利用低功耗物联网摄像头进行智慧城市交通监控的支持 Wi-Fi 的边缘智能框架

Raphael Walcher
{rawalcher}@edu.aau.at 克拉根福大学 信息技术研究所 奥地利克拉根福

Kurt Horvath
{kurt}.{horvath}@aau.at 克拉根福大学 信息技术研究所 奥地利克拉根福

Dragi Kimovski
{dragi}.{kimovski}@aau.at 克拉根福大学 信息技术研究所 奥地利克拉根福

Stojan Kitanov
{stojan}.{kitanov}@unt.edu.mk 特蕾莎修女大学 信息科学学院 斯科普里，北马其顿

arXiv:2611.32517v1 [cs.DC] 24 Nov 2026

摘要

智慧城市的实时交通监控需要超低延迟处理，以支持事件检测和拥堵管理等时间关键型决策。尽管基于云的解决方案具有强大的计算能力，但其固有延迟限制了它们在此类任务中的适用性。本文提出了一种本地化边缘人工智能框架，将低功耗物联网摄像头传感器连接到客户端，或将推理任务卸载至 NVIDIA Jetson Nano（图形处理器）。网络连接通过 Wi-Fi 实现，无需依赖 5G 等广域基础设施或有线网络，即可完成图像分类。我们评估了两种处理策略：在摄像头节点上进行本地推理，以及将任务卸载至 Jetson Nano 并利用 GPU 加速。结果表明，本地处理仅适用于轻量级模型和低帧率，而卸载处理即使面对更复杂的模型，也能实现近实时性能。这些结果证明，基于 Wi-Fi 的经济高效边缘人工智能部署适用于对延迟要求严格的城市监控。

CCS 概念

• **网络** → **网络性能评估**； 网络结构； 基于位置的服务； • **计算方法** →分布式计算方法。

关键词

物联网服务、计算连续体、边缘人工智能、智慧城市

1 引言

智慧城市作为智能系统运行，旨在通过解决出行和交通监控 [1], 以及能源优化和废物管理等关键领域的问题来提升生活质量。在智慧城市的这些应用中，实时交通分析对于实现及时决策 [2] [19]、事件检测、缓解拥堵和自适应信号控制至关重要。此类应用要发挥作用，需要超低延迟和大量计算资源 [10]。尽管传统的基于云的解决方案能够提供充足的计算能力，但其固有的网络延迟和有限的服务质量（QoS）保障，为支持延迟敏感型实时智慧城市应用带来了重大挑战。

边缘计算和人工智能（AI）的新兴范式通过将计算任务移至更接近数据源的位置来应对这一挑战 [13], , 从而降低延迟和带宽消耗。然而，边缘系统的部署通常假设具备 5G 或光纤网络等低延迟通信基础设施。这些假设并不普遍成立，尤其是在覆盖范围有限或人口密集的区域[10]。在人员密集的区域，Wi-Fi 可以提供一种替代方案[11]。此外，许多城市监控（以及智慧城市）需求本质上是本地化的，面向特定的交叉路口、行人区域或停车设施。在此类情况下，广域连接可能并无必要，甚至会适得其反。

本文提出了一种本地化边缘人工智能框架，使用 Wi-Fi 作为主要通信技术，将**低功耗物联网摄像头传感器**连接到共享边缘设备，以卸载车辆分类的人工智能推理任务。

每个摄像头节点都会捕获高分辨率图像，并使用其中央处理器在本地进行处理，或将图像卸载至 Nvidia Jetson Nano 节点进行 GPU 加速推理。该设计优先在 Wi-Fi网络覆盖范围内进行近距离、高性能处理，从而无需依赖广域基础设施。

本文的主要贡献如下：

- 设计并实现了一种基于 Wi-Fi 的边缘人工智能架构，将摄像头传感器节点连接至边缘设备，以实现实时图像分类。
- 评估了在智慧城市中采用本地人工智能推理与卸载式人工智能推理进行车辆分类的可行性。
- 量化了单节点部署中处理性能、丢帧与推理延迟之间的权衡。

本文的组织结构如下：第 2 节回顾了智慧城市无线技术的相关研究，重点介绍城市部署中 5G 与 Wi-Fi 之间的权衡。第 3 节介绍了我们的系统架构和实验设置。第 4 节定义了评估中使用的性能指标。第 5 节报告
并讨论我们的实验结果。第 6 节讨论

本作品依据知识共享署名 4.0 国际许可协议进行许可。

研究成果，并描述仍需解决的挑战。第 7 节对本文作出总结。

2 相关工作

了解边缘通信和计算基础设施的一般特征，对于支持智慧城市应用至关重要。因此，Chen 等人的研究 [4] 有助于理解移动边缘计算（MEC）在打造更智慧城市方面所发挥的作用。作者强调，MEC 通过将计算部署在靠近数据源的位置，能够支持延迟敏感型服务；他们还指出，可利用 5G网络实现所需的带宽和低延迟。其架构设想在网络边缘部署 MEC节点，以支持实时视频分析、增强现实和车载通信等应用。然而，该架构依赖 5G基础设施，因此在 5G覆盖不广或部署成本过高的地区，其短期适用性受到限制。

同样，Garcia 等人 [6] 分析了智慧城市物联网部署中的无线技术选择，重点探讨技术选择对可靠性和性能的影响。他们指出，由于蓝牙、ZigBee、Wi-Fi 和移动通信被广泛使用，2.4 GHz ISM频段过度拥挤，这给全球微波接入互操作性（WiMax）带来了挑战，并导致干扰、丢包和性能下降。研究结果表明，技术选择必须兼顾应用的特定需求与干扰缓解，因此 Wi-Fi 是一种可行但取决于具体情境的蜂窝网络解决方案替代方案。

Taleb 等人 [21] 的工作全面评述了智慧城市架构，指出无线技术是城市物联网服务的关键推动因素。他们认为，5G 和 Wi-Fi 在城市部署中都至关重要，并指出，尽管 5G 在广域、高移动性场景中表现出色，但在基础设施投资必须保持最低水平的情况下，Wi-Fi 可为超本地服务提供经济高效的覆盖。

综合来看，这些研究强调，尽管 5G 为大规模移动智慧城市服务带来了高性能，Wi-Fi 在本地或预算受限场景中仍是切实可行的替代方案。结合 Garcia [6] 于 2018 年的研究发现，将 Wi-Fi 转移至 5/6 GHz频段或许能够解决其中许多问题。这促使我们探索一种基于 Wi-Fi 的边缘人工智能框架，用于实时交通监控，在满足低时延要求的同时，摆脱对广域基础设施的依赖。

3 框架

本节介绍一个智慧城市用例应用的框架，用于直接在路侧执行车辆分类，提出一种低功耗、低成本的解决方案，并通过 Wi-Fi 将信息以短距离传输至客户端应用。

如图 1 所示，该框架的架构由三个主要组件构成：客户端、本地处理节点和远程推理节点，分别以两种针对不同目标优化的配置部署，即**本地**和**卸载**。第一种部署方式称为**本地部署**，由一个节点使用 Raspberry Pi 4 在路侧直接执行车辆分类，同时采集输入图像。此配置采用仅 CPU 推理，实现完全的本地处理。客户端设备通过 Wi-Fi 连接到本地节点，并定期请求图像分类结果。结果包括 JSON 格式的带注释车辆分类结果和图像叠加层（参见第 3.2 节），并向客户端应用提供最新结果。

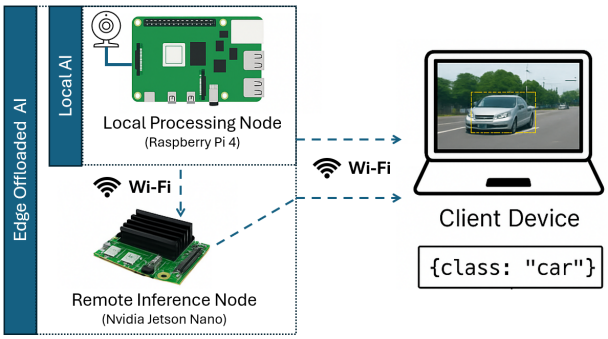

图 1：系统架构概览

在**卸载部署**中，推理在配备 GPU（NVIDIA Jetson Nano）的边缘节点上远程执行，以实现加速。客户端发出请求后，本地节点会将当前图像发送到远程节点，由其执行图像分类。处理完成后，客户端设备从远程节点获取结果。

此设置可直接比较本地（基于 CPU）与卸载式（GPU 加速）的边缘处理，并量化 CPU 和 GPU 推理性能的影响，以及卸载过程引入的额外网络延迟。

3.1 工作流程

本节介绍在所示部署配置中三个主要组件如何交互。根据部署配置，本地处理节点负责图像采集和推理，而远程推理节点仅对本地处理节点提供的图像执行推理。各组件通过以下消息进行通信：

- **Hello(设备 ID)**：向客户端通告设备已上线
- **控制**：包含多种子类型：– 配置实验：声明实验的模型、运行模式和持续时间– 准备开始：确认模型已加载且预热已完成– 开始实验：启动周期性脉冲– 关闭：执行有序的关闭流程

- **脉冲(TimingMetadata)**：由客户端每隔 Δt发出
- **帧**：将图像 F_i 从本地节点传送到远程节点
- **结果**：将推理输出 R_i 返回给客户端

帧 和**结果** 消息都携带由**脉冲**最初发送的 TimingMetadata 数据包。该数据包包含当前推理周期的序列号 i、帧编号以及在各设备上采集的一组时间戳。

各条消息的用途将在以下章节中说明。

3.1.1 工作流程：本地推理。 如图 2 所示，由客户端在与本地处理节点进行的设置阶段启动。在设置阶段，客户端等待本地处理节点发送 Hello 消息，然后发送 Control::ConfigureExperiment，以配置模型、当前部署模式（本地处理或卸载）并定义运行参数（例如，实验时长、固定帧率）。本地处理节点加载模型，执行预热周期，将模型加载到内存中并运行一次虚拟推理过程，以准备运行时环境，然后向客户端回复 Control::ReadyToStart。

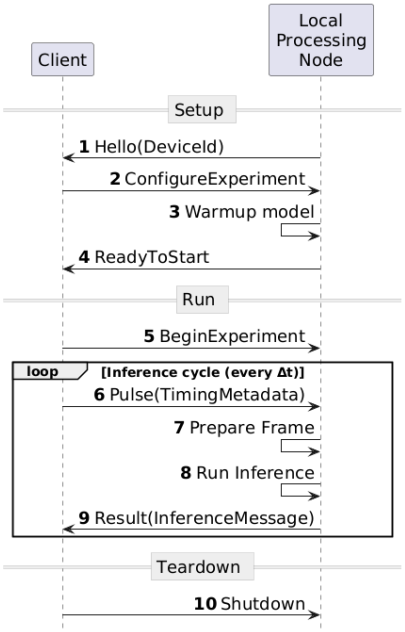

图 2：使用本地推理的本地部署配置工作流程

运行阶段以表示新一轮运行的 Control::BeginExperiment 消息开始。在每个推理周期中，客户端都会发出一条 Pulse 消息，其中携带包含当前序列号 i 的 TimingMetadata 数据包。收到 Pulse 后，本地处理节点从机载摄像头捕获图像 F_i。系统将捕获的帧封装到 Frame(F_i) 消息中。捕获完成后，节点执行推理，并以 Result(R_i) 的形式将结果返回给客户端。客户端通过提交 Shutdown 消息，在拆卸 阶段完成工作流程。

3.1.2 卸载推理工作流程。如图 3 所示，该流程扩展了**本地推理部署**。在设置阶段，客户端还会通过等待 Hello，然后发送 Control::ConfigureExperiment 消息来注册远程推理节点。两个节点都以 Control::ReadyToStart 响应。

在运行阶段，本地处理节点将帧(F_i)传输至远程推理节点。远程推理节点运行模型以获得推理结果 R_i。随后，远程节点将结果(R_i)返回给客户端。这种部署利用了 GPU 加速，但需要承担网络传输开销，因此总处理时间包括推理时间和数据传输延迟。

运行将持续至配置的实验时长结束，随后客户端通过以下方式启动拆解阶段：

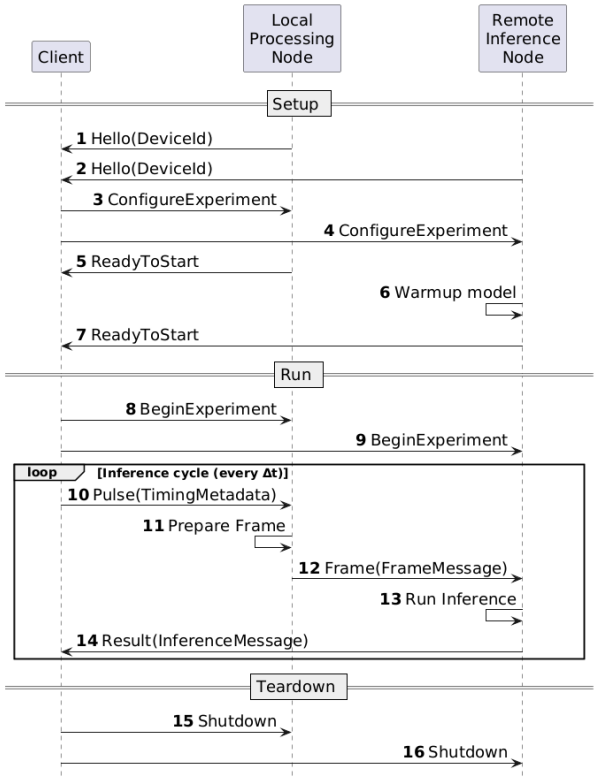

图 3：应用远程推理的卸载部署配置工作流程

向两个节点发送 Control::Shutdown，从而终止实验。

3.1.3 分布式卸载。所提架构不一定受限于本地处理节点 N^{loc} 与远程推理节点 N^{rem}之间的双射关系。可将排队概念 N^{rem} 推广至支持 1： k 节点关系，其中 k是共享一个 N^{rem} 节点进行卸载的 $N^{loc} = \{n_0 \dots n_{k-1}\}$的数量。

采用容量队列策略（$\forall n \in N^{loc}$： $Q_{\text{size}} = 1$）时，每个本地处理节点仅支持处理最新的一帧。

我们必须要求满足以下条件才能分配 $n_i \in N^{loc}$。

$$\frac{\mu_m}{\sum_{i=0}^{k} \lambda_0(n_i)} \geq 1,$$

其中， μ_m 是模型 m 的最高可持续处理速率， λ_0 定义了每节点帧生成速率。

3.2 使用人工智能进行车辆分类

本节介绍该框架的用途。许多交通应用广泛使用这些现有的人工智能模型进行实时交通分类 [20] [23] [16]。我们还使用 *YOLOv5* [12] 作为目标检测器。这些模型基于 COCO 数据集 [14, 24] 以及汽车、卡车、公共汽车和摩托车类别进行训练。

根据模型大小可分为三种：YOLOv5n（纳米型）、YOLOv5s（小型）和 YOLOv5m（中型）。YOLOv5n 变体提供最低的延迟，但精确率较低；而 YOLOv5m 在更高计算成本下实现了更高的准确率。

该选择支持对本地处理节点（仅 CPU）和远程推理节点（基于图形处理器加速）进行系统评估，展示了处理本地性之间的权衡。在模型大小方面，它们的参数数量及相应的内存权重存在显著差异：纳米型模型约为*7 MB*，中型模型为*29 MB*，而采用 FP32 精度时为*85 MB*。YOLOv5 官方模型表 [12] 列出了我们用于推导这些数值的参数数量，其计算依据是 FP 存储中每个参数占用 4 × 32 字节。

图 4 比较了 YOLOv5n 和 YOLOv5m 在相同帧上的输出。两个模型都检测到了多个车辆，但也误分类了一些对象。例如，它们将路侧电气箱识别为汽车；对于 YOLOv5m，它们还将交通信号灯误分类为行人，如图 4a、4c 和 4e 中的红色标记所示。YOLOv5n 还未能检测到卡车和挖掘机等较大物体（以蓝色标记；见图 4e），而中型版本成功识别了这些物体，并且总体上识别出更多车辆。这些示例体现了准确率与模型大小之间的权衡 [17]。较小的模型在计算方面效率较高，但容易出现漏检（图 4a）；较大的模型则能实现更广泛的覆盖范围，但代价是更高的计算成本（图 4f）。



(a) 帧 30175；YOLOV5n **(b) 帧 30175；YOLOV5m**

(c) 帧 24418；YOLOV5n **(d) 帧 24418；YOLOV5m**

(e) 帧 10614；YOLOV5n **(f) 帧 10614；YOLOV5m**

图4：YOLOv5n和YOLOv5m在多视角交通路口数据集帧上的比较 [18]

3.3 实现

该解决方案实现了一种分布式边缘计算架构，包含四个主要软件组件，使用 Rust 和 Python 实现。代码在 Python 中执行实际推理。完整实现可在 GitHub 上的 rust-traffic-watch 仓库中获取[1]

该实现支持跨多个人工智能模型变体（YOLOv5n/s/m）的自动化基准测试、可配置帧率（1-15帧每秒），并支持通过以下软件组件建立的本地部署和卸载部署配置：

- **控制器**：编排实验、协调设备通信，并收集所有分布式节点的性能指标
- **Pi发送端**：捕获帧并执行本地PyTorch推理，或转发数据以卸载至远程计算节点
- **Jetson接收端**：为卸载的工作负载提供加速的TensorRT推理，并优化图形处理器利用率
- **共享库**：用于设备间通信的通用网络协议、二进制序列化和类型定义

所有实验都会生成详细的CSV日志，其中包含端到端延迟、处理开销、网络延迟和检测准确率指标，以便对24种配置进行可复现的分析。

3.3.1 分层时间测量。由于时钟往往无法完全同步，因此在分布式计算节点之间进行准确的时间测量始终存在困难。本段介绍了我们用于避免这一问题的方法。

我们在节点 N $= \{N_c, N_p, N_r\}$之间进行分层时间测量，其中 N_c 是客户端，N_p 是本地处理节点，N_r 是由 N_p触发的远程推理节点。每个节点 N_i 都维护一个本地时钟。客户端 N_c 初始化有序的时间戳结构 T（TimingMetadata）：

$$\mathcal{T} = \langle t_1, \ldots, t_n \rangle,$$

执行期间，N_p 会追加本地处理的时间戳，并触发 N_r，后者会追加推理时间戳。我们仅比较从同一本地时钟获取的时间戳，以避免对齐本地时钟（C_c、C_p、C_r）。因此，我们将节点 N_i 上两个时刻 τ_a、τ_b 之间的持续时间定义为

$$\Delta T_i(t_a, t_b) = C_i(\tau_b) - C_i(\tau_a),$$

使用 $t_a = C_i(\tau_a)$ 和 $t_b = C_i(\tau_b)$。有效的持续时间需要满足：

$$\Delta T_i(t_a, t_b) \quad \text{only if} \quad t_a, t_b \leftarrow C_i,$$

即，运行持续时间的计算方式如下：

$$t_{\text{start}} = C_c(\tau_{\text{start}}),$$
$$t_{\text{end}} = C_c(\tau_{\text{end}}),$$
$$\Delta T_{\text{run}} = t_{\text{end}} - t_{\text{start}}.$$

4 评估设计

在评估中，我们系统地将捕获帧率*CFR* 从每秒 1 帧调整至每秒 15 帧（FPS），并测量不同推理模型对总处理时间、准确率和系统响应性的影响。这些参数使我们能够分析计算位置、网络使用情况与分类性能之间的权衡，为智能

[1]https://github.com/rawalcher/rust-traffic-watch

城市边缘人工智能系统的设计提供参考，同时考量车辆分类智慧城市应用的两种不同实现方式。

4.1 评估指标

表 1 总结了评估过程中使用的符号。该表可作为评估中使用的各种符号和术语的参考。

表 1：描述实验时使用的符号

符号	说明	Unit
CFR	捕获帧率	帧/秒
DFR	丢帧率	帧/秒
OTR	按时处理率	帧/秒
RPI	相对性能提升	-
Z_{total}	实验期间捕获的名义帧总数	帧
Z_{dropped}	实验期间丢弃的帧数	帧
ΔT_{total}	实验总时长	s
$\Delta T_{\text{proc}}(i)$	帧 i的处理时间	s
μ_m	模型 m的最大可持续处理速率	帧/秒
λ_0	每节点帧生成速率	帧/秒
Q_{size}	容量 q　队列大小	帧

4.1.1 性能指标。我们评估图像推理所采用的目标指标是捕获帧率（CFR），定义如下：

$$CFR = \frac{Z_{\text{total}}}{\Delta T_{\text{total}}}.$$

为应对处理速度的变化，我们实现了一个容量受限的队列，大小为 $Q_{\text{size}} = 1$，确保只处理最新帧。处理过程中有新帧到达时，它会替换队列中所有等待处理的帧。该策略优先保证时间上的及时性，而非内容的完整性。

我们将丢帧率(DFR)定义为因处理延迟而丢弃的帧数所占的比例：

$$DFR = \frac{Z_{\text{dropped}}}{Z_{\text{total}}} \times 100\%.$$

在以下情况下，帧会被视为丢弃：

(1) 前一帧仍在处理时该帧到达$\Delta T_{proc}(i-1) > 1/CFR$，且

(2) 该帧在开始处理前被更新的帧替换。

按时处理率(OTR)表示成功处理的帧所占比例，为 $OTR = 100\% - DFR$。

4.1.2 处理时间。每帧的平均处理时间 T_{proc}计算如下：

$$\Delta T_{proc} = \frac{1}{n}\sum_{i} \Delta T_{proc}(i).$$

其中，n 表示系统在不发生帧丢失的情况下成功处理的帧数，而 i 表示当前帧的索引，该索引决定了处理该帧所需的处理时间 ΔT_{proc}。处理时间的累积方式取决于部署配置。$\Delta T_{proc} \in \{\Delta T\text{loc}^{proc}, \Delta T\text{off}^{proc}\}$

对于**本地部署**，处理时间 ΔT_{proc}^{loc} 包括从摄像头进行图像采集、模型推理和结果序列化的累积时间。

对于**卸载部署**，处理时间 $\Delta T_{proc}^{\text{off}}$还包括其他组成部分：向远程节点进行网络传输 t_{in}、远程节点处的排队延迟、模型推理ΔT_{inf}以及结果的网络传输 ΔT_{out}。

4.1.3 性能比较。为评估 GPU 卸载的优势，我们定义相对性能提升 (RPI)：

$$RPI = \frac{DFR_{\text{local}} - DFR_{\text{offload}}}{DFR_{\text{local}}} \times 100\%.$$

如果 $RPI > 0$，卸载可以减少帧丢失，$RPI < 0$ 表明网络开销超过了加速收益。$RPI = 0$，两种方法的效果相同。

4.1.4 特定模型的处理容量。每个模型变体 $m \in$ {YOLOv5n, YOLOv5s, YOLOv5m}均针对每种类型的处理节点及其相应的硬件能力，表现出一个最大可持续处理速率 μ_m t。无帧丢失时的处理容量要求：

$$CFR \le \mu_m.$$

由于采用容量队列实现，系统可以在更高的采集速率下运行，但会按以下方式丢弃帧：

$$DFR(m) \approx \max\left(0, (1 - \frac{\mu_m}{CFR})\right) \times 100\%.$$

此近似假设系统处于稳态运行，且处理时间均匀。

5 结果

我们评估了两种部署策略：在路侧节点（Raspberry Pi 4，仅 CPU ）上直接进行本地人工智能推理，以及通过 Wi-Fi 链路将远程推理卸载到 Jetson Nano（支持GPU）。性能采用第4节中介绍的指标进行量化：丢帧率（DFR）、捕获帧率（CFR）和推理延迟（ΔT_{inf}）。

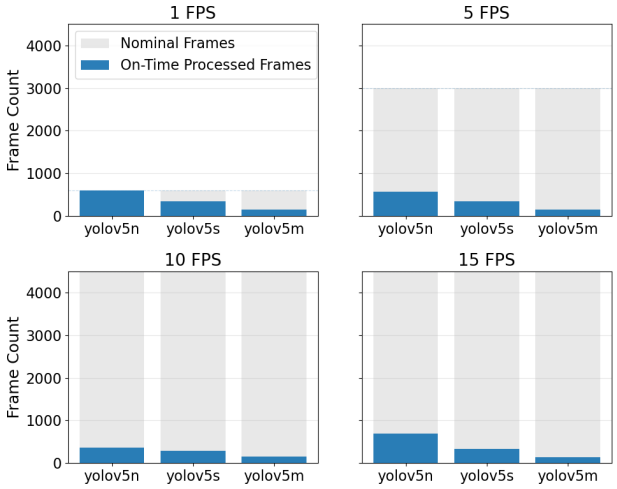

图5：使用本地部署配置时的 OTRoverDFR（y 轴上限为 4000 帧）

审视本地部署配置的结果可知，只有采用最小型模型 YOLOv5n，才能在 $CFR = 1$时持续运行且不发生帧丢失（$DFR = 0$）。这一

设置满足条件 $OTR \geq CPR$，因此只有在前一帧完全处理完毕后，才会接收下一帧。将采集速率提高到 5 帧每秒或 15 帧每秒，会导致成功处理的帧数显著减少；YOLOv5s 的 *DFR* 超过 80%，而 YOLOv5m 的 *DFR* 超过 95%。在这些情况下，较高的 *DFR* 反映了条件 $OTR \ll CFR$，即大多数帧都无法及时处理。



图6：使用卸载部署配置时，按时处理率随 DFR 的变化

另一方面，进行卸载时，情况会发生很大变化。在 $CFR = 1$ 下，我们保持在 $DFR \approx 0$，只有当使用更大的 YOLOv5m 模型达到 $CFR = 5$ 时，才开始丢帧。从 $CFR = 10$开始，所有模型的表现全面变差，其中 *DFR* 从 ≈ 27%（YOLOv5n）到 ≈ 64%（YOLOv5m）不等。在 $CFR = 15$时，我们可以看到与 $CFR = 10$时相同的结果，即远程推理节点面临硬件处理能力限制。

这些发现证实，对于单节点部署，本地推理仅适用于轻量级模型和较低的采集速率。相比之下，将任务卸载到配备 GPU的边缘节点后，即使在较高的采集速率下使用更重的模型，也能实现近实时性能，只要 *OTR*保持足够接近 *CFR*，从而使 *DFR*处于可接受范围 DFR^{max}内。此外，在较低的$CFR \leq 5$ 下，卸载节点的资源能力也足以处理多个节点的数据。

6 讨论

基于实验评估，我们发现，在单个节点上处理视频流可能会受到很大限制。许多现实世界中的边缘人工智能部署都涉及多个处理节点共享一个卸载节点。这种配置带来了若干尚待解决的挑战，例如资源竞争、自适应调度，以及尽管服务能力实际分布在多个节点上，仍要确保可靠的网络性能。

多节点可扩展性使我们能够优化资源使用。增加摄像头节点的数量将提高总数据速率以及人工智能节点 [7]，的计算需求，可能导致其图形处理器过载或使 Wi-Fi网络达到饱和（见第 3.1.3 节）。了解性能下降的极限，并确定最佳的负载均衡和排队策略，将是关键所在。这还需要在干扰情况多变的室外城市环境中开展长期可靠性测试，集成更先进的人工智能模型，以实现实时目标检测与跟踪，并应用压缩技术，在不影响分类准确率的情况下减少带宽需求。

最后，在密集的 Wi-Fi 环境中，需要采用网络优化 技术来缓解干扰和拥塞。这包括信道分配策略、使用 Wi-Fi 7 [5] 功能（如多链路操作（MLO） [3] 和 4K正交幅度调制（4K-QAM） [8], ），以及与网状网络集成，从而在不牺牲性能或超出这些方

y面影响的情况下扩展覆盖范围 p. 这些影响来自 6G。

7 结论

本文介绍了一种基于 Wi-Fi 的边缘人工智能框架，用于智慧城市环境中的实时交通图像分类。我们实现了一个系统：低功耗摄像头节点可以使用其中央处理器执行设备端推理，也可以将处理任务卸载至 NVIDIA Jetson Nano，以进行图形处理器加速推理。通过定义推理延迟和丢帧率等正式性能指标，我们能够评估本地处理和卸载处理策略的可行性。

我们的结果表明，纯本地处理仅适用于低帧率下的轻量级人工智能模型。例如，YOLOv5n 在 1 FPS 下处理了全部帧，未出现丢帧。然而，提高帧率或增加模型复杂度会迅速导致丢帧率超过运行阈值，在某些配置下甚至达到 98%。相比之下，在所有测试模型中，通过 GPU 卸载至 Jetson Nano，即使在低至中等帧率下也能保持足够的处理能力；对于较高帧率下的复杂模型，性能下降也仅 minimal。

这些发现凸显了通过 Wi-Fi 进行 GPU 加速卸载对于在城市交通监控场景中实现可靠、低延迟性能的价值 [22] [15]，且无需昂贵的 5G 或有线基础设施 [9]。该方法对于 **超本地部署** 尤其具有吸引力，因为其覆盖范围限制在单个 Wi-Fi 小区内，能够以较低成本提供高性能，成为基础设施要求更高的解决方案的替代方案。除具体的交通监控应用外，所提出的框架还展示了基于 Wi-Fi 的边缘计算在其他城市感知应用中的广泛可行性，前提是对计算负载和网络条件进行适当管理。这使本地化 Wi-Fi 边缘人工智能成为智慧城市服务的实用支撑，适用于传统广域边缘计算或云端部署难以实施的环境。

致谢

这项工作是一篇预印本！官方数字对象标识符：
https://doi.org/10.34749/ 3061-1008.2025.7

本研究获得奥地利联邦教育、科学与研究部的 OeAD 机构以及马其顿教育与科学部的资助，项目编号为 MK10/2024。

参考文献

[1] Md Eshrat E Alahi、Arsanchai Sukkuea、Fahmida Wazed Tina、Anindya Nag、Wattanapong Kurdthongmee、Korakot Suwannarat 和 Subhas Chandra Mukhopadhyay。2023。物联网赋能技术与人工智能

使用低功耗物联网摄像头进行智慧城市交通监控的支持 Wi-Fi 的边缘智能框架

（AI）在智慧城市场景中的应用：最新进展与未来趋势。《传感器》23，11（2023），5206。[2] Johan Barthélemy、Nicolas Verstaevel、Hugh Forehead 和 Pascal Perez。2019。智慧城市中的边缘计算视频分析与实时交通监控。《传感器》19，9（2019），2048。[3] Marc Carrascosa-Zamacois、Lorenzo Galati-Giordano、Francesc Wilhelmi、Gianluca Fontanesi、Anders Jonsson、Giovanni Geraci 和 Boris Bellalta。2024。MLO的XR流媒体传输性能评估：Wi-Fi 7 能否满足预期？载于*2024 IEEE第29届计算机辅助通信链路与网络建模和设计国际研讨会（CAMAD）*。IEEE，1–6。[4] Ning Chen、Tie Qiu、Laiping Zhao、Xiaobo Zhou 和 Huansheng Ning。2021。智慧城市中物联网的边缘智能网络优化。《*IEEE*无线通信》28，2（2021），26–31。doi:10.1109/MWC.001.2000243[5] Cailian Deng、Xuming Fang、Xiao Han、Xianbin Wang、Li Yan、Rong He、Yan Long 和 Yuchen Guo。2020。IEEE 802.11be Wi-Fi 7：新挑战与新机遇。《*IEEE*通信调查与教程》22，4（2020），2136–2166。[6] Laura García-García、Jose M Jiménez、Miran Taha Abdullah Abdullah 和 Jaime Lloret。2018。智慧城市中的物联网无线技术。《网络协议与算法》10，1（2018），23–64。[7] Raj Hakani 和 Abhishek Rawat。2024。利用YOLOv9和NVIDIA Jetson Nano进行边缘计算驱动的实时无人机检测。《无人机》8，11（2024），680。[8] Roger Pierre Fabris Hoefel。2024。相位噪声和频偏对802.11be与802.11bn无线局域网中4K-QAM和16K-QAM性能的影响。载于*2024 IEEE*无线通信与网络会议（*WCNC*）。1–6。[9] Kurt Horvath、Dragi Kimovski、Stojan Kitanov 和 Radu Prodan。2025。利用人工智能和6G增强交通安全：时延要求与实时威胁检测。载于*2025*年第*10*届信息与网络技术国际会议（*ICINT*）。IEEE，129–136。[10] Kurt Horvath、Shpresa Tuda、Blerta Idrizi、Stojan Kitanov、Fisnik Doko 和 Dragi Kimovski。2025。面向边缘人工智能的6G基础设施：分析视角。*arXiv*预印本 *arXiv:2506.10570*（2025）。[11] Monica Karel Huerta、Jessica Garizurieta、Rubén González、Luis-Ángel Infante、Melina Horna、Renato Rivera 和 Roger Clotet。2023。作为促进韦拉克鲁斯南部（墨西哥）社会包容工具的长距离Wi-Fi网络。《可持续性》15，13（2023），9939。[12] Glenn Jocher、Ayush Chaurasia、Jirka Borovec 和 Ultralytics。2020。YOLOv5。https://github.com/ultralytics/yolov5。[13] Dragi Kimovski、Roland Mathá、Josef Hammer、Narges Mehran、Hermann Hellwagner 和 Radu Prodan。2021。云、雾还是边缘：在哪里进行计算？*IEEE Internet Computing* 25，4（2021）：30–36。doi:10.1109/MIC.2021.3050613[14] 数字对象标识符：。/。Zhiming Luo、Frederic Branchaud-Charron、Carl Lemaire、Janusz Konrad、Shaozi Li、Akshaya Mishra、Andrew Achkar、Justin Eichel 和 Pierre-Marc Jodoin。2018 年。MIO-TCD：用于车辆分类和定位的新基准数据集。*IEEE Transactions on Image Processing* 27，10（2018），5129–5141。[15] Aale Luusua、Johanna Ylipulli、Marcus Foth 和 Alessandro Aurigi。2023 年。城市人工智能：理解人工智能在智慧城市中的新兴作用。*AI & society* 38，3（2023），1039–1044。[16] Madhusri Maity、Sriparna Banerjee 和 Sheli Sinha Chaudhuri。2021 年。基于 Faster R-CNN 和 YOLO 的车辆检测：一项调查。载于*2021 5th international conference on computing methodologies and communication (ICCMC)*。IEEE，1442–1447。[17] Gaurav Menghani。2023 年。高效深度学习：关于使深度学习模型更小、更快、更优的调查。*Comput. Surveys* 55，12（2023），1–37。[18] Andreas Møgelmose。2019 年。多视角交通路口数据集。Kaggle。
h
t
t
ps://www.kaggle.co
m/datasets/andreasmoegelmose/multiview-traffic-intersection-dataset[19] Zhaolong Ning、Jun Huang 和 Xiaojie Wang。2019 年。车载雾计算：实现智慧城市的实时交通管理。*IEEE Wireless Communications* 26，1（2019），87–93。[20] Héctor Rodríguez-Rangel、Luis Alberto Morales-Rosales、Rafael Imperial-Rojo、Mario Alberto Roman-Garay、Gloria Ekaterine Peralta-Peñuñuri 和 Mariana Lobato-Báez。2022 年。基于 YOLO 车辆检测的实时速度估计中统计和人工智能算法的分析。*Applied Sciences* 12，6（2022），2907。[21] Tarik Taleb、Sunny Dutta、Adlen Ksentini、Muddesar Iqbal 和 Hannu Flinck。2017 年。移动边缘计算在使城市更加智能方面的潜力。*IEEE Communications Magazine* 55，3（2017），38–43。数字对象标识符：10.1109/MCOM.2017.1600249CM [22] Radosław Wolniak 和 Kinga Stecuła。2024 年。智慧城市中的人工智能——应用、障碍与未来方向：一篇综述。*Smart cities* 7，3（2024），1346–1389。[23] Yu Zhang、Zhongyin Guo、Jianqing Wu、Yuan Tian、Haotian Tang 和 Xinming Guo。2022 年。基于改进 YOLO v5 的实时车辆检测。《可持续性》14，19（2022），12274。[24] Muhammad Azhad Bin Zuraimi 和 Fadhlan Hafizhelmi Kamaru Zaman。2021 年。使用 YOLO 和 DeepSORT 进行车辆检测与跟踪。载于*2021 IEEE 11th IEEE symposium on computer applications & industrial electronics (ISCAIE)*。IEEE，23–29。