。*IEEE*模式分析与机器智能汇刊，44(11):8183–8195，2022年。Wei Wen、Chunpeng Wu、Yandan Wang、Yiran Chen 和 Hai Li。在深度神经网络中学习结构化稀疏性。载于神经信息处理系统进展，第 29 卷，第 1–9 页，2016年。Yueqi Xie、Ka Leong Cheng 和 Qifeng Chen。用于学习式图像压缩的增强型可逆编码。载于第*29*届 *ACM*国际多媒体会议论文集，第 162–170 页，2021。Tianfan Xue、Baian Chen、Jiajun Wu、Donglai Wei 和 William T Freeman。利用面向任务的光流进行视频增强。国际计算机视觉杂志，127(8):1106–1125，2019年。Fei Yang、Luis Herranz、Yongmei Cheng 和 Mikhail G. Mozerov。用于实用神经图像压缩的可伸缩压缩自编码器。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 4998–5007 页，2021。Jiahui Yu 和 Thomas S Huang。通用可伸缩网络及改进的训练技术。载于*IEEE/CVF* 国际计算机视觉会议论文集，第 1803–1811 页，2019。Jiahui Yu、L. Yang、N. Xu、Jianchao Yang 和 Thomas S. Huang。可伸缩神经网络。载于国际学习表征会议，第 1–12 页，2019。Yulun Zhang、Huan Wang、Can Qin 和 Yun Fu。面向高效图像超分辨率的对齐结构化稀疏学习。载于神经信息处理系统进展，第 34 卷，第 2695–2706 页，2021。Borui Zhao、Quan Cui、Renjie Song、Yiyu Qiu 和 Jiajun Liang。解耦知识蒸馏。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 11953–11962 页，2022年。Yinhao Zhu、Yang Yang 和 Taco Cohen。基于Transformer的变换编码。载于国际学习表征会议，第 1–35 页，2022年。Renjie Zou、Chunfeng Song 和 Zhaoxiang Zhang。细节决定成败：用于图像压缩的基于窗口的注意力。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 17492–17501 页，2022年。

EVC：借助掩码衰减实现实时神经图像压缩 -附录-

EVC 的框架

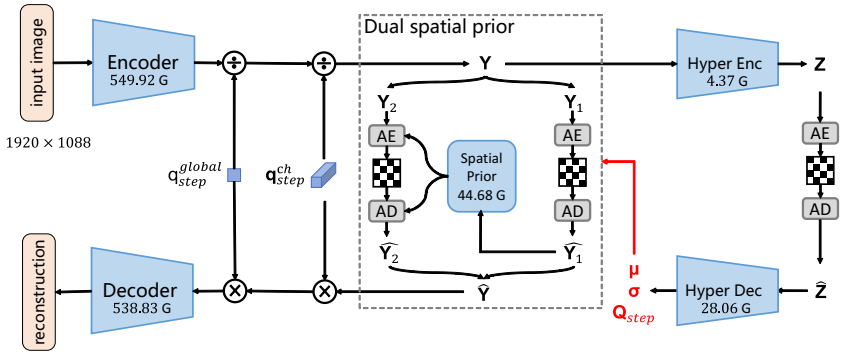
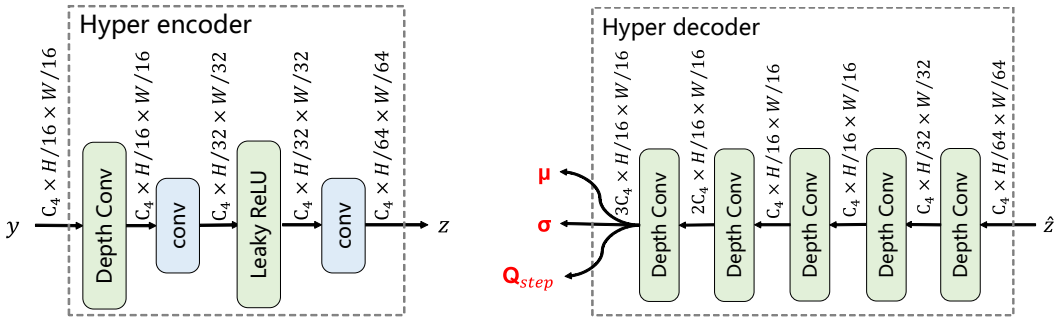


图 9：EVC 的整体框架。

我们的图像压缩 EVC 如图 9 所示。接下来，我们将详细介绍各个模块。

编码器和解码器。这两个模块的结构如图 3 所示。各模块块的输出维度分别由 C_1、 C_2、 C_3和 C_4预先设定。下采样通过将特定卷积层的步幅设为 2 来实现，而上采样则通过子像素卷积实现（Shi 等，2016）。在每个残差块中，卷积层 #1 和 #2 的卷积核尺寸均为 3×3，而卷积层 #3 的卷积核尺寸为 1×1。卷积层 #1 会将通道数扩展至输出维度。在每个深度卷积块中，只有深度卷积层使用 3×3 卷积，其余层均采用 1×1 卷积。卷积层 #3 将通道数扩展 4 倍，而卷积层 #4 则将其恢复。

超编码器和超解码器。图 10 描述了我们超先验的结构。请注意，我们在超编码器中使用了 3×3 conv.。



(a) (b)

图 10：（a）和（b）分别展示了超编码器和超解码器的结构。

双空间先验。遵循 He 等人（2021）、Li 等人（2022a）和 Qian 等人（2022）的做法，我们采用双空间先验来减少空间冗余。其详细结构如图 11 所示。表示 $\mathbf{Y}$ 在空间上被拆分为两部分 $\mathbf{Y}_1$ 和 $\mathbf{Y}_2$ 。首先，根据来自超先验的边信息对 $\mathbf{Y}_1$ 进行编码。然后，给定重建结果 $\mathbf{Y}_1, \mathbf{Y}_2$ 可以得到更有效的估计。我们的双空间先验与自回归先验（Minnen 等人，2018）的思路相似，

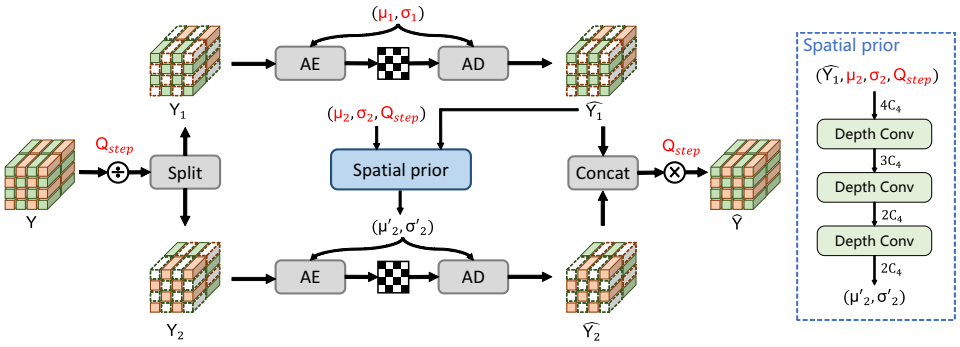


图 11：我们的双空间先验结构。

后者根据邻居对目标进行条件预测。但我们的方法支持并行计算，且推理速度快。

不同容量的模型方案。表3总结了本文使用的三个模型。我们还比较了 1920×1088 输入的模型大小和乘加运算次数。

表 3：模型参数数量（MB）和乘加运算次数（G）的架构方案统计。请注意，其他包括超先验模型、空间先验模型和熵模型。输入图像的分辨率为 1920×1088。

模型	C_1, C_2, C_3, C_4		编码器	解码器	其他	总计
大型 (大型)	192, 192, 192, 192	参数数量	3.19	3.38	10.82	17.38
		MACs	549.92	538.83	76.64	1165.39
中型（中型）	128, 128, 192, 192	参数数量	2.08	2.33	10.82	15.23
		MACs	262.90	258.56	76.64	598.10
小型（小型）	64, 64, 128, 192	参数数量	0.82	1.14	10.82	12.78
		MACs	73.17	73.82	76.64	223.63

B 掩码衰减

我们模型加速方法的流程。图 12 展示了如何通过掩码衰减加速一个庞大的网络。首先，训练这个庞大的网络，并将其作为预训练教师网络。然后，在教师网络中插入掩码层，并对其进行优化以使其变得稀疏。训练完成后，将掩码合并到相邻层中，从而将庞大的教师网络转化为高效的学生网络。

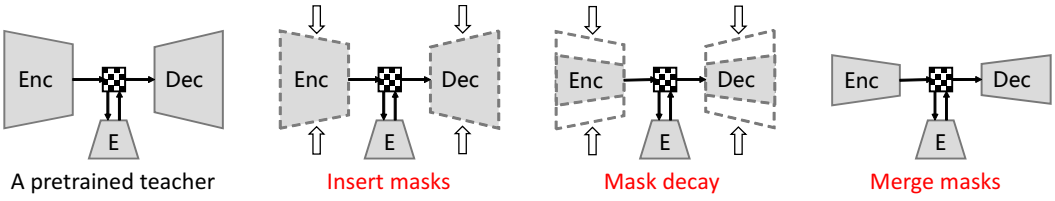


图 12：我们模型加速过程的示意图。“Enc”和“Dec”分别表示编码器和解码器。“E”表示包含超先验模型的熵模型。

通过掩码衰减压缩一层的表示。 在图 4a 中，我们的掩码包含 N_2 个参数，这些参数初始化为 1。训练后，N_s 个参数保持为正，而 $N_2 - N_s$个参数变为零。随后，我们分两步“合并”该掩码。首先，对于 $N_2 - N_s$

个零通道，可以安全地剪除卷积层 #1 和卷积层 #2 中对应的滤波器。其次，对于 N_s 个正参数，将其乘入卷积层 #2。完成这两步后，即可移除掩码。断言 1 保证这两步不会改变输出。因此，我们的稀疏掩码能够被安全地消除。

断言 1. 我们的掩码可以合并到相邻的卷积层中，同时保持输出不变。

证明。如图 4a 所示，假设 $\boldsymbol{W}_1$ 和 $\boldsymbol{b}_1$ 是剪枝卷积层 #1 中的参数，$\boldsymbol{W}_2$ 和$\boldsymbol{b}_2$ 是剪枝卷积层 #2 中的参数。LeakyReLU 应用逐元素函数：

$$f(x) = \begin{cases} x, & \text{if } x \geq 0\,, \\ \text{negative_slope} \times x, & \text{otherwise}\,. \end{cases} \tag{5}$$

那么，给定输入 $\mathbf{X} \in R^{N_1 \times H \times W}$，剪枝模型的前向计算（图 4a 左侧部分）为

$$\mathbf{Y}_1 = \boldsymbol{W}_1 \circledast \mathbf{X} + \boldsymbol{b}_1\,, \qquad (\mathbf{Y}_1 \in R^{N_2 \times H \times W}, \boldsymbol{W}_1 \in R^{N_2 \times N_1 \times 3 \times 3}, \boldsymbol{b}_1 \in R^{N_2})\,, \tag{6}$$

$$\mathbf{Y}_2 = \mathbf{Y}_1 \odot_c \boldsymbol{m}\,, \qquad (\mathbf{Y}_2 \in R^{N_2 \times H \times W}, \boldsymbol{m} \in R^{N_2})\,, \tag{7}$$

$$\mathbf{Y}_3 = f(\mathbf{Y}_2)\,, \qquad (\mathbf{Y}_3 \in R^{N_2 \times H \times W})\,, \tag{8}$$

$$\mathbf{Y}_4 = \boldsymbol{W}_2 \circledast \mathbf{Y}_3 + \boldsymbol{b}_2\,, \qquad (\mathbf{Y}_4 \in R^{N_3 \times H \times W}, \boldsymbol{W}_2 \in R^{N_3 \times N_2 \times 3 \times 3}, \boldsymbol{b}_2 \in R^{N_3})\,, \tag{9}$$

其中，$\circledast$ 表示卷积算子，$\odot_c$ 表示按通道乘积。注意，$\boldsymbol{m}$ 中的 N_s个元素为正，其余 $N_2 - N_s$ 个条目均为零。首先，可以安全地剪除这 $N_2 - N_s$个零通道，此时前向计算变为

$$\mathbf{Y}'_1 = \boldsymbol{W}'_1 \circledast \mathbf{X} + \boldsymbol{b}'_1\,, \qquad (\mathbf{Y}'_1 \in R^{N_s \times H \times W}, \boldsymbol{W}'_1 \in R^{N_s \times N_1 \times 3 \times 3}, \boldsymbol{b}'_1 \in R^{N_s})\,, \tag{10}$$

$$\mathbf{Y}'_2 = \mathbf{Y}'_1 \odot_c \boldsymbol{m}'\,, \qquad (\mathbf{Y}'_2 \in R^{N_s \times H \times W}, \boldsymbol{m}' \in R^{N_s})\,, \tag{11}$$

$$\mathbf{Y}'_3 = f(\mathbf{Y}'_2)\,, \qquad (\mathbf{Y}'_3 \in R^{N_s \times H \times W})\,, \tag{12}$$

$$\mathbf{Y}_4 = \boldsymbol{W}'_2 \circledast \mathbf{Y}'_3 + \boldsymbol{b}_2 \qquad (\mathbf{Y}_4 \in R^{N_3 \times H \times W}, \boldsymbol{W}'_2 \in R^{N_3 \times N_s \times 3 \times 3}, \boldsymbol{b}_2 \in R^{N_3})\,. \tag{13}$$

然后，在第二步中，$\boldsymbol{m}'$ 中的 N_s 正值将被乘入 $\boldsymbol{W}'_2$。我们有

$$\mathbf{Y}_4 = \boldsymbol{W}'_2 \circledast f(\mathbf{Y}'_1 \odot_c \boldsymbol{m}') + \boldsymbol{b}_2\,, \tag{14}$$

$$= \boldsymbol{W}'_2 \circledast (f(\mathbf{Y}'_1) \odot_c \boldsymbol{m}') + \boldsymbol{b}_2\,, \tag{15}$$

$$= (\boldsymbol{W}'_2 \odot_c \boldsymbol{m}') \circledast f(\mathbf{Y}'_1) + \boldsymbol{b}_2\,. \tag{16}$$

最后一个等式成立是因为卷积算子的线性性质。请注意，公式 16 表示剪枝后的模型中的计算过程（图 4a 的右侧部分）。因此，剪枝后的模型在输入$\mathbf{X}$时也会输出$\mathbf{Y}_4$。这意味着输出保持不变。合并掩码 $\boldsymbol{m}$后，卷积层 #1 和 # 2 中的参数分别变为$(\boldsymbol{W}'_1, \boldsymbol{b}'_1)$和$(\boldsymbol{W}'_2 \odot_c \boldsymbol{m}', \boldsymbol{b}_2)$。□

我们的编码器和解码器的掩码衰减。如何剪除残差连接之外的通道也是一个关键问题。图 13 展示了两种方法。请注意，掩码 #1、#2 和 #3 共享权重。如果像图 13b 那样在残差相加之后插入掩码 #2 和 #3，恒等输出会受到这些掩码的影响，因为大多数掩码都处于 $[0, 1]$ 中。像图 13a 那样插入掩码可以缓解这个问题，并确保恒等梯度反向传播。

C 可扩展编码器

在本部分中，我们将进一步介绍实现可扩展编码器的不同方法。

端到端。CBANet（Guo 等，2021）提出了一种用于可伸缩网络的多分支结构。在测试时，通过使用不同数量的分支，可以动态调整计算复杂度。本文将此方法视为基线。也就是说，直接对我们的框架进行端到端训练（参见图 14）。训练期间，所有编码器都会生成用于解码的比特流，并计算相应的损失。然而，这种基线方法在训练时需要大量 GPU 内存，并且

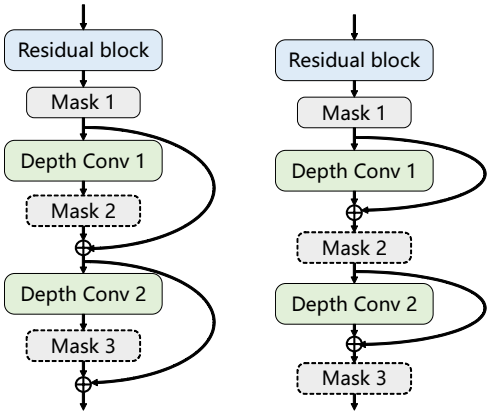


图 13：(a) 和 (b) 展示了为块级维度插入掩码的两种方式。(a) 更好，因为它保证了恒等梯度反向传播。

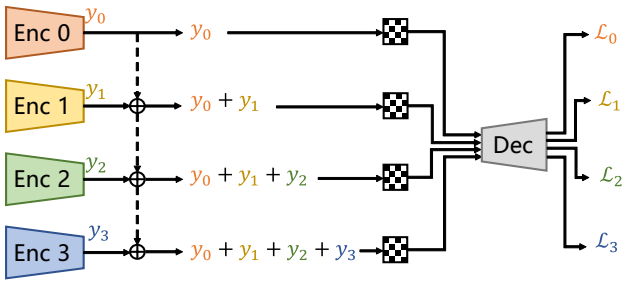


图 14：以端到端方式训练可扩展编码器的一种简单基线。虚线表示梯度已停止传播。

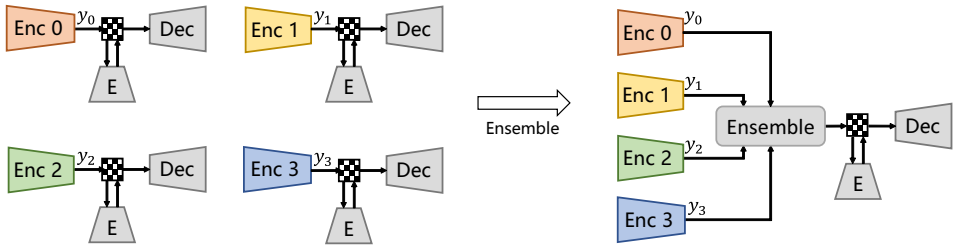


图 15：通过分开训练小型编码器来实现可扩展编码器的示意图。

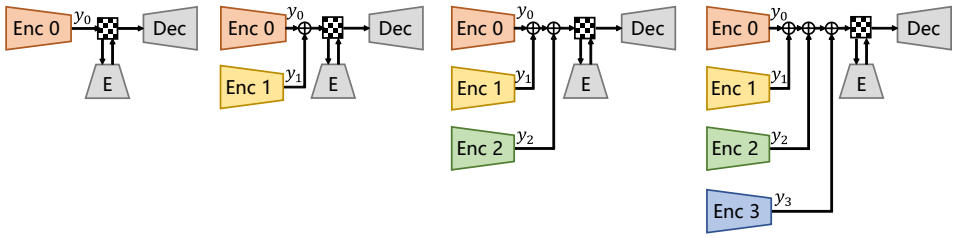


图 16：通过逐一训练小型编码器来实现可扩展编码器的示意图。每次都会鼓励新的小型编码器学习残差表示。

表 4：Kodak 上峰值信噪比的 BD-Rate（%）比较。编码器具有可扩展性，解码器为大型。表中的乘加运算次数对应于以 1920×1088 分辨率进行编码，而其他模块的乘加运算次数总计为 615.47 G。锚点为 VTM。

	S_0	$S_{0,1}$	$S_{0,1,2}$	$S_{0,1,2,3}$	L
乘加运算次数（G）	73.17	146.34	219.51	292.68	549.92
端到端	3.1	2.2	1.8	1.6	-1.1
分开训练	2.8	1.9	1.7	1.5	
逐一	2.8	1.4	0.9	0.6	
Ours	**1.4**	**0.4**	**-0.1**	**-0.2**	

实验表明，这种方法性能较差。端到端训练需要优化整个网络的所有参数，而且无法利用预训练的复杂模型中已有的知识。因此，训练效率低下。

分开训练。另一种基线方法是直接使用统一解码器训练多个独立的小型编码器。如图15所示，使用冻结的解码器和熵模型训练四个小型编码器。测试时，可根据特定的编码复杂度选择不同数量的编码器。在图15右侧，选择了全部四个编码器。集成模块会接收所有编码器生成的比特流，并选择最佳比特流发送。请注意，编码时可以获取原始图像。集成模块能够重建每个比特流，并根据原始图像计算率失真得分。然而，这些独立编码器存在同质化问题。使用统一解码器时，这些编码器生成的比特流相似。图1b显示，增加编码器数量只能略微提升率失真性能。

逐一（RRL）。我们提出残差表示学习（RRL），以缓解同质化问题。如图16所示，我们逐步训练小型编码器。每一步都会添加一个新的小型编码器，并从头开始训练。请注意，其他部分均被冻结。我们鼓励新编码器学习残差表示，从而有效缓解同质化问题，并提升这些小型编码器的多样性。图1b显示，“逐一”方法的表现显著优于“独立编码器”。

我们的方法（RRL +掩码衰减）。最后，掩码衰减进一步提升了率失真性能。如图5所示，掩码衰减直接利用了大型模型的知识。每一步都会通过对教师模型进行剪枝来获得小型编码器。这里的教师模型指表3中的大型编码器，而学生模型指小型编码器。在图5中，虚线边界的编码器表示配备了掩码的教师编码器。训练过程中，这些掩码会逐渐变得稀疏，使网络能够转变为学生结构（即学生编码器）。RRL和掩码衰减都将大型教师模型作为参照，从而使训练更加有效。图1b显示，“我们的方法”（RRL + 掩码衰减）取得了更优的率失真性能，并显著缩小了与教师模型之间的性能差距。

表4总结了我们采用大型解码器的可扩展编码器的结果。我们还针对小型解码器进行了实验，结果总结于表5中。在这些表格中，$S_{0,\ldots,i}$ 表示使用$i+1$ 小型编码器，而 L 表示原始大型编码器。

表 5：Kodak 上 PSNR 的 BD-Rate（%）对比。编码器具有可扩展性，解码器为小型。表中的乘加运算次数是以 1920×1088 分辨率进行编码时的运算次数，其他模块的乘加运算次数总计为 150.46 G。锚点为 VTM。

	S_0	$S_{0,1}$	$S_{0,1,2}$	$S_{0,1,2,3}$	L
乘加运算次数（G）	73.17	146.34	219.51	292.68	549.92
端到端	9.1	8.6	8.2	7.9	4.0
逐一	7.3	6.3	5.8	5.5	
Ours	**6.3**	**5.6**	**5.0**	**4.8**	

D 实验详情

训练。训练数据集采用 Vimeo-90K（Xue 等人，2019年）七元组数据集的训练部分。训练期间，原始图像会随机裁剪为 256×256 个图像块。为进行公平比较，所有模型均训练 200 个训练轮次。对于我们的方法，教师模型经过 60 个训练轮次并结合掩码衰减转换为学生模型，随后学生模型再经过 140 个训练轮次的微调。我们使用 AdamW 作为优化器，批量大小为 16。初始学习率为 $2e-4$，并在第 50、90、130、170 个训练轮次时衰减 0.5。我们方法中掩码衰减的默认衰减率 η 为 $4e-5$。所有模型均在配备 8 个 V100 GPU 的计算机上进行训练。我们的大型模型和小型模型的训练时间分别约为 33 小时和 16 小时。

测试。峰值信噪比（PSNR）的 BD-Rate（Bjontegaard，2001）是我们比较不同模型的主要指标。我们使用 VTM 17.2 作为锚点，负值表示比特率节省，正值表示比特率增加。测试数据集包括 Kodak（Franzen，1999）、HEVC 测试序列（Sharman & Suehring，2017）和 Tecnick（Asuni & Giachetti，2014）。Kodak 包含 24 张分辨率为 768×512 的图像。Tecnick 包含 100 张分辨率为 1200×1200 的图像。HEVC 是一个视频数据集，包含 4 个类别：B（1080P）、C（480P）、D（240P）和 E（720P）。在 HEVC 上测试图像编解码器时，仅使用每个视频序列的第一帧。我们报告 HEVC 各类别的 BD-Rate，并计算平均值以便于比较。对于每张测试图像，我们在边界处填充零，使分辨率成为 64 的倍数。我们在两台计算机上测试延迟。一台配备一块 2080Ti 图形处理器和两颗 Intel(R) Xeon(R) E5-2630 v3 CPU。另一台配备一块 A100 图形处理器和一颗 AMD Epyc 7V13 CPU。我们统计编码和解码时间。请注意，延迟包含算术编码，但不包括磁盘输入/输出时间。

E 其他图像编解码器的实现细节

VTM-17.2[1] 是最先进的传统编解码器（H.266/VVC）的参考软件，于2022年7月17日发布。我们使用 CompressAI-1.1.0[2] 收集评估结果。我们根据官方说明安装这两款软件。使用 VTM 的默认配置。所有结果均通过运行以下命令收集：

```
python –m compressai.utils.bench vtm [图像文件夹路径] –c [VTM 文件夹路径]/cfg/
encoder intra vtm.cfg –b [VTM 文件夹路径]/bin –j 8 –q 24 26 28 30 32 34 36 38 40 42
```

Entroformer[3] 由 Qian 等人（2022年）提出。该方法发布了两种序列模型：并行型和非并行型。我们分别在 Tecnick 和 Kodak 上评估并行模型的率失真性能。但在测量其延迟时，我们发现生成比特流需要大量 GPU 内存。

[1] https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM/-/releases/ VTM-17.2
[2] https://github.com/InterDigitalInc/CompressAI/releases/tag/v1.1.0
[3] https://github.com/damo-cv/entroformer

表 6：与 SwinT 的延迟（毫秒）对比。数值越低越好。为与 SwinT 进行比较，我们在 2080Ti 上使用 768×512 输入测试我们的方法，并直接引用其数值。请注意，SwinT 在编码时不会重建图像，而我们的方法会重建图像。

延迟	SwinT				EVC		
	SwinT-ChARM	SwinT-Hyperprior	Conv-ChARM	Conv-Hyperprior	大型	中型	小型
编码	135.3	100.5	95.7	62.0	63.0	44.7	28.4
解码	167.3	114.0	264.0	219.3	41.1	32.4	24.4

STF[4] 由 Zou 等人（2022）提出。作者发布了两种架构：基于 CNN 的模型和基于 Transformer 的模型。但目前仅提供了对应两个质量等级的两个预训练模型。我们在 Tecnick 上评估其中一个模型，并引用论文中 Kodak 上的数值。

SwinT-ChARM 由 Zhu 等人（2022）提出。由于作者没有发布代码，我们引用其论文和网页[5]中的数值。其论文报告了延迟，表 6 将我们的方法与他们的方法进行了比较。我们的方法延迟更低。

ELIC 由 He 等人（2022）提出。这项工作提出了非均匀通道条件自适应编码。He 等人（2022）设计了 ELIC 和 ELIC-sm 两种模型，其中 ELIC-sm 是基于 ELIC 改编的轻量架构。由于作者没有发布代码，我们根据其论文中的数值进行比较。

Cheng2020 由 Cheng 等人（2020）提出。该方法引入注意力模块和高斯混合模型，以提升率失真性能。由于影响力较大，它一直被视为神经图像编解码器的基线方法。然而，他们的原始模型采用了自回归模块，而且不建议在图形处理器上对自回归模型进行推理。在中央处理器上测试时，处理一张图像需要一秒以上。He 等人（2021）提出了棋盘格模块来缓解这一问题。我们在表 7 中将我们的方法与该模型进行了比较。

Xie2021 由 Xie 等人（2021）提出。该方法采用可逆神经网络作为更强大的主干网络。但与 Cheng2020 一样，Xie2021 也配备了自回归模块，因此延迟较高。我们在表 7 中将我们的方法与该模型进行了比较。

Minnen2018 和 Minnen2020 分别由 Minnen 等人（2018）和 Minnen 与 Singh（2020）提出。自回归上下文模型由 Minnen 等人（2018）首次引入，用于提升神经图像编解码器的率失真性能。Minnen 与 Singh（2020）通过考虑通道级冗余进一步改进了该模型。这两项工作对神经图像压缩产生了深远影响，并且通常被视为比较时的基线方法。我们也在表 7 中将我们的方法与它们进行了比较。

Ball´e2018由 Ball´e 等人（2018）提出。该方法引入了超先验结构，这是图像编解码器领域的一项里程碑。大多数后续模型，包括我们的方法，都采用了这一结构。由于没有上下文模块，该方法具有较高的推理速度。但由于未采用近期提出的技术，与最新的最先进模型相比，其率失真性能较差。

F 更多实验结果

与更多方法的比较。He 等人（2022）分别提出了两种用于高效学习式图像压缩的模型，即 ELIC 和 ELIC-sm。然而，他们没有公开代码和模型，因此我们难以进行公平比较。我们按照他们的测试方式，对比其论文中的数据。首先，他们在补充材料中提到，测试模型速度时未启用确定性推理模式。我们分别在确定性／非确定性模式下测试延迟。我们发现，非确定性会使延迟平均降低 16%，但也容易导致解码灾难性失败（Ball´e 等人，2019）。其次，与我们在正文中使用算术编码测试端到端延迟不同，这里我们按照 He 等人（2022）的方法测试各模块的延迟。因此，在不考虑算术编码的情况下，

[4]https://github.com/Googolxx/STF[5]
https://openreview.net/forum?id=IDwN6xjHnK8

表 7：BD-Rate（%）和推理延迟（毫秒）对比。BD-Rate 根据 Kodak 图像集相对于锚点 VTM 的峰值信噪比-每像素比特数曲线计算。BD-Rate 越低越好。我们的模型分别在确定性／非确定性模式下测试。其他方法由 He 等人（2022）在非确定性模式下测试。

GPU	模型	BD-Rate	推理延迟 总计。	Y 编码器。	Y 解码器。	Z 编码器。	Z 解码器。	参数量。
TITAN Xp	Ball´e2018（Ball´e 等人，2018年）	40.85	30.3	11.9	16.1	1.2	1.1	-
	Minnen2018（He 等人，2021）	20.00	37.6	11.6	17.3	1.5	1.5	5.7
	Cheng2020（He 等人，2021）	3.89	90.8	36.8	44.2	0.9	1.2	7.7
	Minnen2020（Minnen 与 Singh，2020年）	1.11	77.7	10.6	15.4	1.4	1.5	48.8
	Xie2021（Xie 等人，2021）	-0.54	$> 10^3$	58.0	162.2	1.3	1.3	$> 10^3$
	ELIC-sm（He 等人，2022年）	-1.07	41.5	12.6	17.4	1.1	1.4	9.0
	ELIC（He 等人，2022年）	-7.88	82.4	31.2	38.3	1.2	1.6	10.1
2080Ti	EVC-SS（确定性）	6.9	22.1	4.2	7.4	1.4	5.4	3.7
	EVC-SS（非确定性）	6.9	17.9	3.8	5.7	0.9	3.7	3.8
	EVC-MM（确定性）	0.9	31.4	9.1	11.9	1.4	5.3	3.7
	EVC-MM（非确定性）	0.9	26.5	8.3	10.2	0.9	3.5	3.6
	EVC-LL（确定性）	-1.1	44.5	15.5	18.6	1.4	5.3	3.7
	EVC-LL（非确定性）	-1.1	38.4	14.1	16.2	0.9	3.6	3.6

表8：速度（百万像素/秒）对比。数值越高越好。Entroformer（Qian 等人，2022年）和 STF（Zou 等人，2022年）是最先进方法。“内存溢出”表示内存不足。请注意，2080Ti 和 A100 分别具有 12GB 和 80GB 内存。

分辨率	GPU	Type	Entroformer	STF Transformer	STF CNN	EVC 大型	EVC 中型	EVC 小型
768×512	2080Ti	编码	OM	2.23	2.48	6.24	8.80	13.85
		解码	OM	1.94	1.87	9.57	12.14	16.12
	A100	编码	0.48	3.39	4.08	18.64	19.86	22.22
		解码	0.09	2.75	3.33	20.59	23.00	25.21
1920×1080	2080Ti	编码	OM	3.60	4.55	6.79	11.42	22.81
		解码	OM	3.90	3.18	11.57	17.56	28.33
	A100	编码	0.27	5.83	7.46	24.63	36.83	66.04
		解码	OM	5.84	7.36	34.45	44.59	69.82

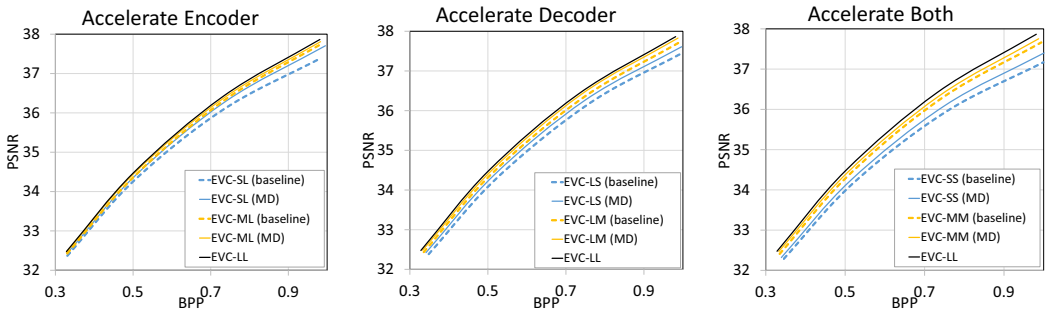
the 延迟低于我们主论文中的结果。第三，遗憾的是，我们没有 TITAN Xp [6] GP美国版，因此我们在 2080Ti[7] 上测试模型。

表 7 总结了结果。对于我们的模型，“S”“M”和“L”分别表示小型、中型和大型。我们用两个字母分别表示编码器和解码器。与以往最先进的方法（Ball´e2018、Minnen2018、Cheng2020、Minnen2020、Xie2021）相比，我们的大型模型（“EVC-LL”）取得了更好的率失真性能和推理延迟表现，也可与 ELIC-sm 相媲美。请注意，我们的方法只需一个模型即可应对所有率失真权衡，而 ELIC 针对每种权衡分别训练了不同的模型。

Spe速度（百万像素/秒）对比。在表 8 中，我们还计算了模型的速度，单位为百万像素/秒，作 -cor如表 1 所示。我们的模型在这一指标上也优于此前的模型。

不同模型的对比。图 17 展示了 Kodak 上的率失真曲线。我们的掩码衰减“MD”显著提升了基线模型的性能。在这些图中，“L”、“M”和“S”分别表示大型，

[6]https://www.nvidia.com/en-us/titan/titan-xp/[7] https://www.nvidia.com/en-us/geforce/graphics-cards/compare/?section=compare-20



(a) (b) (c)

图 17：不同复杂度模型在 Kodak 上的率失真曲线。“MD”表示掩码衰减。(a) 和 (b) 分别展示仅加速编码器或解码器时的曲线。(c) 展示同时加速编码器和解码器时的曲线。

表 9：CLIC2021 和 Tecnick 上峰值信噪比的 BD-Rate（%）对比。L、M 和 S 分别表示大型、中型和小型。锚点为 VTM。BD-Rate 越低，效果越好。乘加运算次数针对1920×1088 输入。

Enc	Dec	乘加运算次数 (G)	方法	CLIC2021	Tecnick
L	L	1165.39	基线	-0.7	-2.0
M	L	878.37	基线	1.0	-0.0
			掩码衰减	**0.6** (24%↑)	**-0.3** (15%↑)
S	L	688.64	基线	4.3	1.9
			掩码衰减	**1.9** (48%↑)	**0.2** (44%↑)
L	M	885.12	基线	2.4	1.0
			掩码衰减	**0.9** (48%↑)	**-0.3** (43%↑)
L	S	700.38	基线	7.1	5.4
			掩码衰减	**4.3** (36%↑)	**2.7** (36%↑)
M	M	598.10	基线	4.6	2.6
			掩码衰减	**1.6** (57%↑)	**0.7** (41%↑)
S	S	223.63	基线	11.6	9.9
			掩码衰减	**8.5** (25%↑)	**7.6** (19%↑)

分别为中型和小型。“EVC-SL”表示编码器为小型、解码器为大型。

表 9 展示了这些模型在 CLIC2021[8] 和 Tecnick 上的结果。掩码衰减以显著优势超越了基线。

稀疏损失和衰减率。为便于阅读，我们将图 7 以更大尺寸重新绘制。图 18 和图 19 是重新绘制的结果。

关于重要通道的讨论。表 10 展示了图 8a 中的具体数值。前两个模块中有四个掩码。我们以第一个掩码（ResB.1.mask1）为例进行分析。掩码衰减需要从教师模型的 192 个通道中选出 64 个通道。在四种不同的衰减率下，只有 83 个通道被纳入考虑，而它们最终共同选出了 44 个通道。这些通道似乎对取得优异性能十分重要。我们将这 83 个通道视为 $\mathbb{C}_F$，并让我们的算法避免选取这些通道。A1B（在第一个模块中避开特定通道）在 $\mathbb{C}_F$ 中只选取了 2 个通道，而 A2B 选取了 3 个。首先，这表明我们的 AMD 确实避开了特定通道。接下来，我们比较这些方法的率失真性能。令人惊讶的是，AMD

[8]https://storage.googleapis.com/clic2021_public/professional_test_2021.zip

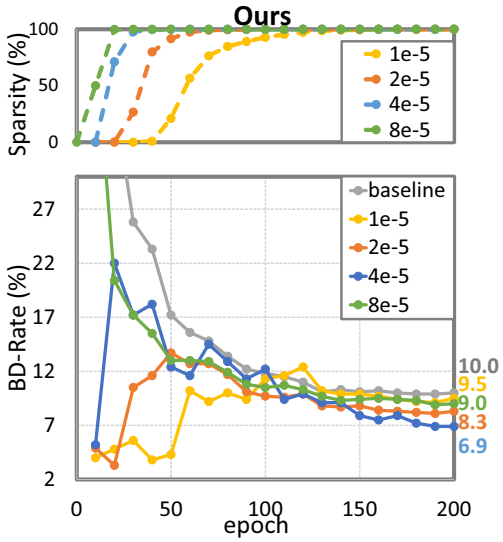


图 18：不使用稀疏损失的掩码衰减（公式 2）。我们尝试了不同的衰减率 η。此图为图 7a 的复现。

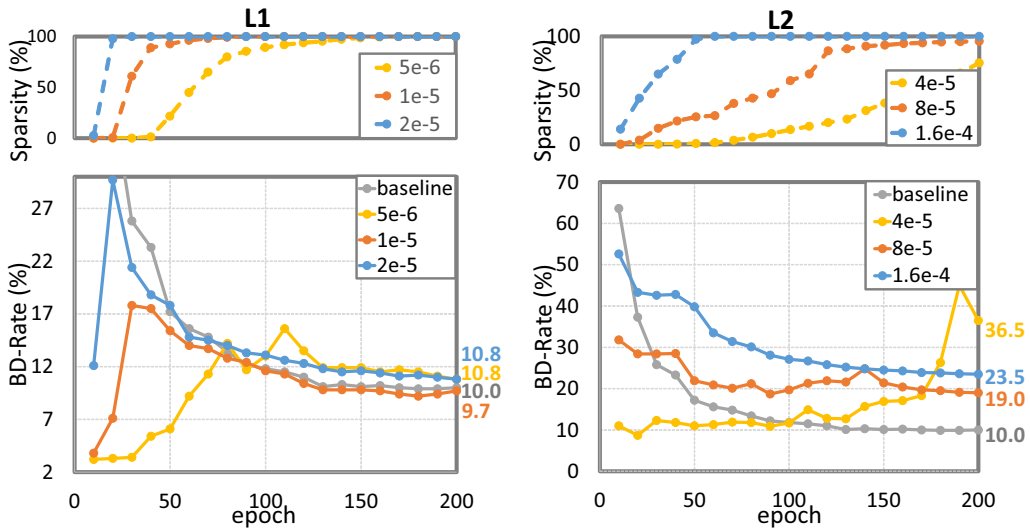


图 19：使用 L1 和 L2 损失的掩码衰减。我们针对每种准则调整了衰减率 η。这两张图分别复现了图 19a 和图 19b。

表 10：不同设置所选通道之间的关系。“ResB.1”和“DCB.2”分别表示残差块和深度卷积块。请注意，它们是我们模型中的前两个块。 $\mathbb{C}$ 表示所选通道的集合，$F = \{1e-5, 2e-5, 4e-5, 8e-5\}$ 表示图 7a 中不同的衰减率。A1B 和 A2B 分别表示在前 1 个和前 2 个块中避开特定通道。我们报告了每个集合的基数。

掩码层	教师模型	学生模型	$\bigcap_{i\in F}\mathbb{C}_i$	$\bigcup_{i\in F}\mathbb{C}_i$	$\complement_{A1B}\cap\mathbb{C}_F$	$\mathbb{C}_{A1B}\cup\mathbb{C}_F$	$\complement_{A2B}\cap\mathbb{C}_F$	$\mathbb{C}_{A2B}\cup\mathbb{C}_F$
ResB.1.mask1	192	64	44	83	2	145	3	144
ResB.1.mask2	192	64	47	83	0	147	2	145
DCB.2.mask1	192	64	44	86	56	94	7	143
DCB.2.掩码3	768	256	179	328	191	393	12	572

性能相当。这表明，不存在对实现卓越性能至关重要的通道。

G 可视化

图20展示了我们模型的重建结果。比较我们的小型模型（EVC-SS）和大型模型（EVC-LL），可以发现两者的重建结果几乎相同。这表明，我们的掩码衰减不会产生额外的伪影。与VTM相比，我们的方法编码和解码速度更快。

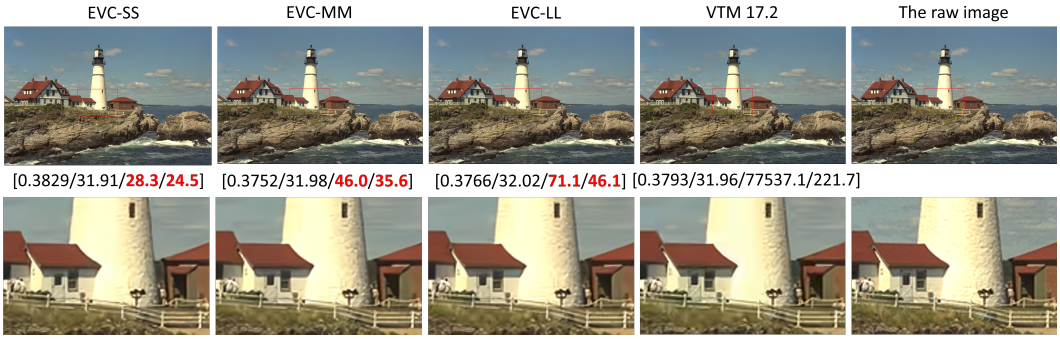


图20：我们模型重建结果的可视化。EVC-SS表示采用小型编码器和小型解码器的模型；M和L分别表示中型和大型。元组中的数字依次表示每像素比特数、峰值信噪比、编码时间（毫秒）和解码时间（毫秒）。请注意，延迟是在配备2080Ti GPU的计算机上测得的。我们的模型比VTM快得多。