arXiv:2601.21362v1 [cs.NI] 29 Jan 2026

ViTMAlis：面向延迟关键型移动视频分析的视觉 Transformer

Miao Zhang，*IEEE*会员，Guanzhen Wu、Hao Fang、Yifei Zhu，*IEEE*会员，Fangxin Wang，*IEEE*会员，Ruixiao Zhang 和 Jiangchuan Liu，*IEEE*会士，

摘要—**边缘辅助移动视频分析（MVA）应用正日益从采用基于卷积神经网络（CNN）的视觉模型，转向采用基于视觉变换器（ViT）的模型，以利用其卓越的全局上下文建模能力和泛化能力。然而，在延迟关键型移动视频分析场景中部署这些先进模型面临着重大挑战。传统的基于CNN的卸载范式中，网络传输是主要瓶颈；与之不同，基于ViT的系统受制于显著的推理延迟，尤其是在密集预测任务中，高分辨率输入会加剧ViT固有的二次计算复杂度。为应对这些挑战，我们提出了一种面向以视觉变换器为骨干的密集预测模型的动态混合分辨率推理策略，能够在运行时灵活权衡速度与准确率。在此基础上，我们推出ViTMAlis，这是一种原生支持ViT的设备到边缘卸载框架，可根据网络状况和视频内容动态调整，以协同降低传输延迟和推理延迟。我们在商用移动设备和边缘设备上实现了功能完备的ViTMAlis原型。大量实验表明，与最先进的准确率优先、内容感知和延迟自适应基线方法相比，ViTMAlis显著降低了端到端卸载延迟，同时提升了用户感知的渲染准确率，为下一代移动智能奠定了实用基础。**

索引词—**视觉变换器、移动视频分析、资源管理**

I. 引言

THE 移动计算平台和边缘基础设施的迅猛发展，推动了移动视频分析（MVA）应用的激增，应用范围涵盖增强现实（AR）和自主移动感知等领域 [1]–[3]。这些应用高度依赖先进的计算机视觉模型，以提供实时、交互式的用户体验。现有的多视角分析系统主要基于卷积神经网络（CNN）构建，但卷积神经网络长期以来的主导地位已正日益受到视觉变换器（ViT）[4]，这一类源自Transformer架构 [5]的骨干网络的挑战。凭借更强的全局上下文建模能力和更弱的归纳偏置，ViT 在各种视觉任务中取得了最先进的性能 [6]–[8]。它们能够处理灵活的输入尺寸，并在不同任务和模态间实现泛化，因此已成为基础视觉模型

Miao Zhang、Guanzhen Wu、Hao Fang 和 Jiangchuan Liu 任职于加拿大西蒙菲莎大学计算机科学学院。Yifei Zhu 任职于中国上海交通大学全球学院。Fangxin Wang 任职于中国香港中文大学（深圳）科学与工程学院（SSE）及智能未来网络研究院（FNii）。Ruixiao Zhang 任职于美国伊利诺伊大学厄巴纳-香槟分校计算机科学系。

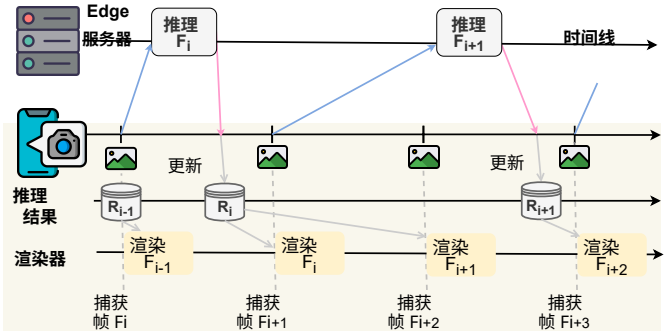

图 1. 边缘辅助移动视频分析。

[9], [10]。具体而言，ViT 现已成为视觉基础模型 [10], [11] 和视觉语言模型 [12], [13]，中的图像或视频编码器，为下一代移动应用提供核心智能。

然而，将 ViT 集成到时延要求严苛的移动视频分析应用中面临着重大挑战。与能够容忍数秒到数分钟处理延迟的一般视频分析任务不同 [14]–[16]，移动视频分析应用通常涉及实时用户交互，因此对时延极为敏感。如图 1 所示，主流的 MVA 边缘辅助范式 [1] 通常会将移动设备捕获的视频帧卸载到边缘服务器进行推理，再将结果返回移动设备进行端侧渲染（例如，在实时相机画面中出现的对象上叠加边界框或标签）。这一卸载过程构成了交互式移动视频分析固有的实时反馈环。环路中即使引入微小的额外延迟，也可能导致输出过时，无法跟上用户的实时画面，从而降低视觉体验。

鉴于 MVA 对时延的严格要求，先前的工作主要致力于通过系统级优化来尽量降低端到端（E2E）卸载延迟。由于这些系统主要依赖 CNN骨干模型，与 ViT 模型相比，这类模型相对轻量，因此通常假设服务器端推理延迟相对稳定，并非端到端卸载延迟的主要组成部分 [1], [17], [18]。因此，它们的优化工作主要集中在减少通过移动网络传输视频数据所产生的延迟。然而，对于ViT骨干模型，这一假设已不再成立，因为模型推理已成为主要瓶颈。

MVA 中执行的视觉任务主要是密集预

测任务（例如目标检测和语义分割），这类任务通常需要高分辨率输入图像来保持准确率 [18]。然而，ViT 中的多头注意力机制会引入推理时间和内存复杂度，而这些开销会随着输入词元数量（因此也随着图像分辨率）呈二次方增长。即使采用窗口注意力等架构优化 [7],，推理延迟仍然显著。例如，我们的测量结果表明，在配备英伟达 RTX5090 GPU 的边缘服务器上处理一帧 1080p帧时，基于 ViT 骨干网络的密集预测模型*ViTDet-L*[7], 推理延迟约为 281 毫秒，对应的帧率仅为 3.56 帧每秒（FPS）。

为降低推理延迟，研究者探索了多种推理时加速技术，利用ViT与输入无关的特性，在推理速度和准确率之间取得平衡 [19]。尽管在ViT的中间层剪枝或合并信息量较低的视觉令牌，已在图像和视频分类等任务中展现出有效性 [20]–[22],，但这些方法往往会造成不可逆的视觉细节损失，从而显著降低密集预测任务的准确率。混合分辨率令牌化技术 [23], [24],允许每个词元表示不同大小的图像区域，从而加速推理，因此已成为一种颇具前景的替代方案。然而，这类技术在密集预测任务中的潜力仍未得到充分探索。

为弥合以ViT为骨干的视觉模型与MVA之间的差距，我们首次探索了系统层面的设计机会，利用视觉变换器的独特特性来应对其实际部署挑战。我们的设计针对动态移动网络条件和多样化的视频内容进行定制，以支持在MVA场景中高效部署以ViT为骨干的视觉模型。具体而言，我们提出了ViTMAlis，这是一种原生支持ViT的卸载系统，旨在优化边缘辅助MVA的资源效率。ViTMAlis的核心是一套专为ViT设计的混合分辨率压缩与推理策略，可协同降低网络传输和服务器端推理延迟。我们的贡献总结如下：

- 我们为以视觉变换器为骨干的密集预测模型提出了一种动态混合分辨率推理策略。该策略采用精心设计的区域划分方法以生成可由预训练 ViT 主干网络无缝处理的混合分辨率视觉令牌。为兼容需要全分辨率特征的密集预测头，该方法还加入了特征重建机制，可根据混合分辨率令牌动态还原全分辨率特征图，从而灵活权衡模型推理速度与准确率。
- 我们开发了ViTMAlis，这是一个原生支持ViT的多视角分析系统，它融合了我们的混合分辨率推理策略，并根据网络状况和视频内容为每个卸载帧动态调整配置。通过将面向准确率的下采样区域选择 方法与内容感知的性能估计相结合，ViTMAlis从系统全局视角优化端到端卸载延迟和渲染准确率。

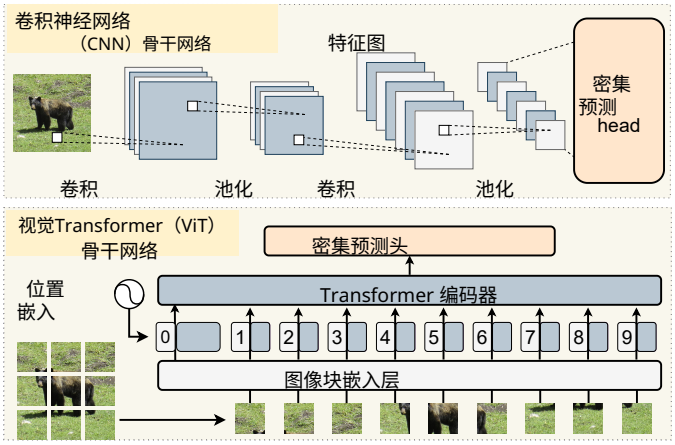

图 2. 卷积神经网络和视觉变换器采用不同策略从图像中提取视觉特征。

- 我们在商用移动设备和边缘设备上实现并部署了功能完备的ViTMAlis原型。该实现证实了所提系统的实际可行性，表明它可以无缝集成到当今的移动计算平台中。
- 我们使用在多种移动场景中捕获的视频进行了大量实验，并在不同移动网络条件下评估了该系统。结果表明，与现有的准确率优先、内容感知和延迟自适应基线方法相比，ViTMAlis 显著降低了端到端卸载延迟，同时提高了用户感知的渲染准确率，验证了其在时延关键型 MVA 场景中的有效性。

二、背景与动机

MVA 涉及实时应用计算机视觉模型分析移动设备摄像头捕获的视频 [2], [17], [18], [25]。由于移动设备的计算资源有限且受到散热限制，难以在本地运行准确但资源消耗大的模型来进行实时推理。边缘辅助 MVA 范式通过将移动设备上的视频帧卸载到边缘服务器进行推理，解决这一挑战。在此场景中，返回的推理结果的质量（即远程推理准确率）和新鲜度都至关重要。如图 1 所示，卸载帧的新鲜度由其端到端卸载延迟决定，该延迟包括帧编码和上传时延、服务器端解码和推理延迟，以及可忽略不计的结果下载延迟。现有的边缘辅助方案主要采用相对轻量的 CNN 骨干模型，通常假设模型推理延迟固定，因此侧重于帧压缩策略，以平衡网络传输时延与推理准确率 [1], [2], [17]。然而，使用 ViT骨干模型时，仅优化网络传输时延并非最优方案。

ViT 作为通用骨干网络。如图 2 所示，当原始视觉 Transformer 用作通用视觉特征提取的骨干网络时，会将输入图像划分为固定大小（例如，

16×16像素）、互不重叠的图像块，对每个图像块进行线性嵌入，添加位置嵌入，然后将得到的嵌入序列输入 Transformer 编码器 [4]。与卷积神经网络不同，原始视觉 Transformer 在整个网络中保持单尺度特征图，从而捕获整幅图像中的全局依赖和长程关系。

MVA 中的密集预测任务，例如目标检测和实例分割，需要对多个尺度的目标实例进行细粒度理解，因此必须使用高分辨率输入图像。这与图像或视频分类任务不同，后者在较低分辨率（例如， 224×224） [26], [27]下也能有效运行。不幸的是，ViT 的 Transformer 编码器中的自注意力机制具有随输入序列长度呈二次方增长的计算复杂度，严重限制了其扩展到高分辨率输入的能力 [24]。例如，将输入图像分辨率从 360p 提高到 1080p，会使 ViTDet-L 的推理延迟增加 7.2× 倍（从 39 毫秒增至 281 毫秒）。这些观察结果表明，需要专门的系统设计，以缓解密集预测任务中采用 ViT 作为骨干网络的模型所产生的显著推理延迟。

ViT 的推理时加速。 在 ViT 中，嵌入后的图像块以类似于自然语言处理（NLP）中词元的方式进行处理。自注意力机制在理论上支持任意长度的输入词元序列。因此，只要适当缩放位置嵌入，预训练ViT模型就能自然处理不同分辨率的图像 [4]。这种输入无关特性为在推理过程中动态调整词元数量和排列方式、平衡运行时推理速度与准确率创造了独特机会。

一种普遍采用的方法是在 ViT骨干网络 的中间层中剪枝或合并信息量较低的词元 [21]–[28]–[30], 通过完全移除词元或合并相似词元来减少计算量。另一类方法则探索在图像预处理阶段执行令牌缩减，以混合分辨率令牌化 [23], [24]取代原始视觉Transformer的均匀网格标记化。它们为重要的图像区域分配更多词元，为不太重要的区域分配更少词元，同时保持完整覆盖

同时更高效地利用有限的令牌预算 [24]。

尽管前景可观，但现有的大多数加速技术都是为图像和视频分类任务开发的，这类任务会将分类头直接连接到 [CLS]词元。在这种情况下，视觉令牌通过 [CLS]词元聚合为单一的全局表示，用于图像级或视频级预测，因此词元的精确空间排列不那么重要。相比之下，密集预测头需要骨干网络提供多尺度特征图（即特征金字塔），以有效处理不同尺度的对象。这些特征图通常构建在规则二维网格上，以保持空间对齐；对于基于网格对齐的多尺度特征进行运算的下游预测头而言，这一点至关重要。然而，令牌剪枝会彻底移除词元，破坏均匀的空间覆盖；令牌合并和混合分辨率分词则会生成代表不同大小图像区域的词元，从而改变令牌粒度。这些做法都会破坏规则网格布局，

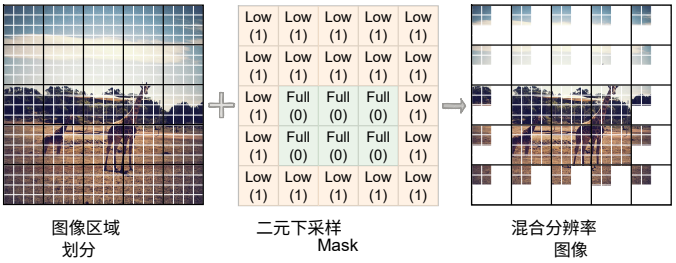

图3. 推理兼容区域划分。

使其难以与密集预测头集成。

III. 动态混合分辨率推理

为克服上述挑战，我们提出了一种动态混合分辨率推理策略，专为密集预测模型设计。与剪枝或合并等方案相比，我们优先采用混合分辨率令牌化，因为它能够在早期节省计算开销，同时无需修改架构或重新训练预训练 ViT 骨干网络。以下小节将详细介绍关键设计，以应对将混合分辨率推理扩展到密集预测模型时面临的独特挑战。

A. 推理兼容区域划分

第一个挑战是确定每个词元所表示的图像区域的空间范围。在标准的预训练 ViT 骨干网络中，每个词元对应一个固定大小的图像块，通常为 16×16 像素 [4]。为与预训练模型保持兼容，下采样区域的尺寸必须是预期图像块大小的整数倍。此外，为密集预测任务设计的 ViT 变体通常会在中间的 Transformer 块中用窗口注意力取代全局注意力，以提高处理高分辨率输入时的效率 [7], [31]。窗口注意力会将输入特征图划分为互不重叠的局部窗口，并将自注意力计算限制在各个窗口内。为在这一机制下保持结构一致性，下采样区域必须在每个窗口内保持空间对齐和一致的图像块粒度。

为解决这一问题，我们提出了一种推理兼容区域划分方法，使其与预训练 ViT骨干网络的图像块化和窗口注意力机制保持一致。图3展示了一个示例：在图像上叠加了一个粗粒度网格（以黑色显示）。每个黑色网格单元称为决策区域，覆盖$r \times r$ 个图像块，其中每个图像块（以白色显示）包含 16×16个像素。下采样决策在决策区域级别进行，从而生成二值区域级下采样掩码。标记为低分辨率的区域随后按 d的因子进行下采样，使表示这些区域所需的图像块数量（以及词元数量）减少 d^2倍。为确保与 ViT骨干网络的窗口注意力计算兼容， r 被策略性地设置为 $w \times d$，其中 w 是 ViT骨干网络用于计算窗口注意力的窗口大小。在所示示例中，两者均

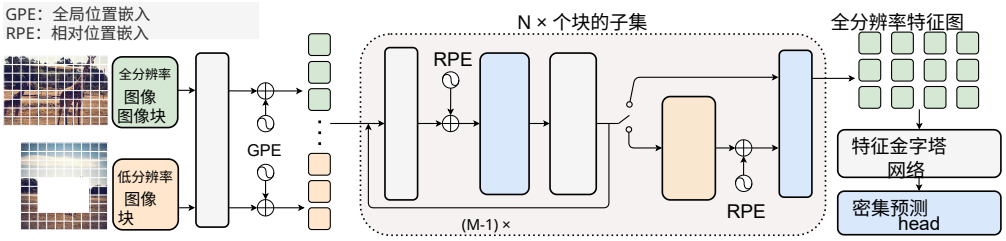

图 4. ViTDet 动态混合分辨率推理工作方式示例。

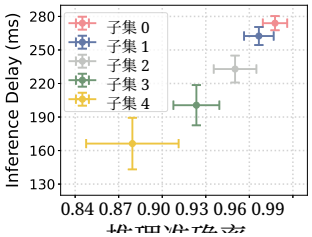

图 5. RPs 的影响。

w 和 d均设置为 2，从而得到 $r = 4$。这一选择确保下采样区域中的词元恰好适配每个2×2 窗口，使混合分辨率序列能够被划分到窗口中，并在不产生结构不匹配的情况下无缝重构。

B. 灵活分辨率恢复

尽管最近的一项研究 [7] 已经证明，从原始 ViT 主干网络生成的单尺度特征图构建特征金字塔是可行的，但混合分辨率词元化的引入会产生不同粒度的词元，从而破坏这种规则的空间布局。如图3所示，下采样后，图像块之间原有的二维空间关系不再得以保留。因此，需要额外的机制来恢复空间对齐，并构建密集预测头所需的连贯特征图。

为应对这一挑战，我们提出了一种灵活特征图构建方法，可在ViT骨干网络内的可配置阶段，选择性地恢复与下采样图像区域相对应的词元分辨率。该设计能够重建密集预测头所需的空间结构化特征图，同时保持计算效率。图4展示了该方法如何与ViTDet集成。在ViTDet中，预训练 ViT 主干网络被均匀划分为 N 个子集（默认 $N = 4$），每个子集包含 M个Transformer 块。在每个子集中，前（$M - 1$）个块计算局部窗口注意力以降低计算开销，而最后一个块则应用全局注意力，以实现跨窗口信息传播。

除非另有说明，本文始终使用 ViTDet 系列中的预训练 ViTDet-L [32] 模型，每个子集包含 $M = 6$ 个块。与处理均匀网格图像块的原始 ViTDet-L 不同，我们的方法会分别将低分辨率和全分辨率图像块投影为标记嵌入。随后，在两组词元上添加全局位置嵌入，再将它们拼接起来并输入 Transformer 块进行处理。原则上，可以在 ViT 骨干网络的任意深度对低分辨率词元进行上采样，以构建全分辨率特征图。然而，我们发现，在每个子集中，于最后一个窗口注意力块与后续全局注意力块之间进行恢复，能在准确率和推理延迟之间取得最佳平衡。因此，我们将该位置指定为每个子集的候选恢复点 *(RP)*。

图 5 展示了在不同子集的候选 RP 处进行恢复时的权衡。在这项初步实验中，不包含对象的决策区域会按 2 倍比例下采样。“子集 0”是一种特殊情况：下采样后的图像块会先上采样至全分辨率，再输入骨干网络。如图所示，在骨干网络中越早进行恢复，准确率越高，但推理延迟的降低幅度越小。值得注意的是，我们的混合分辨率推理策略是一种自适应输入 方案，可针对每个 Transformer 块动态调整输入张量形状。该策略保留了 ViT 骨干网络架构，且不会引入额外的可训练参数，因此能够无缝集成到任何预训练 ViT 骨干网络中。

IV. 原生 ViT 的 MVA 卸载框架

A. 网络与推理联合优化

在 MVA 中，网络传输时延和服务器端视觉模型推理时延是端到端卸载时延的主要组成部分。以网络为中心的方法将模型推理时延视为常数，并专注于通过各种帧压缩技术降低前者。例如，基于感兴趣区域（RoI）的混合质量压缩通过对 RoI 采用较高质量（如较低的量化参数），同时对背景区域进行激进压缩，从而降低带宽 [17]。然而，这些方法只解决了传输时延问题。帧到达服务器后，通常会先解码为统一分辨率再进行处理，因此推理成本并未改变。相反，以推理为中心的 方法往往忽视网络传输方面的挑战，假定输入图像已在服务器上就绪。虽然这些方法采用混合分辨率令牌化等技术来加速服务器端推理 [23]，[24]，但此类方案通常针对独立图像设计，未能利用连续视频帧之间的时间局部性进行更有效的优化。

现有的卸载框架难以同时降低网络传输延迟和模型推理延迟，且往往会造成显著的精度损失。这主要是因为这些框架依赖以 CNN 为骨干网络的视觉模型，而这类模型要求输入图像具有统一分辨率，以保留生成的特征图中的空间关系。因此，现有方法通常只能以统一的下采样分辨率传输帧 [2]，[25]，，这会因重要图像区域中细粒度细节的丢失而导致显著的精度下降

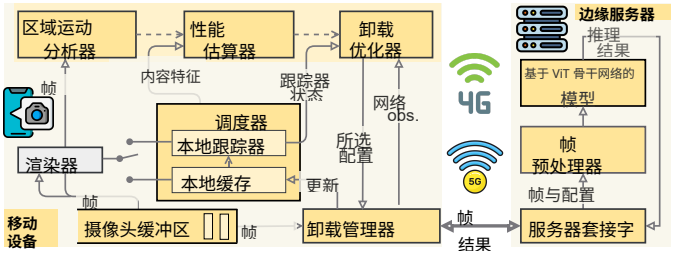

图 6. ViTMAlis 的系统概述。

。相比之下，以 ViT 为骨干网络的模型具有灵活的架构，不再受统一分辨率约束，从而为网络与推理的联合优化开辟了新的途径。

为了充分发挥这一潜力，我们认为，区域特定分辨率的分配不应推迟到服务器端，而应在卸载准备阶段由移动设备启动这一决策过程。通过在传输前对信息量较低的区域进行下采样，我们可以同时降低网络带宽消耗和服务器端计算负载，从而缩短这两个阶段的时延。为保持推理准确率，设备必须评估各区域与服务器端推理任务的相关性，并据此分配分辨率。

B. 系统概述

我们首先概述 ViTMAlis，这是一个基于我们提出的混合分辨率推理策略构建的原生ViT的MVA卸载系统。如图 6 所示，ViTMAlis 遵循典型的边缘辅助MVA范式。移动设备的摄像头以固定速率持续捕获视频帧，并将其存储在 摄像头缓冲区 中。渲染器以相同速率轮询调度器，以获取与待渲染帧对应的分析结果。在边缘辅助 MVA 中，移动网络带宽有限且动态变化，因此维持实时卸载速度仍是一项挑战。当某帧的端到端卸载延迟超过连续两帧之间的间隔（例如，30 FPS 视频中的 33 毫秒）时，新捕获的帧在渲染时必须依赖先前卸载帧的过时推理结果。为缓解这一问题，通常会使用轻量级设备端跟踪器，将这些过时结果传播到后续帧，从而在无法获取最新远程推理结果时减少精度下降 [2], [17], [25]。

ViTMAlis 中的调度器由两个关键组件组成：一个本地缓存，用于存储最近卸载帧的远程推理结果；另一个是本地跟踪器，它利用轻量级跟踪技术为新捕获的帧生成最新结果。当查询某一帧时，调度器会先检查本地缓存。如果缓存的远程推理结果对应于所查询的帧，则返回该结果；否则，使用本地跟踪器的结果。收到新的远程推理结果后，卸载管理器 会更新本地缓存。此更新会触发本地跟踪器重新初始化，以便与最新结果同步，并安排后续帧进行卸载。

对于每个要卸载的帧，我们将一个配置$\boldsymbol{c} \in \mathcal{C}$定义为 $\boldsymbol{c} = (\tau_d,\ \lambda,\ \beta)$，其中 $\mathcal{C}$是所有候选配置的集合。这里，τ_d 确定决策区域的下采样掩码，λ 表示混合分辨率帧的压缩质量参数，β 指定混合分辨率推理在 ViT 骨干网络中的恢复点。

帧卸载工作流如下。区域运动分析器 首先计算每个决策区域的运动值，并将其转发给性能估计器。该估计器将运动值与本地跟踪器提供的区域内容特征相结合，以预测每个候选配置的压缩数据大小和预期推理准确率。最后，卸载优化器 结合这些预测、近期网络观测值和当前跟踪器状态，选择最优卸载配置，记为 $\boldsymbol{c}^\star$。

选定 $\boldsymbol{c}^\star$ 后，卸载管理器根据 $\tau_d^\star$生成二值下采样掩码。利用该掩码生成一张混合分辨率图像（如图3所示），然后使用质量参数 $\lambda^\star$对其进行压缩。压缩后的帧及其配置将传输到服务器。服务器收到后，根据下采样掩码提取低分辨率和全分辨率图像块，构建混合分辨率输入，并使用恢复点 $\beta^\star$执行推理。随后，推理结果将返回移动设备，用于本地状态更新。本文接下来的部分将以目标检测 为案例研究，详细介绍 ViTMAlis 的关键系统设计，因为它是现有 MVA 研究中受到最广泛研究的任务。

C. 面向精度的区域选择

令 $\mathcal{R}$表示所有决策区域的集合。对于每个计划卸载的帧，我们需要确定二值下采样掩码 $\mathbf{B} \in \{0,1\}^{|\mathcal{R}|}$，其中第 j 个条目在决策区域 j 经过下采样时取值为 1，目标是在降低端到端卸载延迟的同时，尽可能减少推理精度下降。由于最优的 $\mathbf{B}$选择会随视频内容而变化，因此必须为每一帧动态调整掩码。从理论上讲，我们可以独立决定是否对每个区域进行下采样，从而得到 $2^{|\mathcal{R}|}$ 种可能的组合。由于探索如此庞大的决策空间对于实时应用而言计算开销过高，我们提出一种启发式方法来简化选择过程。

为此，我们分析 MVA 帧的内容特征。由于摄像机移动性，MVA 中的视频内容通常表现出显著的动态性和结构复杂度。为量化每个决策区域 j 的特征，我们定义了两个指标：运动(m_j) 和任务相关性(ρ_j)。运动 m_j 由区域运动分析器使用轻量级背景减除模型 [33] 进行估计，其计算方式为区域 j 中的前景像素（即表现出显著运动的像素）数量与帧中前景像素总数之比。任务相关性 ρ_j 反映该区域对于下游预测任务的重要性

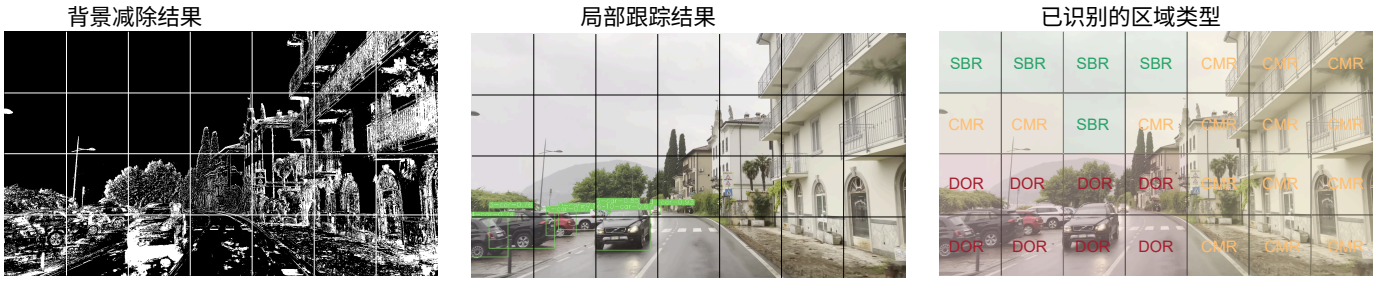

图7. 基于背景减除和局部跟踪识别区域类型的示例。

，并根据特定任务的输出进行计算。对于目标检测， ρ_j 对应于区域目标密度，定义为边界框部分或完全与区域 j 重叠的对象所占比例。

基于这些指标，我们观察到一帧内的区域可以系统地划分为三类：(1)静态背景区域（*SBR*），其特征为低运动和低任务相关性，在较长时间内几乎保持不变，且受相机运动的影响极小（例如天空或海洋）；(2) 相机运动引起的区域（*CMR*），其特征为高运动但低任务相关性，包含在现实世界中静止的对象（例如树木或建筑物），但这些对象因摄像机移动性而看似在移动；以及 (3) 动态对象区域（*DOR*），其特征为高运动和高任务相关性，包含能够独立运动的对象（例如行人或车辆）。我们利用运动阈值 δ^m 和任务相关性阈值δ^ρ 来识别给定区域 j的类别。如果 $m_j < \delta^m$，则将该区域分类为SBR。否则，如果 $\rho_j > \delta^\rho$，则将其分类为DOR。不满足上述任一条件的区域被分类为相机运动引起的区域。图7展示了这种区域类型划分的一个示例。

鉴于对 DOR 进行下采样可能导致严重的精度下降，我们严格保留这些区域的全分辨率。为指导下采样策略，我们定义一个下采样类型变量 $\tau_d \in \{0,1,2\}$，其取值分别指定要进行下采样的目标区域集合：不进行下采样（0）、仅对 CMRs 进行下采样（1），或对 CMRs 和 SBRs 均进行下采样（2）。类型 0 对应于高质量视频帧卸载，通常适用于网络和计算资源充足的场景。我们排除仅对 SBRs 进行下采样的选项，因为实证观察始终表明，与仅对 CMRs 进行下采样相比，该选项的效果并不理想。这是因为 SBRs 通常由低频内容构成，即使在全分辨率下也能高效压缩；而对高频 CMRs 进行下采样，在对准确率影响相近的情况下，往往能显著节省更多带宽。

令 $\boldsymbol{\phi}$ 表示当前帧的区域类型映射，其中每个元素 $\phi_j \in$ {SBR、CMR、DOR} 指定决策区域 j的类型。一旦确定 τ_d ，便生成二值下采样掩码$\mathbf{B}$ ，即$\mathbf{B} = f_{\tau_d}(\boldsymbol{\phi})$，其中 f_{τ_d} 将所有属于 τ_d指定的下采样目标类别的区域赋值为 1。关键在于，即使τ_d固定，$\mathbf{B}$ 也会随帧变化，因为 $\boldsymbol{\phi}$ 会随相机运动动态演变。

D. 内容感知性能估计

令 $N_d = \sum_{j\in\mathcal{R}} B_j$ 表示由$\mathbf{B}$指定的下采样决策区域数量。我们在目标移动设备上进行的大规模离线分析表明，帧编码延迟 T^{enc} 同时取决于下采样区域的数量 N_d 和压缩质量参数 λ。因此，我们使用性能分析结果 $T^{\text{enc}}(N_d, \lambda)$ 直接估算目标设备上的编码延迟。尽管解码延迟 T^{dec} 会因配置而略有变化，但与其他延迟组成部分相比仍然较小。因此，我们使用服务器离线分析测得的均值来近似 T^{dec} 。

服务器端模型推理延迟 T^{inf} 取决于恢复点 β 和输入令牌序列长度，而后者由 N_d 决定。对于固定的 β，性能分析结果显示， N_d 与推理延迟之间近似呈线性关系。因此，我们使用一组线性模型 $\text{LM}_\beta^{\text{inf}}(N_d)$ 来估算 T^{inf} ，其中每个模型对应一个特定的恢复点，并根据目标服务器上的离线分析结果进行参数化。关于网络延迟，由于视觉模型推理结果通常很小（仅数百字节），结果下载延迟主要由传播而非传输决定。因此，我们将总传播延迟（包括帧上传和结果下载）近似为往返时间（RTT），记为 L^{net}。

帧上传延迟主要由压缩帧大小和当前网络吞吐量决定。虽然可以使用现有方法估算平均上传吞吐量 U [34]–[36], ，但估算压缩数据大小 $S(\boldsymbol{c})$更具挑战性，因为它取决于 N_d、λ以及特定的帧内容。为此，我们使用所有下采样区域中的运动值总和（记为 $m^d = \sum_{j\in\mathcal{R}} B_j\, m_j$）作为下采样区域内容特征的代理指标。为捕捉全局内容动态，我们将归一化帧运动 m^f 定义为前景像素数与图像总像素数之比。随后，我们训练一个轻量级多层感知机（MLP）来估算 $S(\boldsymbol{c})$:

$$\hat{S}(\boldsymbol{c}) = \text{MLP}^{\text{size}}(\tau_d, N_d, m^d, m^f, \lambda) \tag{1}$$

在估算出所有时延组成部分后，性能估计器将端到端卸载延迟 $T(\boldsymbol{c})$ 预测为:

$$\hat{T}(\boldsymbol{c}) = \hat{T}^{\text{enc}}(N_d, \lambda) + \frac{\hat{S}(\boldsymbol{c})}{\hat{U}} + \hat{T}^{\text{dec}} + \text{LM}_\beta^{\text{inf}}(N_d) + \hat{L}^{\text{net}} \tag{2}$$

服务器端模型的推理准确率 $A(\boldsymbol{c})$ 受 $\boldsymbol{c}$ 中所有配置参数的影响，也受与密集预测任务相关的帧内容影响。为捕捉这种内容依赖性，我们从区域任务相关性 ρ_j 中提取两个统计量：均值 $\mu(\rho_j)$ 和标准差 $\sigma(\rho_j)$，二者均基于所有决策区域 $j \in \mathcal{R}$ 计算。具体而言， $\mu(\rho_j)$ 反映任务相关内容在图像中的分布范围，而 $\sigma(\rho_j)$ 衡量其空间集中程度，表明任务相关内容是均匀分散的，还是聚集在特定区域内。这些统计量与运动特征和配置参数一同输入轻量级多层感知机模型，以估算每个 $\boldsymbol{c}$ 的推理准确率：

$$\hat{A}(\boldsymbol{c}) = \text{MLP}^{\text{acc}}(\boldsymbol{c}, N_d, m^d, m^f, \mu(\rho_j), \sigma(\rho_j)) \qquad (3)$$

E. 卸载配置选择

算法1详细说明了卸载优化器如何为每个计划卸载的帧选择配置 $\boldsymbol{c}^\star$。算法首先根据 m_j 和 ρ_j 将每个决策区域 j 分类为 SBR、CMR 或 DOR（第 1 行）。随后，它会评估每个候选配置 $\boldsymbol{c}$ 的预估端到端卸载延迟$\hat{}T(\boldsymbol{c})$和推理准确率$\hat{}A(\boldsymbol{c})$（第 2-11 行）。在我们的实现中，卸载优化器通过对最近两次卸载操作的观测值取平均，来估算平均上传吞吐量和网络延迟。这种短窗口平均方法开销低，可作为适用于目标场景的基线；此外，我们的框架也支持按需直接集成更先进的预测方法（例如基于 Transformer 的模型）。

根据预估的端到端卸载延迟和推理准确率，卸载优化器会找出位于帕累托前沿 上的配置子集（第 12 行）。其中，每个配置在帕累托意义上都是最优的，即改善某一性能目标（例如降低时延）必然会损害另一项目标（例如降低准确率）。如果前沿上只有一个配置，那么它同时实现了最低的端到端卸载延迟和最高的推理准确率，因此会被选中（第 13-14 行）。否则，选择将根据系统当前状态进行。

具体而言，卸载优化器评估两项运行时指标： η，即自上一卸载帧以来的帧间隔 ，以及 κ，即跟踪保持率，定义为自本地跟踪器最近一次重新初始化以来被持续跟踪的对象所占比例。高 η 或低κ 表明更迫切需要进行低时延卸载，以确保渲染结果的新鲜度。因此，若 η超过陈旧度阈值 δ^η （即自上次卸载以来间隔较长），或 κ 低于保持率阈值 δ^κ（即跟踪保持较差），卸载优化器会选择估计端到端卸载延迟最低的配置，以便快速进行本地状态更新（第16-17行）。在所有其他情况下，它会选择帕累托前沿上拐点[37] 处的配置（第18-19行）。该点体现了端到端卸载延迟与推理准确率之间最有利的平衡，超过该点后，其中一个目标的进一步改善将导致另一个目标显著下降。

算法1： 卸载配置选择

数据： m^f, m_j, ρ_j, $\mathcal{C}$, $\hat{}U$, $\hat{}L^{\text{net}}$, κ, η
结果： 选定配置 $\boldsymbol{c}^\star$

1 $\phi \leftarrow$ 根据 m_j 将 $j \in \mathcal{R}$ 分类为SBR/CMR/DOR，并且 ρ_j;
2 $\mathcal{Z} \leftarrow \emptyset$;
3 **对于每个** $\boldsymbol{c} \in \mathcal{C}$ **执行**
4 从 $\boldsymbol{c}$ 提取参数 τ_d， λ， β;
5 $\mathbf{B} \leftarrow f_\tau(\tau_d, \phi)$;
6 $N_d \leftarrow \sum_{j \in \mathcal{R}} B_j$;
7 $m^d \leftarrow \sum_{j \in \mathcal{R}} B_j\, m_j$;
8 $\hat{}S(\boldsymbol{c}) \leftarrow$ 多层感知机$^{\text{size}}(\tau_d,\ N_d,\ m^d,\ m^f,\ \lambda)$;
9 $\hat{T}(\boldsymbol{c}) \leftarrow$ 计算时延$(\boldsymbol{c},\ \hat{S}, \hat{U}, \hat{L}^{\text{net}})$ ； // 方程 (2)
10 $\hat{}A(\boldsymbol{c}) \leftarrow$ 多层感知机$^{\text{acc}}(\boldsymbol{c}, N_d, m^d, m^f, \mu(\rho_j)\ \sigma(\rho_j))$;
11 $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{(\boldsymbol{c}, \hat{T}(\boldsymbol{c}), \hat{A}(\boldsymbol{c}))\}$;
12 $\mathcal{C}^{\text{opt}} \leftarrow$帕累托前沿$(\mathcal{Z})$;
13 **如果** $|\mathcal{C}^{opt}| = 1$ **那么**
14 $\boldsymbol{c}^\star \leftarrow \mathcal{C}^{\text{opt}}$的唯一元素;
15 **否则**
16 **如果** $\kappa < \delta^\kappa$ 或 $\eta > \delta^\eta$ **则**
17 $\boldsymbol{c}^\star \leftarrow \arg\min_{(\boldsymbol{c},\ \hat{}T,\ \hat{}A) \in \mathcal{C}\text{优化}} \hat{}T$;
18 **else**
19 $\boldsymbol{c}^\star \leftarrow$膝点$(\mathcal{C}^{\text{opt}})$;
20 **返回** $\boldsymbol{c}^\star$;

V. 系统实现

我们使用商用硬件实现了ViTMAlis的原型。移动客户端为NVIDIA Jetson Orin Nano [38]，一种具有代表性的嵌入式人工智能计算平台，而边缘服务器是配备Intel i 9-13900K CPU和NVIDIA GeForce RTX 5090 GPU的台式工作站。移动设备与服务器均运行Ubuntu 22.04，并通过 TP-Link AX3000 路由器连接。在不进行网络整形的情况下，该配置可实现接近1 Gbps的吞吐量。为了在真实网络状况下评估性能，我们使用网络仿真工具Mahimahi [39]。具体而言，Mahimahi回放真实世界的4G和5G轨迹，以仿真移动设备与服务器之间随时间变化的上行吞吐量和网络延迟。在数据传输方面，我们采用ZeroMQ [40]，因其具备轻量级消息传递能力且开销较低。尽管如此，ViTMAlis的通信层与协议无关，并支持与其他传输方式无缝集成。

原型使用 Python 实现，多层感知机模型则使用 PyTorch 进行训练和部署。我们使用 OpenCV [41] 将卸载帧编码为 JPEG，并在收到后进行解码。对于本地跟踪器而言，轻量级实现至关重要，因为它必须同时跟踪多个对象，并对卸载过程中生成的中间帧执行额外的追赶跟踪 [25] （即将延迟到达的远程结果与当前实时视频对齐）。评估了多种方法 [1], [25], [42], 后，我们选择了基于光流的跟踪方法 [25]，因为它在跟踪准确率和速度之间实现了最佳平衡。

表 I本文使用的视频来源

Name	来源	光照	运动	录制场景
walkS	YouTube [43]	晴天	Low	晴朗的乡村 walk
walkR	YouTube [44]	雨天	Low	城市漫步于细雨
walkB	自拍	人工照明	Low	在.....内部行走熙熙攘攘的商业建筑
骑行	YouTube [45]	晴天	High	晴天骑行从 Brentwood 到好莱坞
driveN	YouTube [46]	夜晚	High	夜间驾车墨西哥城周边

VI. 评估

A. 评估设置

视频数据集。 表 I 总结了用于评估的视频。为确保评估全面，数据集涵盖了多种运动模式、光照条件（白天/夜晚、室内/室外）和天气场景。视频来源包括 YouTube [43]–[46] ，以及我们使用佩戴式 Insta360 GO 3S 相机录制的视频 [47]。重要的是，所有视频均由移动摄像机拍摄，以贴近反映 MVA 的内容特征。我们从每个视频来源中提取了一段 60 秒的视频片段，用于模型训练和性能分析；另提取了一段 300 秒的视频片段，用于系统评估。我们的评估数据集共包含 45,000 帧。

实验测试平台。为确保评估过程具有实验可复现性，并在相同视频内容下进行公平比较，我们在移动设备上回放预录制的视频片段，以模拟实时摄像头输入。具体而言，为模拟以 1080p 和 30 帧/秒运行的真实摄像头，我们利用 NVIDIA Jetson Orin Nano 上的硬件解码加速器，并构建 GStreamer[48]流水线以实现高效的视频解码。由于硬件解码速度超过 30 FPS，我们采用节奏控制队列来调节帧向摄像头缓冲区的传输，从而确保严格遵循 30 FPS 的实时输入速率。

服务器端推理模型。评估时我们使用 ViTDet-L，该模型使用图像块大小为 16×16 像素的数据进行预训练。为便于进行帧区域划分，我们使用较小的窗口大小 9×9对预训练模型进行微调。默认下采样比率 d设为 2，这意味着每个决策区域由 18×18个图像块组成（回顾 §III-A）。

网络轨迹。 该轨迹数据集由多个公开来源 [49]–[51] 构建而成，以捕获多样化的移动模式。它共包含60条轨迹，在4G和5G网络之间平均分配。每条轨迹持续300秒，以与测试视频时长对齐，并以1秒的细粒度时间分辨率变化。各轨迹的平均上传吞吐量在4G下为10.4至36.4 Mbps，在5G下为12.2至135.5 Mbps。平均网络延迟在4G下为39毫秒，在5G下为34毫秒。

配置与超参数设置。对于压缩质量参数 λ，我们考虑 7 种选项，范围从 70 到 100（最高质量），增量为 5。β 有 5 个候选选项，如 §III-B 所述。请注意，当 $\tau_d = 0$ 时，β 固定为 0，因为没有下采样区域就无需进行分辨率恢复。根据性能分析统计数据，运动阈值 δ^m 设为 0.001，以便有效区分 SBR 与其他区域类型。相关性阈值 δ^ρ 设为 0，以将所有目标相关区域视为关键区域，从而优先保证准确率。阈值δ^η 和 δ^κ 根据经验分别设为 30 和 0.7。最后，我们使用 Optuna 超参数优化框架 [52] 来优化多层感知机架构。因此，多层感知机^{size} 和多层感知机^{acc} 均配置为包含 128、64 和 1 个神经元的三层结构。

性能指标。为评估系统整体性能，我们重点关注以下关键指标：

- 渲染准确率。它反映用户感知的体验质量（QoE）。其定义为最终渲染结果（向用户显示）与相应真实检测结果之间的 F1 分数，并在视频的所有帧上进行汇总。真实检测结果通过在原始全分辨率视频帧上运行原始 ViTDet-L 模型获得。
- 端到端卸载延迟：这是从开始帧编码到接收到推理结果的总时长。该指标衡量返回的推理结果的新鲜度，并直接决定本地状态更新的及时性。
- 推理准确率。该指标定义为服务器端模型的推理结果与相应真实结果之间的 F1 分数，并在视频中所有卸载帧上进行汇总。
- 卸载间隔。该指标衡量连续两个卸载帧之间的帧步长。

基线方法。我们考虑以下基线方法：

- *Back2Back*：一种来自先前工作的标准基线 [1], [53]，在收到上一次卸载的推理结果后，立即卸载最近捕获的帧。如果某帧的远程推理结果未能在其渲染截止时间前返回，则重复使用最新的远程结果。
- *TrackB2B*： 一种以准确率为中心的基线方法。它通过在 Back2Back 的基础上加入本地跟踪器，体现了典型的边缘辅助MVA范式 [3], [25] 。当远程推理结果因卸载时延过长而无法及时获得时，跟踪器会生成预测结果，以维持渲染准确率。此外，它会卸载全分辨率帧并对其进行推理，以保持推理准确率。
- *TrackRoI*：一种内容感知基线。它代表了适应视频内容的标准基于 RoI 的压缩策略 [16], [54]。该方法基于 TrackB2B 构建，通过屏蔽非 RoI 区域来减少数据传输；在我们的设定中，这些区域对应于未被分类为 DOR 的决策区域

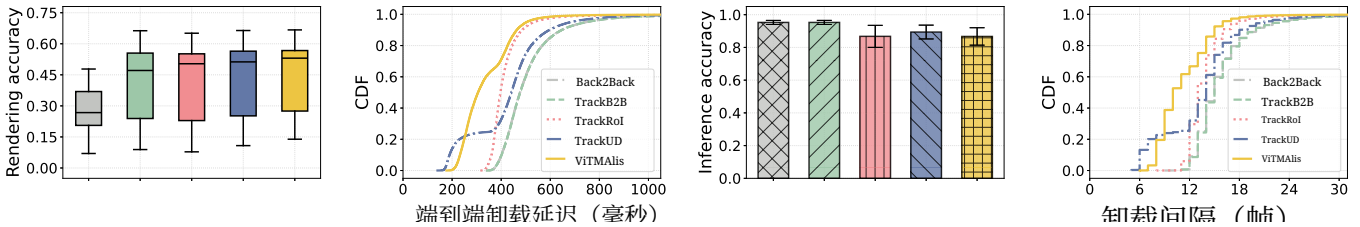

图8. 不同解决方案的整体性能。图中展示了全部 300 组视频轨迹对的统计数据。对于推理准确率，柱形高度表示所有配对样本的均值，误差线表示标准差。

中 .

- *TrackUD*：一种延迟自适应基线。受自适应视频流启发 [55], [56], ，它基于 TrackB2B 构建，并动态调整帧分辨率，以降低较高的系统延迟。具体而言，它会监测最近的端到端卸载延迟。如果延迟超过经经验确定的 15 个帧间隔阈值，则将帧均匀下采样 2 倍，以重新与实时输入同步。否则，采用全分辨率，以优先保证推理准确率。

所有基线方法均采用默认 JPEG 压缩质量 95。该设置符合标准实践（例如 OpenCV 中的默认值），并且经实验证明，在强基线方法中能取得最佳性能。

B. 整体系统性能

图8比较了所有评估方案的整体性能。如图所示，Back2Back 的渲染准确率最低，中位数为 0.268。尽管其推理准确率均值较高，约为 0.953，但较长的端到端卸载延迟会导致返回的推理结果过时，最终降低渲染准确率。具体而言，约 40% 的卸载间隔超过 15 帧，这表明渲染视图至少落后于真值 15 帧。以准确率为中心的基线方案 TrackB2B 与 Back2Back 具有相同的卸载行为，但由于采用了本地跟踪器，其中位渲染准确率大幅提升至 0.471。

与 TrackB2B 相比，TrackRoI 通过内容感知掩码机制进一步将中位渲染准确率提升至 0.504。屏蔽部分帧区域不可避免地会降低卸载帧的推理准确率，使推理准确率均值从 0.953 降至 0.868。不过，这也降低了端到端卸载延迟，提升了返回的远程推理结果的新鲜度，最终有助于提高渲染准确率。如图9所示，TrackRoI 的端到端卸载延迟之所以降低，主要是因为网络通信延迟较低；屏蔽部分帧区域减少了通过网络传输的数据量。在

相比之下，延迟自适应基线 TrackUD 与 TrackB2B 相比，同时降低了网络延迟和推理延迟。网络延迟和推理延迟的共同降低，有效减少了网络状况较差时的端到端卸载延迟，从而确保远程推理结果的新鲜度。尽管在均匀下采样帧上进行推理导致均值推理准确率下降，

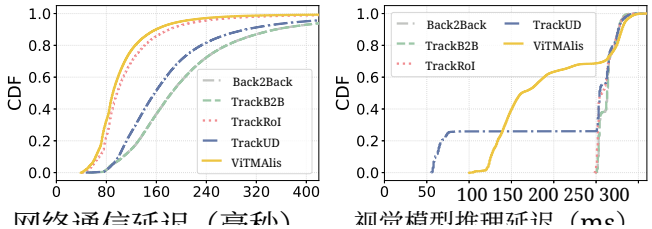

图 9. 延迟深入分析。

最终渲染准确率（0.512）仍高于 TrackRoI。

图 9 显示，我们的动态混合分辨率推理策略将视觉模型推理延迟的中位数从 263 ms 以上降至 162 ms。同时，通过在移动设备上策略性地优化混合分辨率决策，ViTMAlis 将网络通信延迟的中位数降至 90 ms，与 Back2Back 和 TrackB2B 相比，降幅近 51%。这些改进共同显著降低了端到端卸载延迟和卸载间隔，从而实现更快的反馈和更频繁的本地状态更新，以确保新鲜度。得益于我们以准确率为导向的区域选择策略，推理准确率均值的下降幅度适中（0.867），与 TrackRoI 和 TrackUD 的表现相当。总体而言，ViTMAlis 实现了最高的渲染准确率中位数 0.530，验证了我们的内容与网络感知框架能够有效平衡卸载质量与及时性。

C. 编解码延迟与开销分析

图 10 比较了不同方案的编码延迟和解码延迟。在各基线方法中，与 Back2Back 和 TrackB2B 相比，TrackUD 的编解码延迟第一四分位数显著更低。这是因为部分帧是在下采样分辨率下处理的。相比之下，TrackRoI 和 ViTMAlis 的编码延迟中位数略高，这是因为生成掩码帧或混合分辨率帧需要预处理开销。然而，这些帧的像素数量较少，因此与其他基线方法相比，解码延迟中位数更低；不过，由于掩码依赖于内容，其延迟方差也较高。具体而言，Back2Back、TrackB2B、TrackRoI、TrackUD 和 ViTMAlis 的总编解码延迟中位数分别为 30、30、38、30 和 39 毫秒。如图所示，ViTMAlis 的编解码开销与内容感知基线 TrackRoI 相当，与

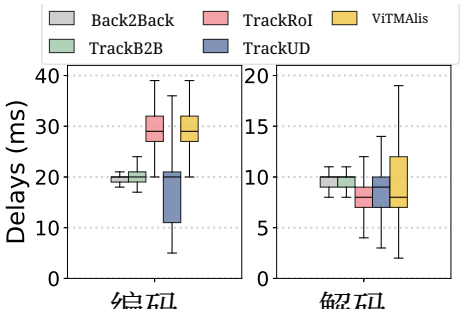

图10. 编解码延迟

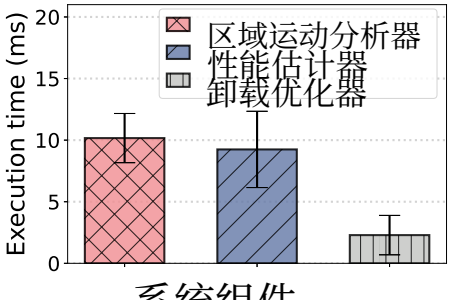

图11. 系统开销

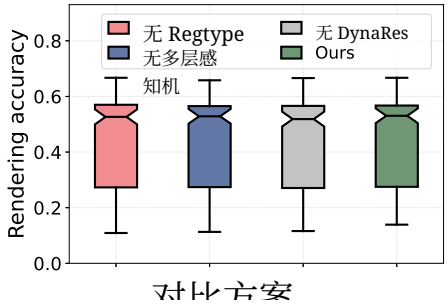

图12. 消融实验

表II 不同估计方法的性能比较。

方法	预测 type	平均绝对误差 ↓	均方根误差 ↓	平均绝对百分比误差 ↓	R^2 ↑
线性回归	大小 (KiB)	98.0	131.6	51.6	0.612
	准确率	0.164	0.206	48.6	0.549
离线 Mean	大小 (KiB)	53.3	83.1	26.8	0.846
	准确率	0.082	0.122	22.3	0.841
多层感知机（我们的方法）	大小（KiB）	38.6	67.4	17.3	0.898
	准确率	0.078	0.114	17.7	0.860

其他基线方法相比，增加的开销可以忽略不计，而渲染准确率却显著提升。

ViTMAlis 的系统开销主要来自三个关键组件：区域运动分析器、性能估计器和卸载优化器。如图11所示，区域运动分析器在 Jetson Orin Nano GPU 上处理每帧的平均耗时为 10 ms。需要注意的是，该组件以独立线程的形式实现，与本地跟踪器并行运行，因此，与启用跟踪的基线方法相比，几乎不会引入额外延迟。性能估计器采用两个轻量级多层感知机模型来预测压缩数据大小和推理准确率，评估所有候选配置平均仅需 9 ms。最后，卸载优化器执行算法1以确定最优配置，该过程效率很高，平均耗时约为 2 ms。

D. 消融实验

为验证我们设计选择的有效性，我们开展了两阶段消融实验。首先，我们评估性能估计器的预测准确率，以证明采用基于多层感知机的建模是合理的。其次，我们评估移除关键系统组件对渲染准确率的影响。**性能估计准确率**。我们将基于多层感知机的估计器与两种替代方法进行比较：（1）线性回归，该方法采用与多层感知机模型完全相同的输入特征集，但尝试使用线性函数预测性能指标；（2）离线均值，这是一种静态基线，使用离线分析期间观察到的平均值来估计压缩数据大小和推理准确率。为确保泛化能力，所有估计器均在汇总了所有评估领域的分析视频片段的复合数据集上进行训练或校准，从而减轻特定场景带来的偏差。

表II 总结了四项指标的结果：平均绝对误差 (MAE)、均方根误差 (RMSE)、平均绝对百分比误差 (MAPE) 和决定系数 (R^2)。如表所示，线性回归的拟合效果较差，数据量和准确率的 R^2 值分别仅为 0.612 和 0.549。这证实了输入特征与性能目标之间的关系具有高度非线性，无法通过简单的线性函数捕获。离线均值基线虽然将 R^2 提高到约 0.84，但误差率仍然很高 (MAPE $>$ 22%)，因为它忽略了视频内容的动态变化。

相比之下，我们基于多层感知机的估计器在所有指标上都取得了最佳性能。对于数据量预测，它将 R^2 提高到 0.898，并将 MAPE 降至 17.3%。同样，在准确率预测方面，它实现了最低的 MAE (0.078) 和 RMSE (0.114)。这些结果表明，我们的轻量级多层感知机能够有效捕获输入特征与性能目标之间复杂的非线性依赖关系。此外，我们进行了跨领域验证（在步行场景上训练，在驾驶/骑行场景上测试），以评估鲁棒性。结果显示，性能下降不到 5%。我们认为，这种鲁棒性源于估计器依赖抽象的统计特征，而非特定领域的视觉模式，因此能够有效地泛化到不同的移动场景。

对系统性能的影响。接下来，我们评估各个关键系统组件对整体性能的贡献。具体而言，我们将完整系统与三个消融变体进行比较：(1) 无 RegType，即移除细粒度区域分类，改为对所有非 DOR 进行下采样；(2) 无多层感知机，即以文中所述的静态离线均值估计器替代内容感知性能估计器；(3) 无 DynaRes，即禁用动态分辨率恢复机制，将特征图上采样推迟至 ViT 骨干网络的最后一层。

图12展示了我们提出的方案及其消融变体的渲染准确率分布。如图所示，消融实验使渲染准确率中位数从 0.530（完整系统）降至 0.526（无 Regtype）、0.528（无多层感知机）和 0.518（无 DynaRes）。我们观察到，尽管所有变体都实现了更低的端到端卸载延迟均值，但这是以推理准确率均值下降为代价的，

这最终会降低渲染质量。

对于无 RegType 和无 DynaRes 的情况，性能下降源于决策空间本身受到限制。由于缺少相应组件所提供的精细控制，优化器无法利用那些资源效率高且准确率足够的配置。相比之下，无多层感知机的方案仍可访问完整的配置空间，但受限于其对静态非自适应性能分析的依赖。由于固定的性能估算无法捕捉视频内容复杂度的实时变化，优化器会变得僵化，往往倾向于选择有限的一组次优配置，而非针对特定帧进行调整。尽管性能有所下降，但值得注意的是，所有消融变体仍优于基线方法。这证实，即使没有这些特定优化，我们的核心混合分辨率压缩与推理策略依然稳健，能够有效优化以 ViT 为骨干网络的 MVA 系统的准确率-延迟权衡。

第七节. 相关工作

A. 边缘辅助移动视频分析

近期的 MVA 研究通过流水线优化、分布式计算基础设施和多目标调度，逐步解决边缘卸载的时延与可扩展性限制。Liu等人 [17]的基础性工作通过解耦渲染与卸载流水线，最大限度地降低了单任务（即目标检测）卸载时延。他们提出动态感兴趣区域编码 和并行流传输与推理 来加速卸载，同时采用基于运动矢量的本地跟踪来维持检测准确率。为满足高分辨率视频日益增长的计算需求，Elf [18] 根据区域提议对帧进行分区，并将部分推理任务并行卸载至多台服务器，从而提升可扩展性。为解决此类分布式环境中的卸载路径选择难题，FPSelector [57] 利用强化学习优化多路径传输，专门管理连续子任务之间的数据依赖关系，以防止无效的卸载决策。

从单任务优化拓展至多任务优化后，AccuMO [2] 在多任务增强现实场景中协调资源分配，例如同时进行深度估计和里程计。它根据任务准确率对卸载频率和帧内容的敏感度，动态平衡卸载频率与本地执行，从而最大限度地提高应用整体准确率。最后，为解决服务器端的可扩展性问题，ARISE [3] 协调多个并发客户端的卸载计划，在严格满足各自服务级别协议的同时，最大限度地提升服务器批处理效率。尽管取得了这些进展，现有研究主要依赖以 CNN 为骨干网络的视觉模型或传统计算机视觉算法。因此，这些研究缺乏针对 ViT 的专门优化，既未能应对其独特的计算挑战，也未能利用其特定架构带来的机会。

B. 视觉*Transformer*推理加速

为缓解 ViT 的二次计算开销，已有大量研究致力于以模型为中心的加速技术，例如词元剪枝、令牌合并和混合分辨率推理。例如，DynamicViT [28]引入了一个可学习的预测模块，可在中间层按层级选择性地丢弃信息量较低的词元，从而减少冗余。ToMe [29] 使用二分图匹配算法逐步合并相似词元，在无需重新训练的情况下减少词元数量。Quadformer [23]利用四叉树算法构建混合分辨率令牌序列，根据显著性评分器估算的重要性，将图像递归地拆分为不同图像块尺寸。尽管这些技术成功降低了推理复杂度，但通常假设数据可在本地获取且计算资源充足，只关注降低理论浮点运算次数。因此，它们往往忽视移动设备的资源限制，以及卸载场景中固有的网络传输成本。

与算法优化相辅相成的是，系统层面的研究着重于协同推理和自适应卸载，以加速实际部署中的视觉变换器。在设备到云端的场景中，Janus [58]提出了一种框架，通过动态结合令牌剪枝与自适应模型拆分，在网络状况波动时保持性能。同样面向云端卸载，LVMSciSSOR [59]采用水平模型并行和启发式调度，最大限度地降低端到端延迟，从而解决大型视觉模型中间数据膨胀的问题。转向设备间协作，SpViT [60] 提出了一种自适应拆分框架，将 ViT 注意力头和神经元分配到多个本地移动设备上，以实现并行推理。尽管这些方法有效，但通常依赖复杂的分区或结构化剪枝，会带来显著的同步开销，或需要对架构进行大幅变更。不同于这些方向，我们的工作提出了一种非侵入式混合分辨率策略，在输入层面优化令牌表示，在不拆分模型的情况下，同时降低传输和计算成本。

VIII. 结论

本文介绍了 ViTMAlis，这是一种原生支持 ViT 的卸载框架，旨在应对基于 ViT 的模型在边缘辅助 MVA 中面临的巨大资源和时延挑战。利用 ViT 与输入无关的特性，ViTMAlis 提出了一种混合分辨率推理策略，可根据实时视频内容和网络状况，动态平衡端到端卸载延迟与推理准确率。该框架结合轻量级设备端运动分析和局部跟踪，实现以准确率为导向的区域下采样和配置优化。在真实移动视频和网络轨迹上的大量实验表明，与强基线方法相比，ViTMAlis 显著降低了端到端卸载延迟，同时实现了更高的渲染准确率。

参考文献

[1] J. Meng、Z. J. Kong、Y. C. Hu、M. G. Choi 和 D. Lal，“边缘辅助增强现实是否需要复杂的系统设计？”，载于第五届边缘系统、分析与网络国际研讨会（*EdgeSys*）论文集，2022年，第 7–12 页。

[2] Z. J. Kong、Q. Xu、J. Meng 和 Y. C. Hu，“AccuMO：面向准确率的边缘辅助移动增强现实多任务卸载”，载于第29届移动计算与网络国际年会（MobiCom）论文集，2023年，第 1–16 页。[3] Z. J. Kong、Q. Xu 和 Y. . Hu，“ARISE：通过主动调度实现高容量 AR 卸载推理服务”，载于第22届移动系统、应用与服务国际年会（MobiSys）论文集，2024年，第 451–464 页。[4] A. Dosovitskiy、L. Beyer、A. Kolesnikov、D. Weissenborn、X. Zhai、T. Unterthiner、M. Dehghani、M. Minderer、G. Heigold、S. Gelly 等人，“一张图像胜过16×16 个词：面向大规模图像识别的 Transformer”，arXiv 预印本 arXiv:2010.11929，2020年。[5] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin，“Attention is all you need”，《神经信息处理系统进展》第31届会议（NeurIPS）论文集，第 30 卷，2017年。[6] R. Strudel、R. Garcia、I. Laptev 和 C. Schmid，“Segmenter：用于语义分割的 Transformer”，载于 IEEE/CVF国际计算机视觉会议（ICCV）论文集，2021年，第 7262–7272 页。[7] Y. Li、H. Mao、R. Girshick 和 K. He，“探索用于目标检测的简洁视觉Transformer骨干网络”，载于欧洲计算机视觉会议（ECCV），2022年，第 280–296 页。[8] Y. Xu、J. Zhang、Q. Zhang 和 D. Tao，“VitPose：用于人体姿态估计的简易视觉 Transformer基线方法”，《神经信息处理系统进展》（NeurIPS），第 35 卷，38 584，2022年，第 38 571 页。[9] A. Kirillov、E. Mintun、N. Ravi、H. Mao、C. Rolland、L. Gustafson、T. Xiao、S. Whitehead、A. C. Berg、W.-Y. Lo 等人，“Segment Anything”，载于 IEEE/CVF国际计算机视觉会议（ICCV）论文集，2023年，第 4015–4026 页。[10] N. Ravi、V. Gabeur、Y.-T. Hu、R. Hu、C. Ryali、T. Ma、H. Khedr、R. Rádle、C. Rolland、L. Gustafson 等人，“SAM 2：对图像和视频进行分割”，arXiv预印本 arXiv:2408.00714，2024年。[11] Y. Zhang、X. Huang、J. Ma、Z. Li、Z. Luo、Y. Xie、Y. Qin、T. Luo、Y. Li、S. Liu 等人，“Recognize Anything：一种强大的图像标注模型”，载于 IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，2024年，第 1724–1732 页。[12] A. Radford、J. W. Kim、C. Hallacy、A. Ramesh、G. Goh、S. Agarwal、G. Sastry、A. Askell、P. Mishkin、J. Clark 等人，“通过自然语言监督学习可迁移的视觉模型”，载于第38届国际机器学习会议（ICML）论文集。PMLR，2021年，第 8748–8763 页。[13] M. Maaz、H. Rasheed、S. Khan 和 F. S. Khan，“Video-ChatGPT：借助大型视觉语言模型实现细粒度视频理解”，载于计算语言学协会第62届年会（ACL）论文集，2024年。[14] H. Zhang、G. Ananthanarayanan、P. Bodik、M. Philipose、P. Bahl 和 M. J. Freedman，“利用近似和延迟容忍实现大规模实时视频分析”，载于第14届USENIX网络系统设计与实现研讨会（NSDI）论文集，2017年，第 377–392 页。[15] Y. Li、A. Padmanabhan、P. Zhao、Y. Wang、G. H. Xu 和 R. Netravali，“Reducto：摄像头端过滤，实现资源高效的实时视频分析”，载于ACM数据通信特别兴趣小组通信网络应用、技术、架构与协议年会（SIGCOMM）论文集，2020年，第 359–376 页。[16] S. Liu、T. Wang、J. Li、D. Sun、M. Srivastava 和 T. Abdelzaher，“AdaMask：通过自适应帧掩蔽实现以机器为中心的视频流，支持DNN推理卸载”，载于第30届ACM国际多媒体会议（MM）论文集，2022年，第 3035–3044 页。[17] L. Liu、H. Li 和 M. Gruteser，“用于移动增强现实的边缘辅助实时目标检测”，载于第25届移动计算与网络国际年会（MobiCom），2019年，第 1–16 页。[18] W. Zhang、Z. He、L. Liu、Z. Jia、Y. Liu、M. Gruteser、D. Raychaudhuri 和 Y. Zhang，“Elf：通过内容感知并行卸载加速高分辨率移动深度视觉”，载于第27届移动计算与网络国际年会（MobiCom）论文集，2021年，第 201–214 页。[19] K. Teja C.-V.、S. Mittal、M. Emani、V. Vishwanath 和 A. K. Somani，“优化Transformer推理技术综述”，《系统架构杂志》，第 144 卷，文章编号 102990，2023年。[20] Z. Kong、P. Dong、X. Ma、X. Meng、W. Niu、M. Sun、X. Shen、G. Yuan、B. Ren、H. Tang 等人，“SpViT：实现更快速的视觉Transformer，借助延迟感知软令牌剪枝”，载于欧洲计算机视觉会议（ECCV）。施普林格，2022年，第620–640页。[21] M. Fayyaz、S. A. Koohpayegani、F. R. Jafari、S. Sengupta、H. R. V. Joze、E. Sommerlade、H. Pirsiavash 和 J. Gall，“用于高效视觉变换器的自适应令牌采样”，载于欧洲计算机视觉会议（ECCV）。施普林格，2022年，第396–414页。[22] J. B. Haurum、S. Escalera、G. W. Taylor 和 T. B. Moeslund，“该使用哪些令牌？探究视觉变换器中的令牌缩减”，载于IEEE/CVF国际计算机视觉会议（ICCV）论文集，2023年，第773–783页。[23] T. Ronen、O. Levy 和 A. Golbert，“采用混合分辨率令牌化的视觉 Transformer”，载于IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，2023年，第4612–4621页。[24] J. D. Havtorn、A. Royer、T. Blankevoort 和 B. E. Bejnordi，“Msvit：用于视觉Transformer的动态混合尺度令牌化”，载于IEEE/CVF国际计算机视觉会议（ICCV）论文集，2023年，第838–848页。[25] T. Y.-H. Chen、L. Ravindranath、S. Deng、P. Bahl 和 H. Balakrishnan，“Glimpse：移动设备上的连续实时目标识别”，载于第13届ACM嵌入式网络传感器系统会议（SenSys）论文集，2015年，第155–168页。[26] M. Caron、H. Touvron、I. Misra、H. Jégou、J. Mairal、P. Bojanowski 和 A. Joulin，“自监督视觉变换器中的新兴特性”，载于IEEE/CVF国际计算机视觉会议（ICCV）论文集，2021年，第9650–9660页。[27] K. He、X. Chen、S. Xie、Y. Li、P. Dollár 和 R. Girshick，“掩码自编码器是可扩展的视觉学习器”，载于IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，2022年，第16 000–16 009页。[28] Y. Rao、W. Zhao、B. Liu、J. Lu、J. Zhou 和 C.-J. Hsieh，“Dynamicvit：采用动态令牌稀疏化的高效视觉Transformer”，载于第35届神经信息处理系统大会（NeurIPS）论文集，2021年，第13 937–13 949页。[29] D. Bolya、C.-Y. Fu、X. Dai、P. Zhang、C. Feichtenhofer 和 J. Hoffman，“令牌合并：让你的ViT更快”，载于第十一届国际学习表征会议（ICLR），2023年。[30] S. Long、Z. Zhao、J. Pi、S. Wang 和 J. Wang，“超越注意力令牌：纳入令牌重要性和多样性以提升视觉Transformer效率”，载于IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，2023年，第10 334–10 343页。[31] C. Ryali、Y.-T. Hu、D. Bolya、C. Wei、H. Fan、P.-Y. Huang、V. Aggarwal、A. Chowdhury、O. Poursaeed、J. Hoffman等，“Hiera：一种摒弃繁复设计的层次化视觉Transformer”，载于国际机器学习会议（ICML）。PMLR，2023年，第29 441–29 454页。[32]M. Research，“Detectron2 - projects - vitdet”，https://github.com/facebookresearch/detectron2/tree/main/projects/ViTDet，2026，[在线；访问日期：15-Jan-2026]。[33] OpenCV，“如何使用背景减除方法”，https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html，2026，[在线；访问日期：15-Jan-2026]。[34] J. Lee、S. Lee、J. Lee、S. Sathyanarayana、H. Lim、J. Lee、X. Zhu、S. Ramakrishnan、D. Grunwald、K. Lee等，“Perceive：基于实时调度模式的深度学习蜂窝上行链路预测”，载于第18届国际移动系统、应用与服务会议（MobiSys）论文集，2020年，第377–390页。[35] A. Narayanan、E. Ramadan、R. Mehta、X. Hu、Q. Liu、R. A. Fezeu、U. K. Dayalan、S. Verma、P. Ji、T. Li等，“Lumos5g：商用毫米波5G吞吐量的测绘与预测”，载于ACM互联网测量会议（IMC）论文集，2020年，第176–193页。[36] G. Lv、Q. Wu、W. Wang、Z. Li 和 G. Xie，“Lumos：通过准确的吞吐量预测提升视频流媒体体验质量”，载于IEEE计算机通信会议（INFOCOM）。IEEE，2022年，第650–659页。[37] K. Li、H. Nie、H. Gao 和 X. Yao，“基于权衡效用的拐点识别”，arXiv预印本 arXiv:2005.11600，2020年。[38] 英伟达，“Jetson Orin Nano开发者套件入门指南”，https://developer.nvidia.com/embedded/learn/get-started-jetson-orin-nano-devkit，2026，[在线；访问日期：15-Jan-2026]。[39] R. Netravali、A. Sivaraman、S. Das、A. Goyal、K. Winstein、J. Mickens 和 H. Balakrishnan，“Mahimahi：用于{记录与回放} {HTTP}的精准工具”，载于2015年USENIX年度技术会议（ATC），2015年，第417–429页。[40] ZeroMQ，“开源通用消息传递库”，https://zeromq.org/，2026，[在线；访问日期：15-Jan-2026]。

[41] OpenCV，“OpenCV：开源计算机视觉库”，https://opencv.org/，2026, [在线；访问日期：15-Jan-2026]。[42] ——，“Vit跟踪器”。https://github.com/opencv/opencv_zoo/tree/main/models/object_tracking_vittrack，2026, [在线；访问日期：15-Jan-2026]。[43] YouTube，“沿着英格兰乡村Shaftesbury的Gold Hill漫步，4K”，https://www.youtube.com/watch?v=0gnm6gGMC3c，2026, [在线；访问日期：15-Jan-2026]。[44] ——，“在墨西哥城雨中漫步，双耳音频”，https://www.youtube.com/watch?v=h-_FmmmBKZI，2026, [在线；访问日期：15-Jan-2026]。[45] ——，“从Brentwood骑行至好莱坞 ｜ 4k”，https://www.youtube.com/watch?v=mjaHoNEuM-o，2026, [在线；访问日期：15-Jan-2026]。[46] ——，“墨西哥城4K——夜间驾车”，https://www.youtube.com/watch?v=oxZBX-e84O8, 2026, [在线；访问日期：15-Jan-2026]。[47] Insta360，“Insta360 GO 3S——小巧强大的4K相机”，https://www.insta360.com/product/insta360-go3s，2026, [在线；访问日期：15-Jan-2026]。[48] GStreamer，“GStreamer：开源多媒体框架”，https://gstreamer.freedesktop.org/，2026, [在线；访问日期：15-Jan-2026]。[49] M. Ghoshal、Z. J. Kong、Q. Xu、Z. Lu、S. Aggarwal、I. Khan、Y. Li、Y. C. Hu 和 D. Koutsonikolas，“5G毫米波网络上行链路性能的深入研究”，载于ACM SIGCOMM研讨会《5G及未来网络测量、建模与用例》论文集，2022年，第29–35页。[50] M. Ghoshal、I. Khan、Z. J. Kong、P. Dinh、J. Meng、Y. C. Hu 和 D. Koutsonikolas，“车载蜂窝网络性能”，载于第23届ACM互联网测量会议（IMC）论文集，2023年。[51] D. Raca、J. J. Quinlan、A. H. Zahran 和 C. J. Sreenan，“超越吞吐量：包含信道和上下文指标的4G LTE数据集”，载于第9届ACM多媒体系统会议（MMSys）论文集，2018年，第460–465页。[52] Optuna，“用于自动化超参数搜索的开源超参数优化框架”，https://optuna.org/，2026, [在线；访问日期：15-Jan-2026]。[53] P. Dash、Z. J. Kong、Y. C. Hu、C. Turner、D. Wolfensparger、M. G. Choi、A. Kshitij 和 V. E. McLandrich，“如何在边缘辅助的增强现实中对帧传输和服务器推理进行流水线化，以优化增强现实任务准确率？”，载于第6届边缘系统、分析与网络研讨会（EdgeSys）论文集，2023年，第36–41页。[54] K. Du、A. Pervaiz、X. Yuan、A. Chowdhery、Q. Zhang、H. Hoffmann 和 J. Jiang，“用于深度学习推理的服务器驱动视频流传输”，载于ACM数据通信特别兴趣小组计算机通信应用、技术、架构与协议年会（SIGCOMM）论文集，2020年，第557–570页。[55] H. Mao、R. Netravali 和 M. Alizadeh，“使用Pensieve的神经自适应视频流传输”，载于2017年ACM数据通信特别兴趣小组会议（SIGCOMM）论文集，2017年，第197–210页。[56] H. Fang、H. Zhao、J. Shi、M. Zhang、G. Wu、Y. C. Chou、F. Wang 和 J. Liu，“低轨卫星星座上的稳健直播流媒体传输：测量、分析与感知切换的自适应”，载于第32届ACM国际多媒体会议（MM）论文集，2024年，第5958–5966页。[57] Y. Zhu、Y. Huang、X. Qiao、X. Liu、X. Su、A. Brunström、Ö. Alay 和 S. Tarkoma，“FPSelector：用于移动增强现实卸载的灵活路径选择器”，《IEEE移动计算汇刊》（TMC），第24卷第9期，第8423–8440页，2025年。[58] L. Jiang、S. D. Fu、Y. Zhu 和 B. Li，“Janus：动态网络环境下的协作式视觉Transformer”，载于IEEE计算机通信会议（INFOCOM）。IEEE，2025年，第1–10页。[59] Y. Liu、R. Lu 和 D. Wang，“Lvmscissor：通过蛞蝓群算法在移动边缘拆分并调度大视觉模型推理”，《IEEE移动计算汇刊》（TMC），第24卷第8期，第7538–7553页，2025年。[60] S. Zhao、T. Liu、H. Jin 和 D. Yao，“Spvit：通过自适应拆分与卸载加速移动设备上的视觉Transformer推理”，《IEEE移动计算汇刊》（TMC），第24卷第10期，第9303–9318页，2025年。