A Reproduction of LiFteR

摘要

视频编解码器是视频流媒体的核心技术。尽管传统编解码器（如 AVC 和 HEVC）已取得显著成功，但基于深度神经网络（DNN）的学习编解码器因其在编码效率和用户体验质量（QoE）方面的优势，正逐渐受到关注。然而，现有学习编解码器通常采用紧密帧间参考设计，导致解码速度慢、帧率低，从而降低了用户体验。本文复现了一种基于松散帧间参考（LFR）的新型视频流媒体系统——LiFteR（NSDI'24, CCF A）。LiFteR 通过重新定义帧间参考关系，在学习编解码器中引入并行处理机制，显著提升了帧率。系统包含三个核心设计：（1）基于 LFR 的视频调度器，用于根据松散帧间参考调度视频数据；（2）基于 LFR 的学习编解码器，在提升编码效率的同时最小化对解码速度的影响；（3）流媒体支持模块，使学习编解码器能够在现有基础设施中实现自适应码率流媒体传输。复现结果与原文结论相近，与现有性能最佳的学习编解码器和传统编解码器相比，分别实现了高达 23.8% 和 19.7% 的用户体验质量提升，同时通过帧率配置实现了最高 3.2 倍的帧率提升。本文成功复现了 LiFteR 系统，并验证了其在实际应用中的有效性。

关键词：学习编解码器；流媒体传输；视频编解码

1 引言

1.1 选题背景

随着视频流媒体行业的快速发展，视频编解码器作为视频传输中的核心技术，直接影响着视频质量和带宽利用率。传统的视频编解码器（如 AVC、HEVC 等）基于手工设计的模块，虽然在视频压缩和解码方面取得了显著的成功，但其压缩效率和用户体验质量（QoE）的提升空间有限。近年来，基于深度神经网络（DNN）的学习编解码器逐渐崭露头角，展现出比传统编解码器更高的压缩效率和更好的视频质量。学习编解码器通过端到端的优化，能够更好地捕捉视频中的复杂特征，从而在低比特率下保持较高的视觉质量。这种潜力使得学习编解码器成为视频流媒体领域的研究热点。

然而，尽管学习编解码器在压缩效率上具有显著优势，其在实际应用中的表现却受到解码速度的限制。由于学习编解码器依赖于复杂的神经网络计算，其解码速度往往较慢，导致帧率低下，进而影响用户的观看体验。特别是在实时视频流媒体场景中，解码速度的不足可能导致视频卡顿、缓冲等问题，严重降低了用户体验。因此，如何在不牺牲压缩效率的前提下提升学习编解码器的解码速度，成为当前研究中的一个重要挑战。

1.2 选题依据

现有的学习编解码器虽然在压缩效率上表现出色，但其解码速度的瓶颈限制了其在视频流媒体中的应用。本文提出的 LiFteR 系统通过引入**松散帧间参考（LFR）**的设计，有效解决了这一瓶颈问题。LFR 通过重新定义帧间依赖关系，允许多个帧并行处理，从而显著提升学习编解码器的解码速度。具体来说，LiFteR 系统通过以下三个关键设计实现了这一目标：

1. **LFR 视频调度器：**基于 LFR 的调度机制，将视频数据分发到编解码器，平衡了编码效率和解码速度。

2. **LFR 学习编解码器：**通过自注意力机制和高度并行的运动估计模块，提升了编码效率，同时最小化了解码速度的影响。

3. **流媒体支持模块：**通过自适应比特率训练、帧率配置和增强的自适应比特率（ABR）算法，将学习编解码器无缝集成到现有的流媒体基础设施中。

实验结果表明，笔者复现的 LiFteR 系统在多种硬件配置和网络条件下均表现出色，相较于现有的学习编解码器和传统编解码器，在用户体验质量（QoE）上分别提升了 23.8% 和 19.7%，并且通过帧率配置实现了最高 3.2 倍的帧率提升。这些成果表明，LiFteR 不仅具有理论上的创新性，还具备实际部署的可行性。

报告复现文章被发表于计算机网络领域的顶级会议 NSDI（USENIX Symposium on Networked Systems Design and Implementation）上，属于 CCF A 类会议，具有较高的学术影响力。作为一篇系统论文，LiFteR 不仅提出了理论上的创新，还通过实际的系统实现和广泛的实验验证了其有效性。因此，本文具有较高的复现价值，能够为后续的研究和实际应用提供重要的参考。

1.3 选题意义

LiFteR [2] 系统提出的**松散帧间参考（LFR）**思想具有广泛的参考意义，能够为现有的学习编解码器提供通用的优化思路。现有的学习编解码器大多采用**紧密帧间参考（TFR）**的设计，即每一帧的解码依赖于其前一帧的完整重建。这种设计虽然能够最大化编码效率，但也导致了帧间的高度依赖，限制了并行处理的可能性，从而影响了解码速度。LiFteR 通过引入 LFR，打破了这种依赖关系，允许多个帧共享同一个参考帧，显著提升了解码速度。

LiFteR 的另一个重要意义在于其从视频媒体的特性出发，挖掘了潜在的优化空间。视频帧之间存在时间上的相似性，这种相似性不仅存在于相邻帧之间，也存在于非相邻但时间上接近的帧之间。LiFteR 通过利用这种时间上的相似性，重新定义了帧间依赖关系，既保持了较高的编码效率，又实现了解码速度的提升。这种设计思路不仅适用于 LiFteR 系统，还可以广泛应用于其他学习编解码器，为其解码性能的优化提供参考。

此外，LiFteR 系统通过自适应比特率训练、帧率配置和增强的 ABR 算法，成功将学习编解码器集成到现有的流媒体基础设施中。这一成果表明，学习编解码器不仅可以在理论上实现高效的视频压缩，还可以在实际的流媒体系统中得到有效应用。这对于推动学习编解码器在视频流媒体领域的广泛应用具有重要意义。本报告对 LiFteR 进行了复现和初步的优化，项目开源名称为 OpenLiFteR。

2 相关工作

2.1 自适应视频传输

自适应视频传输技术是近年来视频流媒体领域的研究热点之一。其主要目标是通过动态调整视频的码率、分辨率等参数，以适应网络带宽的变化，从而提升用户的观看体验（QoE）。现有的自适应视频传输技术主要分为推送式（push-based）和拉取式（pull-based）两种策略。推送式策略通过分析客户端的播放统计数据，从服务器端推送合适的视频码率给每个客户端 [3]。拉取式策略则依赖于客户端根据预测的带宽或缓冲区状态，主动从服务器下载合适码率的视频片段 [8]。此外，视频超分辨率（VSR）技术也被应用于提升视频流媒体的质量，通过增加下载视频片段的分辨率来增强用户的观看体验 [13]。这些技术虽然在提升视频流媒体质量方面取得了显著进展，但它们主要关注网络协议和客户端策略的优化，而本文则从视频编解码器的角度出发，提出了一种新的视频流媒体系统 LiFteR，能够与现有的自适应视频传输技术无缝集成。

2.2 传统编解码器

传统视频编解码器，如 H.264（AVC）[10] 和 H.265（HEVC）[9]，依赖于手工设计的模块，如基于块的运动估计、离散余弦变换和熵编码等。这些编解码器在视频压缩效率和计算复杂度之间取得了良好的平衡，广泛应用于视频流媒体系统中。然而，随着深度学习技术的快速发展，传统编解码器在压缩效率和视频质量方面的优势逐渐受到挑战。尽管传统编解码器在计算速度上具有优势，但其压缩效率的提升空间有限，难以满足未来高分辨率、高帧率视频流媒体的需求。

2.3 学习编解码器

近年来，基于深度神经网络（DNN）的学习编解码器逐渐成为视频压缩领域的研究热点。与传统的编解码器不同，学习编解码器通过端到端的优化，能够显著提升视频的压缩效率和视觉质量 [4]。例如，DVC [4] 和 RLVC [12] 等学习编解码器在相同的码率下，能够提供比传统编解码器更高的视频质量。然而，现有的学习编解码器普遍采用紧帧参考（TFR）的设计，导致解码速度较慢，难以满足实时视频流媒体的需求。尽管一些研究工作尝试通过优化神经网络结构或引入渐进式编码来提升学习编解码器的速度 [7]，但这些方法在硬件性能下降或帧率要求提高时，仍然难以满足实时需求。本文提出的 LiFteR 系统通过引入松帧参考（LFR）的设计，显著提升了学习编解码器的解码速度，能够灵活应对不同硬件和帧率要求的变化。

2.4 本文方法概述

论文提出了 LiFteR 系统，通过松散帧引用来改进深度学习编解码器在流媒体应用中的性能。传统的视频编解码器通常采用紧帧引用的设计，即每一帧的编码和解码都依赖于其前一帧的完全重建。这种设计虽然能够保证较高的编码效率，但在深度学习编解码器中，由于复杂的神经网络计算，TFR 会导致解码速度大幅下降，进而影响视频流媒体的帧率和用户体验。LiFteR 系统通过引入 LFR 的设计，允许帧引用不再局限于紧邻的前一帧，而是可以引用时间上接近的任意帧，从而实现了帧的并行处理，显著提升了编解码器的帧率。LiFteR 系

统主要由三个核心模块组成：基于 LFR 的视频调度器、基于 LFR 的神经编解码器以及流媒体支持模块。这些模块共同协作，解决了传统深度学习编解码器在流媒体应用中的瓶颈问题。

2.5 基于 LFR 的视频调度器

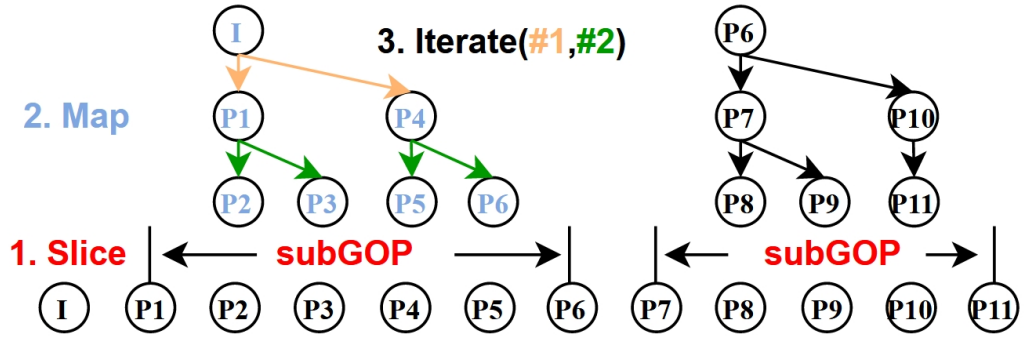

图 1. 基于 LFR 的视频调度过程

LiFteR 系统的核心创新之一是引入了基于 LFR 的视频调度器。该调度器通过构建一个二叉树结构的依赖图来管理帧之间的引用关系，从而平衡编码效率和解码速度。在传统的 TFR 设计中，帧的引用关系是线性的，每一帧都依赖于其前一帧，这导致了串行处理的瓶颈。而在 LFR 设计中，如图 1所示，帧的引用关系被重新定义，允许多个帧共享同一个参考帧，从而实现了帧的并行处理。具体来说，视频调度器通过二叉树的预序遍历来分配帧的引用关系，确保每一帧的引用帧在时间上尽可能接近，同时最小化引用成本（即帧与参考帧之间的时间差）。这种设计不仅减少了帧处理的迭代次数，还显著提升了帧率。此外，视频调度器还通过限制每棵树处理的帧数来控制内存消耗，确保系统能够在有限的硬件资源下高效运行。

2.6 基于 LFR 的学习编解码器

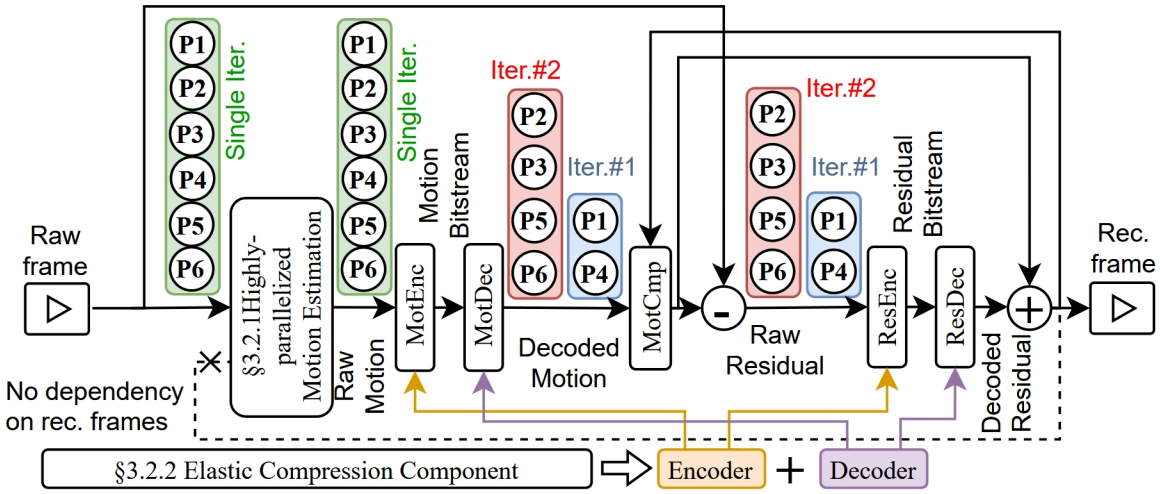

图 2. LFR 学习型编解码器的设计

为了在 LFR（低帧率）设计下保持较高的编码效率，如图 ??所示，LiFteR 系统提出了一种基于 LFR 的神经编解码器。该编解码器通过引入自注意力机制来捕获帧间的相关性，从而弥补由于松散引用导致的编码效率损失。具体来说，编解码器中的弹性压缩组件（ECC）结

合了卷积神经网络（CNN）和自注意力模块，以高效处理帧间的依赖关系。自注意力模块能够动态学习帧间的时空依赖关系，从而显著提升编码效率。此外，编解码器还采用了高度并行的运动估计模块，通过直接使用原始帧而非重建帧进行运动估计，大幅减少了帧处理的时间开销。这种设计不仅增强了编解码器的并行处理能力，还通过减少对重建帧的依赖，进一步降低了由于松散引用带来的编码效率损失。实验表明，基于 LFR 的神经编解码器在保持高编码效率的同时，能够显著提升帧率，尤其是在低端硬件上表现尤为突出。此外，ECC 模块的多帧时序估计进一步增强了模型对复杂运动模式的适应性，使其在高动态场景下仍能保持稳定的压缩性能。这种结合自注意力和并行化优化的设计，能平衡解码速度和计算开销。

2.7 流媒体支持模块

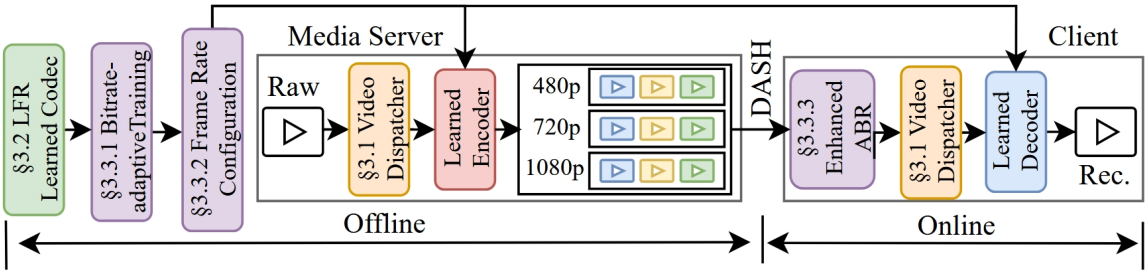

图 3. LiFteR 流媒体传输

为了使 LiFteR 系统能够无缝集成到现有的视频流媒体基础设施中，论文提出了流媒体支持模块。如图 3所示，该模块通过自适应比特率训练、帧率配置和增强的自适应比特率（ABR）算法，确保了 LiFteR 系统在不同网络条件下的稳定性和高效性。首先，LiFteR 系统通过压缩级别嵌入技术实现了自适应比特率训练。该技术将压缩级别作为输入传递给编解码器，使得系统能够在不存储多个编解码器版本的情况下，动态调整视频的压缩级别，从而适应不同的网络带宽。其次，LiFteR 系统通过子 GOP 探测技术实现了帧率配置。该技术通过调整子 GOP 的大小来适应不同的硬件性能，确保系统能够在不同硬件平台上达到目标帧率。最后，LiFteR 系统引入了虚拟缓冲区的概念，解决了传统 ABR 算法在学习编解码器中可能导致的保守决策问题。虚拟缓冲区通过跟踪已接收但未播放的视频片段长度，避免了由于解码速度较慢而导致的误判，从而提升了 ABR 算法的决策准确性。此外，流媒体支持模块还结合了网络状态预测技术，通过实时分析网络波动，进一步优化了比特率切换的平滑性和稳定性。原文实验验证了 LiFteR 系统在不同网络条件下均能提供优于传统编解码器和现有学习编解码器的 QoE 表现，尤其是在高动态网络环境下，其稳定性和效率表现尤为突出。

3 复现细节

论文作者没有公开源代码，报告复现代码参考了部分开源仓库。

3.1 与已有开源代码对比

3.1.1 编解码器框架

编解码器框架参考了经典学习编解码器 DVC [4] 的框架，参考仓库的开源地址为 https://github.com/ZhihaoHu/PyTorchVideoCompression。该框架利用深度神经网络直接学习视频的压缩表示，避免了传统视频编码中复杂的模块化设计。DVC 通过联合优化运动估计、运动压缩、残差压缩和码率控制等任务，实现了基于 pyTorch 的单帧学习编解码框架。

（*1*）*引用部分*：报告中代码直接使用了 DVC 开源仓库的视频数据集预处理代码。这个部分的代码主要为 shell 脚本和 python 脚本，用于将数据集中视频解码为编解码输入所需图片并保存于指定目录为训练和测试做准备。

（*2*）*原创部分*：报告所复现 OpenLiFteR 是多帧学习编解码框架，与 DVC 不同。OpenLiFteR 需要支持多帧输入输出和 LFR 视频帧数据结构。因此，报告对开源框架中的数据加载（DataLoader）、编解码器（VideoCompressor）、网络架构（net）、训练（train）、测试（test）都进行了原创编写，涵盖了框架运行过程的方方面面。

3.1.2 运动估计神经网络

报告所复现 OpenLiFteR 的光流估计模块参考了 SpyNet [6],参考仓库的开源地址为 https://github.com/sniklaus/pytorch-spynet。SpyNet 是一种基于深度学习的轻量级光流估计网络，由 Ranjan 等人在 2017 年 CVPR 会议上提出。它通过结合传统的光流计算方法和卷积神经网络（CNN），实现了高效且准确的光流估计。

（*1*）*引用部分*：报告复现代码直接使用了 SpyNet 开源仓库的预训练模型权重为训练阶段的模型初始权重。SpyNet 提供了预训练模型权重，该模型权重是在针对光流处理的数据集进行较大规模的光流估计训练。采用该模型权重能更好地提升 OpenLiFteR 在光流估计模块的泛化性。

（*2*）*原创部分*：报告复现代码在 SpyNet 基础上添加了对 LFR 数据结构的支持，实现了高度并行的光流估计模块（Tree_SpyNet）。该模块采用 LFR 帧间关系对输入视频帧进行处理，使得一次可以输入多个帧进行并行光流估计，改变了原本串行光流估计的结构，显著提升了光流估计模块的推理速度。

3.1.3 弹性压缩模块（ECC）

报告所复现 OpenLiFteR 的在 ECC 模块中参考了 TimeSformer [1]，参考仓库的开源地址为 https://github.com/facebookresearch/TimeSformer?tab=readme-ov-file。TimeSformer 是一种基于自注意力机制的卷积无关视频分类模型，由 Gedas Bertasius 等人提出。该模型将标准的 Transformer 架构扩展到视频领域，通过直接从帧级图像块序列中学习时空特征，实现了较好训练速度和准确率。

（*1*）*引用部分*：TimeSformer 的核心是基于自注意力机制的时空建模，其代码中已经实现了多头注意力机制（Multi-head Attention）和 Transformer 编码器块。OpenLiFteR 的网络实现中引用了这部分的代码。

（*2*）*原创部分*：OpenLiFteR 需要实现 ECC 模块去提取视频帧 GOP 的时空信息，因此报告复现代码在 Multi-head Attention 和编码器块基础上，新增了帧间层（IFL），并将其插

入到编码器和解码器的 CNN 层之后。这一修改通过编码视频帧之间的时空特征，进一步提升了学习编解码器对运动估计的压缩性能。

3.2 实验环境搭建

实验环境的搭建是确保 LiFteR 系统能够在不同硬件和网络条件下进行评估的关键步骤。为了全面测试 LiFteR 的性能，实验环境涵盖了硬件配置、视频数据集以及对比方法的选择。

(*1*) *硬件配置*：实验在配备 Intel(R) Xeon(R) Gold 6230 CPU（2.10GHz）和 NVIDIA GeForce RTX 4090 GPU 的硬件平台上进行，操作系统为 Ubuntu 24.04 LTS，编程环境为 Python 3.10.15。该配置提供了充分的计算能力和高效的深度学习支持，确保了实验的高效运行和准确评估。

(*2*) *视频数据集*：实验使用了两个视频数据集：UVG [5] 和 MCL-JCV [11]。这两个数据集合并后包含 37 个分辨率为 2K 的视频，帧率为 30fps，总时长约为 5 分钟。这些视频涵盖了多种场景和动作，确保了评估的多样性和全面性。LiFteR 的训练使用了 Vimeo-90k 数据集，该数据集包含 7 帧图像序列，其中第一帧为 I 帧，其余为 P 帧。训练过程中，I 帧使用 BPG 图像编解码器进行编码和解码，而 P 帧则使用 LiFteR 的学习型编解码器进行处理。训练图像的分辨率缩放为 256×256，并使用 Adam 优化器进行优化，学习率从 10^{-4} 逐步降低到 10^{-6}。

(*3*) *对比方法*：为了与 LiFteR 进行对比，实验选择了传统编解码器（x264 和 x265）。传统编解码器配置为三种模式：veryfast、medium 和 veryslow，分别代表不同的编码速度和压缩质量。学习型编解码器则按照其原始论文中的配置（GOP=7）进行设置，以确保公平比较。

3.3 创新点

3.3.1 运动估计预训练优化

原文中关于运动估计的设计主要在于通过类似二叉树的图结构进行帧间依赖关系的优化，从而实现依赖开销和依赖深度的优化。

在基于光流和残差的学习编码器中，视频的压缩可以简单概括为四个步骤，运动估计、运动估计压缩、残差估计和残差压缩。其中，残差估计的输入还依赖于运动估计的输出。因此，我们可以认为，在基于光流和残差的学习编码器中，运动估计的质量决定了视频压缩的质量基础。然而，原文中并没有提到如何对运动估计模型进行合理的训练和微调，仅仅给出了基本的模型工作原理。

因此，本报告对 LiFteR 的运动估计进行了预训练优化，有效减少了编解码器的训练难度的同时提升了学习编解码中运动估计模块的泛化能力。在报告复现版本中，运动估计模块在训练过程中会先根据经过大规模训练的预训练模型的权重进行初始化。随后再结合 LFR 和新的视频训练数据进行后续的模型权重微调。

3.3.2 编解码器网络结构优化

在原文中，LiFteR 学习编解码器通过 ECC 模块中的 IFL 层对多个视频帧的时序信息进行提取，以提供更好的压缩性能。IFL 的核心是通过自注意力机制（self-attention）捕捉帧间和帧内的依赖关系。然而，IFL 依赖于 TimeSformer 中较为复杂的网络结构，会增加较大的计算量和复杂度。

因此，报告复现对 IFL 的模型结构进行了进一步的优化，并将其加入到现有的压缩流程中。优化主要集中在减少计算复杂度和增强特征表示两个方面。(1) 为了减少计算复杂度，首先通过 reduction_ratio 降低通道数，从而减少自注意力机制的计算量。接着，在降维后的特征上进行多头注意力计算，进一步降低计算复杂度。最后，采用轻量化的前馈网络以增强特征表示能力。(2) 在增强特征表示方面，报告复现引入了多尺度特征融合方法，在 IFL 中融合多尺度特征，以捕捉不同时间尺度的帧间依赖关系。这种多尺度特征融合不仅提升了特征的表达能力，还增强了模型对不同运动模式的适应性，从而进一步提升了整体压缩性能。

4 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

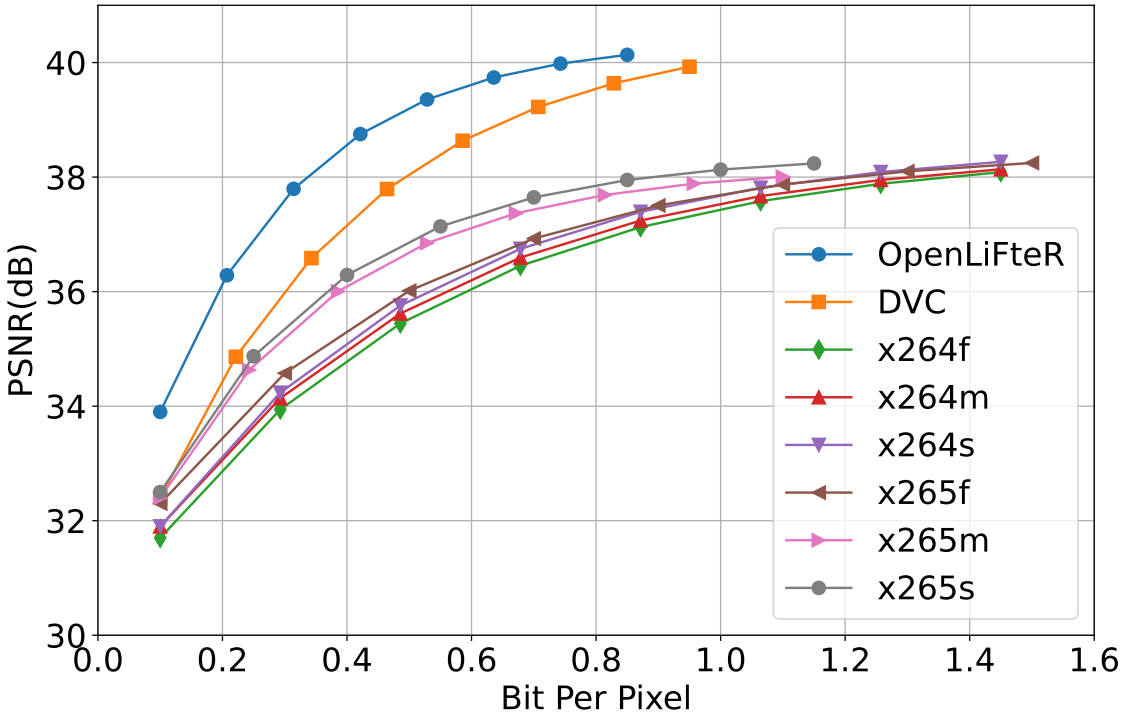

图 4. 不同编解码器压缩性能对比

如图 4所示，LiFteR 在编码效率方面展现出显著优势，全面超越了现有的基线方法。具体而言，在传统视频编码系统中表现最优的 x265-veryslow 编码器需要 0.59 bpp（比特每像素）才能实现 38.20 dB 的 PSNR（峰值信噪比），而 LiFteR 仅需 0.28 bpp 即可达到相同的重建质量，带宽使用量减少了 52.5%。这一显著的性能提升主要得益于 LiFteR 框架中两个创新性设计：首先，LFR（低帧率）技术通过引入帧间依赖关系，有效降低了冗余信息的传输；其次，ECC（高效上下文捕捉）模块中的自注意力机制能够精确捕捉长距离依赖关系，从而提高了压缩效率。值得注意的是，尽管 LFR 视频分发器和高并行化运动估计机制可能在一定程度上增加了帧与其参考帧之间的时空距离，但通过精心设计的补偿机制，这些技术的优势足以抵消其潜在的负面影响，最终实现了更高的压缩效率和更优的重建质量。值得思考的是，若没有加入 ECC 模块带来的性能补充，LFR 可能会导致较为明显的压缩性能下降。

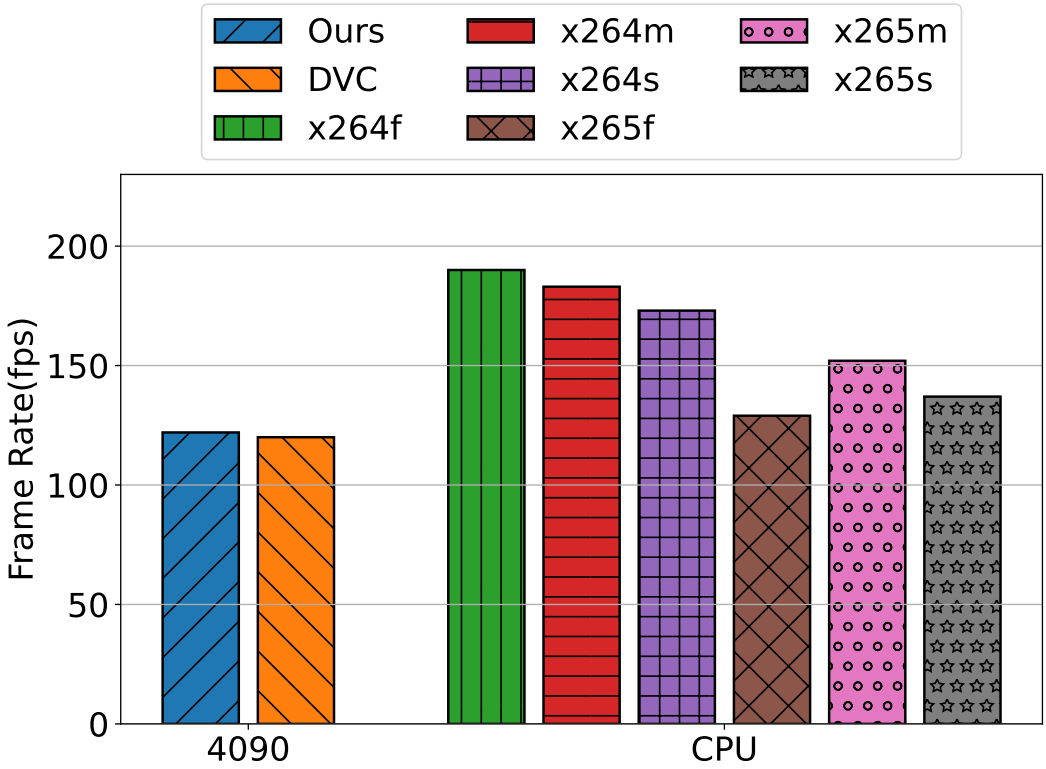

图 5. 不同编解码器解码性能对比

在图 5中，笔者对使用学习编解码器的系统在不同 GPU 上的帧率与使用传统编解码器的系统在 CPU 上的帧率进行了对比。实验结果表明，报告复现系统在不同硬件平台上均能实现稳定的实时帧率，展现了其良好的硬件适应性。然而，可以预见的是 DVC 在低端硬件上的帧率表现会明显逊于 LiFteR，尤其是在计算能力有限的设备上。目前两者吞吐量相当是由于计算设备的算力充足导致。尽管 LiFteR 通过并行化设计和优化显著提升了帧率，但其性能仍未能超越传统视频编解码器。传统编解码器在速度优化方面经过长期工程化改进，尤其是在 CPU 上的高效实现，使其在高帧率场景中仍具有明显优势。LiFteR 的帧率表现与其设计目标一致，即在保证编码效率的同时，尽可能提升解码速度，但在极端性能优化方面仍需进一步改进以缩小与传统编解码器的差距。未来，通过更深入的硬件协同优化和算法改进，LiFteR 有望在帧率性能上取得更大突破。

5 总结与展望

本文详细介绍了基于松散帧间参考的视频流媒体系统 LiFteR 的设计与实现。LiFteR 系统通过引入松散帧间参考机制，显著提升了学习编解码器的解码速度，解决了传统学习编解码器在实时视频流媒体中帧率低下的问题。系统核心设计包括基于松散帧间参考的视频调度器、学习编解码器以及流媒体支持模块，分别负责视频数据的调度、编码效率的提升以及自适应码率流媒体传输的实现。此外，报告还详细描述了复现过程中的技术细节，包括编解码器框架、运动估计神经网络、弹性压缩模块（ECC）的优化，以及实验环境的搭建。

值得一提的是，本文并没有对 LiFteR 进行更加深入的消融实验，这是由于如今已经出现一系列更加前沿的条件学习编解码器（Conditional Coding），其基本方法不同于 LiFteR 所采用的基于残差光流的学习编码器。因此对 LiFteR 的消融分析并不能真正地使得松帧间引用

这一思想在学习编解码器这一领域提供最前沿的洞见。从目前的方法来看即使我们能通过一系列的模型稀疏化配合松散帧间参考提升解码帧率，但其效率仍然成倍慢传统手工编码器在 GPU 上的运行速度。因此，学习编解码器要在未来真正走入市场和用户的视野，其实时性能仍然是一个值得研究的问题。

参考文献

[1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, volume 2, page 4, 2021.

[2] Bo Chen, Zhisheng Yan, Yinjie Zhang, Zhe Yang, and Klara Nahrstedt. {LiFteR}: Unleash learned codecs in video streaming with loose frame referencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 533–548, 2024.

[3] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. C3:{internet-scale} control plane for video quality optimization. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, pages 131–144, 2015.

[4] 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.

[5] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 297–302, 2020.

[6] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170, 2017.

[7] Oren Rippel, Alexander G Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14479–14488, 2021.

[8] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM transactions on networking*, 28(4):1698–1711, 2020.

[9] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[10] Yiqi Tew and KokSheik Wong. An overview of information hiding in h. 264/avc compressed video. *IEEE transactions on circuits and systems for video technology*, 24(2):305–319, 2013.

[11] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *2016 IEEE international conference on image processing (ICIP)*, pages 1509–1513. IEEE, 2016.

[12] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte. Learning for video compression with recurrent auto-encoder and recurrent probability model. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):388–401, 2020.

[13] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 645–661, 2018.