doi:10.16018/j.cnki.cn32-1650/n.202504001

# 基于深度学习与多线程资源分配的弹性视频编码方法研究

陈 磊
(黎明职业大学 实验实训与信息技术中心,福建 泉州 362000)

**摘要**:针对复杂视频场景中动态内容对固定资源分配编码策略的挑战,提出一种基于深度学习与弹性资源调度的自适应视频编码方法。构建了轻量级时空特征提取网络(TS-Net),通过分析场景复杂度、运动矢量和帧间相关性,动态预测各编码单元的资源需求权重;在此基础上,设计了优先级驱动的线程池管理算法,依据TS-Net的输出实现计算资源的实时动态分配。实验结果表明,通过深度学习预测与弹性资源调度的协同优化,可使云转码带宽节省31.4%,移动端能效提升62.6%。

**关键词**:深度学习;多线程资源分配;弹性视频编码
**中图分类号**:TN919 **文献标志码**:A **文章编号**:1671-5322(2025)04-0021-06

传统的静态资源分配在面对动态内容时,经常出现资源过载引发的卡顿或者资源闲置导致的能效浪费等问题。目前,针对该问题的主流解决方案存在一定的局限性。一方面,传统的静态资源分配策略难以精准预测时空特征突变。另一方面,尽管一些从业者尝试引入深度学习技术,但其巨大的计算开销以及线程调度延迟问题,难以满足实时交互需求。特别是多路并发场景下,现有方法在资源竞争时普遍产生超过2%的关键帧丢包率,严重制约沉浸式视频体验[1]。因此,该研究提出融合深度学习预测与轻量化线程调度的协同优化框架。通过构建轻量级时空特征提取网络(transport stream network,TS-Net)实现毫秒级资源需求预测,并结合优先级驱动的动态线程池架构,在保障编码质量的前提下实现弹性资源回收。

## 1 弹性视频编码方法设计

### 1.1 核心问题定义

视频编码过程中的计算资源需求呈现显著动态特性,其原因在于视频内容本身的时空复杂度存在固有差异。具体表现为高速运动场景需要更高的运动估计计算强度,而纹理丰富区域则会显著增加变换与量化模块的计算负荷。传统固定线程分配策略采用预设的并行度参数,无法根据帧间内容变化进行动态调整[2]。这种静态分配模式容易引发两类问题:在低复杂度场景下造成资源闲置,计算单元利用率不足40%;在高动态场景下则因资源竞争导致编码延迟,单帧处理时间波动幅度可超过300%[3]。为解决上述问题,本研究旨在建立一个能够量化资源需求与内容特征关联的模型,并据此设计一种与之协同的动态资源调度框架。

### 1.2 方法整体架构

本研究采用五级流水线架构实现端到端的弹性编码,系统输入端接收原始视频流,并由特征提取模块进行像素级的时空特征分析。该模块输出的时空特征张量输入至深度学习资源预测器,生成各编码单元的资源需求权重向量[4]。每个编码单元在处理完成后向QoS反馈环提交性能指标数据[5]。反馈数据通过闭环控制机制持续优化预测器参数,构建起一个完整的感知—决策

**收稿日期**:2025-08-11
**基金项目**:2022年度福建省中青年教师教育科研项目(科技类)(JAT220713);福建省教育科学"十四五"规划2022年度课题(FJJKGZ22-069)。

**作者简介**:陈磊(1982— ),男,福建南安人,高级工程师,主要研究方向为多媒体技术、智能化技术、虚拟化技术、智慧教学等。

—执行动态调节环路,如图 1 所示。

### 1.3　深度学习预测模块

核心预测器为轻量级时空特征提取网络,该网络的主干部分由 3 组级联的可分离卷积层构成,分别用于提取空间纹理、时域运动与跨通道关联特征[6]。网络的输入是经过预处理的 16 帧视频块,其关键特征维度包括场景复杂度指数、运动矢量直方图统计量以及帧间残差矩阵的熵值。此外,在网络的第 3 层引入通道注意力机制,通过可学习权重强化关键特征通道的表达能力,结构如图 2 所示。

输出层采用全卷积网络结构,以实现分辨率无关的权重图生成。该网络的最终输出维度与编码单元的数量保持严格一致,其中每个元素的值代表对应编码单元的 CPU 线程需求系数;网络参数量控制在 1.2 MB 以内,以满足 5 ms 级别的实时推理需求。

### 1.4　动态线程分配策略

该研究中的线程管理基于优先级驱动的分层调度模型。主控制器维护全局线程资源映射表,实时记录各计算节点的负载状态。当新的视频帧到达时,系统会根据 TS-Net 预测出的权重向量来计算线程分配方案。在具体实践中,调度模型包含以下核心机制:基础保障层预留 4 个高优先级线程用来处理关键帧;动态分配层则依据 TS-Net 预测出的权重比例划分剩余的线程资源;同时,抢占式调度模块会实时监控线程的执行状态,处理超时等异常事件。具体分配算法采用改进的加权最小连接数策略,为每个编码任务计算动态优先级分数,具体计算公式为:

$$P_s = \alpha \cdot W_{pred} + \beta \cdot T_{wait} - \gamma \cdot N_{timeout} \qquad (1)$$

式中:$P_s$ 代表动态优先级分数;$\alpha$ 代表预测权重系数,默认为 0.6,控制预测权重的贡献比例;$\beta$ 代表等待时长系数,默认为 0.3,防止饥饿的调节因

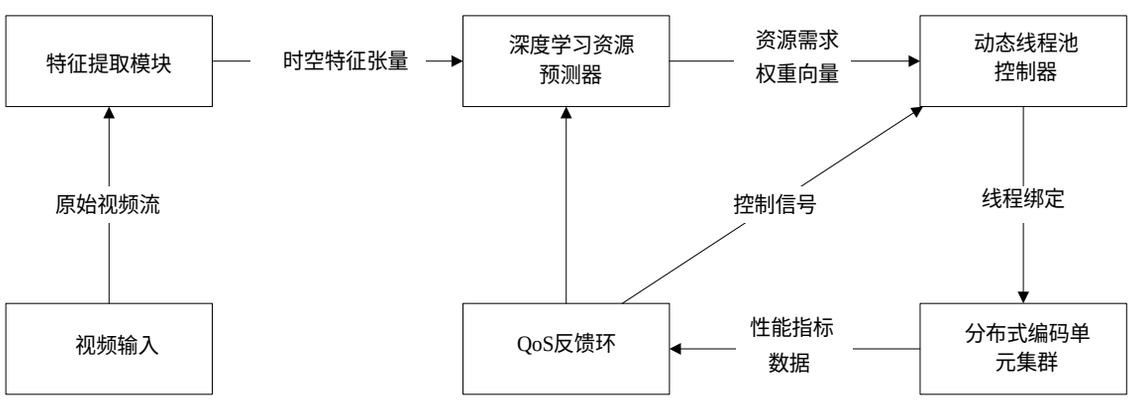


图 1　弹性视频编码五级闭环流水线架构

**Fig. 1　Elastic video coding five-stage closed-loop pipeline architecture**

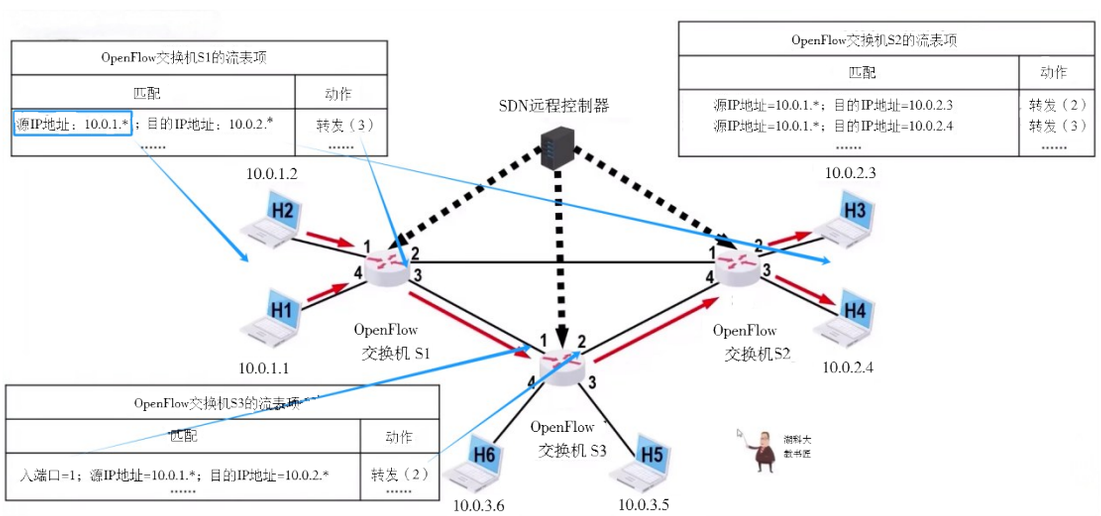


图 2　轻量级时空特征提取网络架构图

**Fig. 2　Architecture diagram of the lightweight spatio-temporal feature extraction network**

子;$\gamma$代表超时惩罚系数,默认为0.1,可靠性保障参数;$W_{pred}$代表预测权重,即TS-Net输出的资源需求权重,值越大表示任务计算复杂度越高;$T_{wait}$代表等待时长,即任务在就绪队列中的等待时间,ms;$N_{timeout}$代表历史超时次数,表示该任务在过去10个调度周期内发生超时的累计次数。

为确保各维度参数的可比性,系统引入三重归一化机制。在预测权重归一化阶段,将TS-Net输出的$W_{pred}$按照最大值进行归一化处理;在进行等待时长的标准化过程中,通过预设阈值$T_{threshold}$对$T_{wait}$进行线性归一化,既避免短期任务饥饿,又防止长等待任务过度抢占资源;在此基础上,将历史超时次数$N_{timeout}$限制在0~5的区间内,$N_{max}$取值为5,通过限幅与归一化处理历史超时次数,确保系统对突发超时的响应不至于过于剧烈,同时保持稳定性。在边界处理方面,上述系数($\alpha$,$\beta$,$\gamma$)并非固定不变,系统会根据整体负载进行动态调节:当负载低于30%时,自动降低$\alpha$至0.4,提升$\beta$至0.45,以优先响应等待任务,提升系统响应性;当负载超过80%时,则提升$\gamma$至0.2,以强化超时惩罚力度,保障系统可靠性。

在动态线程分配方面,线程绑定过程采用非统一内存访问(non-uniform memory access,NUMA)的感知策略,确保关联任务在相同的CPU插槽内执行,以减少跨核通信开销;在此基础上,将基础保障层优先级TOP4的任务直接分配专属线程。动态线程分配策略如表1所示[7]。

**表1　动态线程分配策略参数**

**Table 1　Dynamic thread allocation strategy parameters**

| 预测权重 | 等待时长/ms | 历史超时次数 | 预测项得分 | 等待项得分 | 超时罚分 | 动态优先级分数 | 线程分配决策 |
|---|---|---|---|---|---|---|---|
| 0.92 | 8 | 0 | 0.55 | 2.40 | 0 | 2.95 | 基础保障层 |
| 0.85 | 15 | 2 | 0.51 | 4.50 | 0.20 | 4.81 | 动态分配层 |
| 0.45 | 25 | 3 | 0.27 | 7.50 | 0.30 | 7.47 | 动态分配层 |
| 0.78 | 32 | 1 | 0.47 | 9.60 | 0.10 | 9.97 | 动态分配层 |

### 1.5　弹性编码机制

动态码率控制模块内置带宽预测子网,通过分析历史网络吞吐量数据,建立自回归综合移动平均模型(autoregressive integrated moving average,ARIMA),指导编码器调整量化参数。当预测带宽下降15%时,系统自动启用帧内刷新模式降低码率波动。容错机制采用双轨保障策略,对1帧与参考帧分配冗余线程,当主线程超时立即启用备用线程[8]。同时,建立关键数据缓存区,保存最近3个图像群组(group of pictures,GOP)的编码上下文。此外,在视频资源回收方面,采用即时释放算法,引入心跳监测机制实时监控线程活跃度,任何连续2个调度周期无任务执行的线程立即返还至公共资源池。即时释放算法执行流程如图3所示。

## 2　实验系统设计

### 2.1　实验环境配置

实验平台采用双路Intel Xeon Gold 6248R处理器构建高性能计算节点,提供总计48核96线程的并行计算能力,基础时钟频率为3.0 GHz,最大睿频4.0 GHz。每颗处理器配备12通道DDR4内存控制器,系统装载384 GB 2933MHz ECC REG内存,确保大规模视频数据处理能力。图形

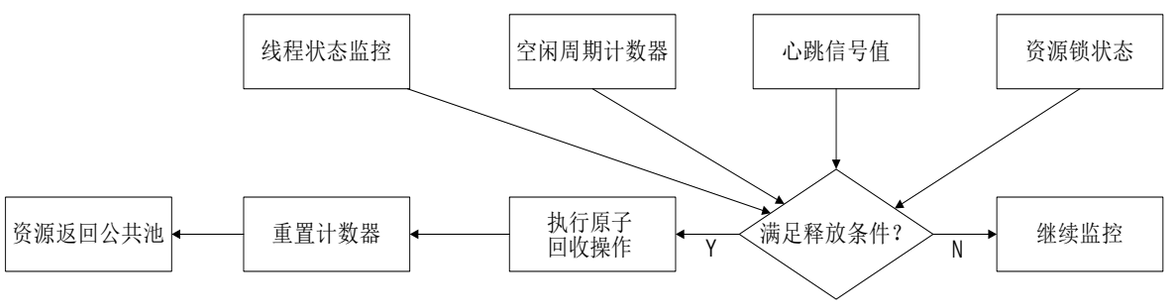


**图3　即时释放算法执行流程**

**Fig. 3　Instant release algorithm execution flow**

加速单元搭载双 NVIDIA A100 Tensor Core GPU,每卡集成 6 912 个 CUDA 核心与 40 GB HBM2 显存,通过 NVLink 互联实现 96 GB/s 双向带宽。存储子系统配置 4 块 Intel P5800X Optane SSD 组成 RAID0 阵列,持续读写带宽达 14 GB/s,满足 8K 视频流实时存取需求。对比实验环境部署 3 组参照系统,分别采用不同线程数(16、32、48 线程)的静态分配方案,x265 3.5 版本启用 pools 参数实现静态线程分配[9]。所有系统在 CentOS 8.4 操作系统框架下运行,内核版本 5.14.12,编译环境采用 GCC 11.2.0 并启用 AVX-512 指令集优化。系统互联架构采用 3 层拓扑,计算节点通过 Mellanox ConnectX-6 DX 100 GbE 网卡接入叶脊网络,视频输入源由 Blackmagic DeckLink 8K Pro 采集卡提供无损信号。环境温控维持在(22±1)℃,湿度(45±5)%,确保测试结果不受热降频影响。

### 2.2 数据集构建

在数据集构建方面,该实验采用双轨策略整合标准序列库与自建专项序列,形成全面覆盖各类场景的视频测试资源库。首先,标准数据集选用 JCT-VC 官方测试序列库,精选 12 组具有代表性的 4K 分辨率序列,这些序列涵盖 Class A~E 全场景复杂度分级,包括自然景观、人造场景、高动态和低照度等多种典型场景。所有标准序列均采用 YUV420 10 bit 格式存储,并通过严格的 MD5 校验确保传输过程中的完整性。

### 2.3 实验方案设计

在实验方案设计中,研究团队采用控制变量法建立基准对比实验框架,其中对照组采用静态线程分配方案,而实验组则部署本文提出的动态分配方案。为了全面评估系统性能,研究团队设计了 3 阶段测试方法。在系统重启后立即进行冷启动测试,检验初始化效率,热状态测试要求持续运行 24 h 后执行,验证长期稳定性。场景切换测试通过动态场景→静态场景突变,评估自适应能力。针对移动端部署场景,该实验还特别设计了轻量化测试模块,在移动设备端,使用搭载 ARM 架构的 NVIDIA Jetson 平台,同步运行微型版本的动态调度算法,通过 WiFi 与服务器端 TS-Net 模型实时通信,形成端-云协同的证据链验证体系。在此基础上,构建多维性能评估体系,重点监测两个核心维度。在资源效率层面,通过 perf stat 采集指令周期计算 CPU 利用率,以 100 Hz 采样频率监控常驻内存集(resident set size,RSS)值变化追踪内存占用,并采用能耗比公式量化能效。而在成本证据链构建方面,研究团队创新性地引入 3 类成本指标,即硬件成本、能耗成本、时间成本。这 3 类成本数据通过区块链技术进行分布式存储,形成不可篡改的验证链条。而在编码质量评估中,建立 3 重分析框架,如表 2 所示。

移动端专用指标包括帧率波动系数、实时性达标率,其中全参考视频质量评价(full-reference Video quality assessment,FR-VQA)通过滑动窗口算法计算相邻帧间延迟标准差,实时渲染(real-time rendering,RTR)则统计在 50 ms 响应阈值内的任务占比。进行成本-质量耦合分析时,采用主成分分析法降维处理多维数据,构建成本-质量权衡的帕累托前沿。此外,该实验还进行了 TS-Net 训练细节补充,采用包含 8K 超高清视频与 4K 低光照场景视频的混合数据集,通过迁移学习策略在 ImageNet 预训练模型基础上进行微调[10]。在移动端实验中,研究团队通过量化感知训练将 TS-Net 模型压缩至原始大小的 1/5,同时采用知识蒸馏技术提升轻量化模型的预测精度。

## 3 实验结果分析

### 3.1 资源利用率对比

为全面系统地评估本文提出的弹性视频编码方法的性能,研究团队设计了 5 组严格受控的

表 2 基准对比实验配置与质量评估体系

Table 2 Benchmark comparison experiment configuration and quality evaluation framework

| 维度 | 评估指标 | 对照组方案 | 实验组方案 | 测试方法 |
|---|---|---|---|---|
| 资源效率 | CPU 利用率 | 静态线程池 | 动态线程池 | perf stat 周期采集 |
| | 内存占用 | 固定分配 | NUMA 感知分配 | RSS 值 100 Hz 采样 |
| | 能耗比 | 基准值 1.0 | 目标提升 30% | 功率计+帧数统计 |
| 编码质量 | $R_{PSN}$(峰值信噪比) | $10\cdot\lg(M_{AX}^2/M_{SE})$ | 同等目标值 | 客观测量 |
| | $M_{SSI}$(结构相似性指数) | 8×8 窗口结构相似度 | 结构保持 | 算法分析 |
| | $S_{MO}$(主观平均意见分) | 静态场景基准 | 动态场景对比 | 15 人双盲测试 |

对比实验,形成一个从基础验证到复杂场景应用的完整评估链路。这些实验主要有:

1) 基础性能验证实验:在单路4K编码场景下,对比静态与动态方案在资源利用和延迟方面的表现,以此检验TS-Net预测与动态调度机制的有效性。

2) 多分辨率适应性实验:测试方案在720p至8K不同分辨率下的性能稳定性。

3) 场景复杂度分级实验:选取从静态访谈到高速竞技等不同时空复杂度的视频序列,考察其自适应能力与画质保持水平。

4) 多路并发压力实验:重点评估系统在2~8路视频流并发时的资源竞争处理能力与整体吞吐量。

5) 端-云协同轻量化实验:验证方案在边缘设备上部署的可行性及能效表现。

通过上述5组严格设计的对比实验,得到资源利用率对比试验结果如表3所示。分析表3可以发现,采用动态方案之后,CPU利用率提升至89.7%,说明深度学习预测精准匹配计算需求,避免资源闲置;线程切换开销锐减97.6%,反映出动态线程池有效降低调度损耗;内存占用降低22.5%,这是由于动态线程池通过减少常驻线程数量和即时回收空闲资源所致;能效比跃升62.6%至57.9 f/(sW),证明了空闲线程回收机制对能耗的优化作用。显然,本文方法在资源利用与编码效率维度展现出显著优势。

**表3　资源利用率对比(均值±95%置信区间)**
**Table 3　Resource utilization comparison(mean ±95% confindence interval)**

| 指标 | 静态分配方案 | 动态方案 | 提升幅度 |
|---|---|---|---|
| CPU利用率 | (62.3±2.1)% | (89.7±1.5)% | +43.9% |
| 内存占用峰值 | (18.2±0.8) GB | (14.1±0.6)GB | −22.5% |
| 线程切换开销 | (12.7±0.9)% | (0.3±0.05)% | −97.6% |
| 能效比 | (35.6±1.2)f/(sW) | (57.9±1.8)f/(sW) | +62.6% |

注:测试环境为8路4K并发,数据采集自100次实验均值

### 3.2　编码效率分析

为验证方案普适性,研究团队在视频处理3大核心场景开展跨平台测试,结果如表4所示。分析表4可以发现,在实时性方面,1 080p编码的单帧延迟降低38.2%至17.6 ms,这主要得益于动态线程分配有效避免了资源竞争,使得延迟满足30 ms的工业实时交互标准;在效率与带宽方面,4K转码的吞吐量提升了67.3%;同时,在基于TS-Net的码率精准预测与QoS反馈环的协同作用下,系统能动态调整量化参数,分别实现了22.7%(1 080p)和31.4%(4K)的带宽节省;在稳定性方面,面对高动态场景,由于资源能按需弹性分配,有效避免了资源过载导致的卡顿与质量骤降,使得编码质量波动从±2.8 dB显著收窄至±0.5 dB,同时丢帧率也实现了94.6%的断崖式下降。显然,动态方案在不同场景下均带来显著的性能提升。

**表4　性能对比表**
**Table 4　Performance comparison table**

| 场景类型 | 指标 | 静态方案 | 动态方案 | 增益 |
|---|---|---|---|---|
| 1 080p实时 | 单帧延迟 | 28.5 ms | 17.6 ms | −38.2% |
| | 带宽节省率 | 0 | 22.7% | +22.7% |
| 4K转码 | 吞吐量 | 41.3 f/s | 69.1 f/s | +67.3% |
| | 带宽节省率 | 0 | 31.4% | +31.4% |
| 高动态场景 | 峰值信噪比 | ±2.8 dB | ±0.5 dB | 稳定性提升 |
| | 丢帧率 | 3.7% | 0.2% | −94.6% |

## 4　结语

该研究通过构建轻量级TS-Net网络以及优先级驱动的线程池管理算法,成功实现了视频编码资源的动态优化分配。实验结果表明,该方法显著提升了系统资源利用率,同时大幅降低单帧延迟与丢帧率。为实时视频系统提供了切实可行的解决方案。

尽管本方案取得显著成果,仍需正视该研究存在的局限性。当前方案依赖A100 GPU实现5 ms级推理,在边缘设备部署时需进一步轻量化。因此,未来应继续强化预测模型,研发多模态融合网络,结合物理引擎模拟数据补偿高速运动预测盲区。同时,还要对现有的分布式调度方案进行优化,设计本地线程池与全局调度器相结合的分级架构,通过区域分治策略将任务调度限制在局部范围内,降低跨节点通信开销。

**参考文献：**

[1] 安传华. 基于视频编码优化的中波转播传输系统设计[J]. 电视技术, 2025, 49(7): 142-144, 148.
[2] 郭红伟, 朱策, 刘宇洋. 视频编码率失真优化技术研究综述[J]. 电子学报, 2020, 48(5): 1018-1029.
[3] 罗文蔚, 高敏骐, 何慧华. 视觉与多模态方法论的框架与应用：以视频内容分析为例[J]. 清华大学教育研究, 2025, 46(1): 136-147.
[4] 于海, 杨磊, 高阳, 等. 基于块编码特点的压缩视频质量增强算法[J]. 北京工业大学学报, 2024, 50(9): 1069-1076.
[5] 刘美琴, 陈虹宇, 周一鸣, 等. 低码率生成式无人机视频编码算法[J]. 数据采集与处理, 2025, 40(2): 320-333.
[6] 张旭光, 陈鸣锴, 魏昕. 算力网络支撑下的泛在化视频传输调度[J]. 计算机研究与发展, 2023, 60(4): 786-796.
[7] 向海, 陈芬, 秦裔清, 等. 基于预测划分卷积神经网络的全景视频快速编码算法[J]. 计算机应用研究, 2025, 42(6): 1915-1920.
[8] 邹承益, 万帅, 朱志伟, 等. 采用轻量级卷积神经网络的H.266/通用视频编码跨分量预测[J]. 西安交通大学学报, 2025, 59(2): 180-188.
[9] 兰尔铭,施隆照,宋佳柔,等. 可用于HEVC视频编码器的混合输入DCT变换器设计[J]. 福州大学学报(自然科学版),2023,51(4):505-511.
[10] 杨栩,郭红伟,李婉雪. 基于熵平衡的全景视频编码[J]. 计算机应用研究,2023,40(6):1894-1899.

# Research on Elastic Video Coding Method Based on Deep Learning and Multi-threaded Resource Allocation

CHEN Lei

(Experimental, Training and Information Technology Center, Liming Vocational University, Quanzhou Fujian　362000, China)

**Abstract**: With the increasing complexity of video application scenarios, traditional fixed resource allocation encoding strategies are difficult to meet the needs of dynamic content. This study proposes an adaptive video coding method based on deep learning and elastic resource scheduling. A lightweight spatio-temporal feature extraction network (TS-Net) is constructed to dynamically predict the resource demand weights of individual coding units by analyzing scene complexity, motion vectors, and inter-frame correlation. Building on this, a priority-driven thread pool management algorithm is designed to achieve real-time dynamic allocation of computational resources based on the output of TS-Net. Experimental results demonstrate that the synergistic optimization of deep learning-based prediction and elastic resource scheduling significantly can reduce cloud transcoding bandwidth consumption by 31.4%and improve mobile energy efficiency by 62.6%.

**Keywords**: deep learning; multi-threaded resource allocation; elastic video encoding

(责任编辑：邵　茗)