用于去中心化任务卸载的设备端深度强化学习：训练过程中的性能权衡

Gorka Nieto ,Idoia de laIglesia、Cristina Perfecto 和 UnaiLopez-Novoa

arXiv:2601.03976v1 [cs.ET] 7 Jan 2026

摘要

允许能力较弱的设备将计算任务卸载到性能更强的设备或服务器上，能够开发出无法在设备本身正常运行的新应用。决定在何处以及为何运行这些应用是一项复杂的任务。因此，人们采用了不同的方法来制定卸载决策。在本研究中，我们提出一种去中心化的深度强化学习（DRL）智能体，以解决计算位置的选择问题。不同于大多数现有研究，我们在一个真实测试平台上对其进行分析，该平台由多种运行智能体的边缘设备组成，用于确定每项任务的执行位置。这些设备通过 5G 通信连接到多接入边缘计算（MEC）服务器和云服务器。我们不仅评估智能体满足任务要求的性能，还考察在本地运行此类智能体所带来的影响，并从延迟和能耗角度评估本地训练与远程训练之间的权衡。

索引词

物联网（IoT）、边缘-云连续体、任务卸载、深度强化学习（DRL）、5G。

Gorka Nieto 任职于西班牙阿拉萨特的巴斯克研究与技术联盟（BRTA）旗下 Ikerlan技术研究中心，同时任职于西班牙巴斯克大学（EHU）毕尔巴鄂工程学院（电子邮箱：gni- eto004@ikasle.ehu.eus）。

Idoia de la Iglesia 任职于西班牙阿拉萨特的巴斯克研究与技术联盟（BRTA）旗下 Ikerlan技术研究中心（电子邮箱：idelaiglesia@ikerlan.es）。

Cristina Perfecto 和 Unai Lopez-Novoa 任职于西班牙巴斯克大学（EHU）毕尔巴鄂工程学院（电子邮箱：cristina.perfecto@ehu.eus 和 unai.lopez@ehu.eus）。

本研究部分得到了COGNIT项目（资助协议编号：101092711）的支持，该项目由欧盟委员会资助；同时也得到了巴斯克政府资助的XWAVE研究资助（KK2025/00056）的支持。

I. 引言

随着物联网不断拓展至智慧城市、医疗保健或工业自动化等更多领域，能够感知、处理数据并相互通信的互联物联网设备数量持续快速增长。这导致生成、处理和交换的数据量不断增加，且数据异质性日益增强 [1]。此外，这些设备往往面临一个关键问题：能耗。大多数物联网节点受资源限制，处理能力有限。尽管如此，它们通常依赖有限的电池电量或能量采集，这直接影响其运行寿命和可扩展性 [2]。

由于上述局限性，物联网设备有时可能不适用，因此可能需要其他类型的设备。例如，为了处理物联网设备生成的大量数据并创造更有价值的服务，近年来，将物联网与云计算相结合开辟了新的解决方案 [3]。然而，在延迟至关重要的特定场景中，云计算范式并不适用。因此，多接入边缘计算范式应运而生，通过分别扩展设备的计算资源能力，以及减少由距离造成的延迟，克服物联网和云计算概念各自的局限性，从而使其能够应用于更多场景，例如医疗保健、智慧城市或时延敏感型多媒体通信等 [4]。

这些范式的结合构成了物联网-边缘-云连续体。该统一系统整合了物联网设备、边缘和云资源，使计算任务能够分布在异构设备上执行，涵盖从资源受限的物联网节点到强大的云服务器。作为其缺点，这也带来了管理异构性、分布式数据和可扩展性方面的重大挑战 [5]。

因此，任务卸载已成为一种极为有效的策略，可用于优化资源、缩短延迟、降低能耗等。此外，随着物联网设备需要运行要求越来越高的应用，边缘设备对智能任务卸载机制的需求日益增长 [6]。而且，由于常见工业环境（例如无线通信条件以及需求各异的应用）具有动态性，相关条件可能发生变化，因此需要能够适应这些变化的工具。这正是机器学习（ML）技术，尤其是深度强化学习（DRL）技术在这一主题中变得如此重要的原因：它们能够适应不断变化的条件 [7], ，从而实现能够自主行动的人工智能（AI）智能体。

此外，与中心化方法相比，去中心化方法能够实现自主决策，从而提高可用性和容错性，同时避免单点故障和连接问题的风险 [8]。设备端机器学习是指直接在设备上运行机器学习任务（推理或训练），而不依赖外部服务器或云服务。尽管 TensorFlow Lite 等框架针对在配备操作系统的嵌入式设备上进行推理做了优化，但仍需要能够简化整个开发流程的工具，涵盖从模型训练和优化到部署和监控的各个环节，因为这些框架并未针对训练进行优化 [9]。

因此，本文分析了用于计算卸载的设备端深度强化学习智能体的行为与性能，重点关注训练过程。为此，我们在设备端实现了深度强化学习模型的完整流程，使人工智能智能体能够在线学习。这种方法可以准确评估设备端深度强化学习智能体训练所涉及的实际权衡，并在动态的真实环境下，将其与集中式训练、分布式推理方案进行比较，展示该方法的可行性及其优势与局限性。我们的结果表明，使用轻量级深度强化学习模型时，设备端训练带来的计算耗时和能耗均在合理范围内，且整体性能优于将训练过程卸载至远程位置的方案。

本文其余部分安排如下。第 II 节概述相关工作。第 III 节介绍问题 formul化及所提出的解决方案，并在第 IV 节所述的真实环境中对其进行评估。第 V 节分析结果，最后，第 VI 节总结本文并探讨未来研究的潜在方向。

II. 相关工作

任务卸载已通过多种决策技术得到广泛研究，包括传统优化方法 [10], 启发式 [11] 和元启发式 [12] 方法、控制理论方法 [13], 博弈论方法 [14], 基于机器学习的方法（例如监督学习（SL） [15]或强化学习（RL） [16]），以及结合多种范式的混合方法 [17]。传统优化方法虽能确保准确性，但由于计算复杂度高，不适用于大规模动态环境 [18]。启发式和元启发式方法能够更快地给出解决方案，但通常是静态且针对特定问题的，适应网络状态变化的能力有限 [19]。控制理论方法

在连续动力学场景中有效，但需要准确的系统模型；由于现实世界的不确定性，这类模型难以构建 [20]。相比之下，博弈论方法通常依赖于关于智能体知识的强假设，且计算开销较大 [21]。相比之下，深度强化学习（DRL）提供了一种无模型的自适应框架，其中每个人工智能智能体都会学习一项策略，以最大化围绕优化目标定义的奖励 [22]。

尽管许多方法（例如 [23], ）提出了集中式决策的深度强化学习方法，但正如 [24]所示，分布式或去中心化的深度强化学习方法正变得越来越重要。在这些研究中，尽管推理或数据收集阶段采用了去中心化执行，但其中大多数方法仍然依赖集中式训练范式。在此类部署中，从分布式人工智能智能体收集的经验数据会被汇总，并用于更新共享策略或价值函数，而这些策略或函数通常由中央服务器或协调器维护。此外，集中式训练需要频繁地将经验数据传输至中央服务器，这可能引入延迟并成为瓶颈——尤其是在涉及众多智能体的场景中 [25]。此外，边缘部署环境中常用的无线网络通常不稳定，容易出现丢包和波动带宽等问题，从而妨碍同步和数据一致性。

尽管人们对去中心化/分布式计算卸载和基于边缘的深度强化学习日益关注，但很少有研究探讨在真实边缘设备（如*Jetson Nano*或树莓派）上运行此类系统所面临的实际挑战。能够在现实部署环境中进行实证评估的研究则更少，因此人们对于此类系统在实际约束条件（如资源有限和连接不稳定）下的性能表现仍缺乏了解。 [26],所介绍的工作就是其中一个例子，该工作为卸载决策提供了机器学习技术的实际应用。其实现了一种分布式集体深度强化学习算法，利用了来自不同多接入边缘计算环境的知识。该实现依赖于静态室内环境中的稳定 Wi-Fi连接，从而最大限度地降低信道波动对模型的影响，并免除了在动态环境中进行进一步分析的需要。

关于设备端机器学习，一些研究采用开放神经网络交换 (ONNX) 或 *TensorFlow Lite* 等轻量级推理引擎部署预训练模型。这些引擎非常适合资源受限的设备，但不支持设备端训练，因此限制了模型的适应能力 [27]。这一局限以及设备端训练面临的其他挑战，使得人们需要更灵活、功能更强大的机器学习框架，以支持边缘设备上的推理和训练。例如， [28] 的研究探讨了使用高层级 (*PyTorch*) 和低层级 (*LibTorch*) 库，在物联网设备上进行基于机器学习的图像分类。他们评估了

模型的性能，并量化了机器学习模型训练的绿色足迹，包括执行时间、内存使用量、功耗和 CPU温度等指标。

鉴于此，尽管人们有意在边缘部署深度强化学习，以保持这些模型的适应性，尤其是在计算卸载场景中，但对本地训练智能体进行的现实世界评估仍然十分有限。现有研究大多聚焦于理论模型或仿真，忽视了在不稳定的现实部署环境中进行在线训练时遇到的实际约束和权衡。在此类环境中，由于环境难以预测，模型可能出现异常行为，而在受控的仿真环境中，这种情况可能会被忽略。此外，在这些条件下，集中式训练与设备端学习之间的比较分析尚付阙如。为弥补这一空白，本文实现了智能卸载策略，明确考量本地训练的成本与收益，尤其是在连接时断时续或计算资源有限的现实环境中。该方法能够实现持续设备端学习，从而在集中式训练不切实际或效果不佳时支持实时决策。

III. 问题表述与评估算法

本节介绍待解决的优化问题以及用于解决该问题的方法。在本研究中，不同的用户设备（UE）必须决定是在本地运行接收到的计算任务，还是将其卸载至多接入边缘计算（MEC）或云服务器（见图1）。因此，除了执行这些接收到的任务外，它们还必须运行一个卸载决策算法实例，以决定是本地执行任务还是将其卸载。

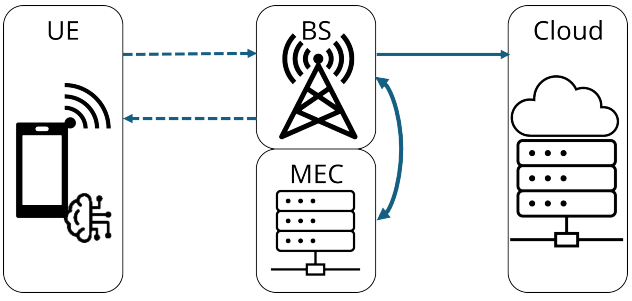


图1. 系统模型中涉及的实体

我们首先介绍用户设备可以收集的关键指标，这些指标将由算法分析，以便做出明智的决策。接下来，我们概述算法可以采取的动作，并

然后解释优化目标。随后，我们介绍所评估的深度强化学习算法，以及本方法中的训练方案。

A. 问题 *formul*化

每个用户设备（UE）上都部署了一个深度强化学习智能体。UE 会持续实时监控特定参数，这些参数将作为深度强化学习算法决策过程的输入。

该问题建模为马尔可夫决策过程（MDP）。在每个时间步 i，都会有一个新任务（Ψ）到达，深度强化学习智能体会评估当前状态 s_i。如公式 **1** 所示，该状态由一个五元组向量空间定义：

$$s_i = \{R_i^n, R_i^m, T_i^n, C_i, \delta_i\} \tag{1}$$

其中，R_i^n 和 R_i^m分别表示用户设备和移动边缘计算服务器当前的可用资源；T_i^n表示用户设备通过所连接的基站（BS）获得的吞吐量；C_i 和 δ_i 分别表示新到达任务的计算规模和时延约束。

由于从云端服务器获取指标可能会产生较高延迟，我们假设云端在任何场景下都有充足的计算资源，并且始终配有多台备用服务器为客户端提供服务。因此，做出决策时不会考虑这些信息。因此，所选状态变量构成了一组精简但足够的实时指标，可在资源受限设备上可靠地监控。为避免额外的内存开销，并确保持续进行设备端学习的可行性，我们不纳入历史或长期统计数据。

考虑到前述参数，深度强化学习智能体必须决定每个传入任务的部署和运行位置。该动作表示传入任务的执行方式，可表示为 $\alpha_i \in \{0, 1, 2\}$，其中包括 $\alpha_i = 0$ 本地执行、$\alpha_i = 1$ 卸载至移动边缘计算服务器，以及 $\alpha_i = 2$卸载至云服务器。

我们将性能指标定义为在计算任务成功的条件下用户设备的能耗。计算任务是否成功或失败，取决于其是否满足时延要求。若不满足该要求，性能将产生惩罚值 η。需要注意的是，总延迟包括决策时间、传输与接收时间（如果传输了数据）以及处理时间，而这些时间均受构成 s_i 的参数和所选动作 α_i 的影响。

因此，深度强化学习智能体 r 的奖励函数等于性能值，如公式（2）所示。

$$r_i = \begin{cases} -e_i & \text{, if success} \\ \eta & \text{, if fail} \end{cases} \tag{2}$$

这种奖励函数在能源效率与任务可靠性之间取得平衡，避免在动态网络条件下做出可能违反时延约束的卸载决策。因此，它反映了现实边缘环境中的实际部署需求。

优化问题的目标是最大化总体奖励（公式 2），由此定义公式（3）中的问题：

$$\begin{aligned} &\max_{\alpha_i} \quad \sum_{i=0}^{T-1} r_i \\ &\text{s.t.} \\ &c_1: \quad \alpha_i \in \{0,1,2\} \quad ,\forall i \in I \end{aligned} \tag{3}$$

其中， c_1 是一个动作约束，表示每个任务只能由用户设备自身执行、卸载到移动边缘计算服务器，或卸载到云服务器。

总之，公式 1 表示算法在决定动作以优化公式 2 所定义的性能时所考虑的参数。本质上，5 元组向量 s_i 提供了必要的输入，使算法能够做出明智的决策，并直接影响性能结果。

B. 深度强化学习算法

由于这两种深度强化学习算法在稳定性和计算复杂度方面各有权衡，本小节将对其进行实现与说明：演员-评论家（AC）结合策略函数与价值函数，以实现更稳定、高效的学习；深度 Q 网络（DQN）则利用深度神经网络（DNN）进行基于价值的学习，从而估计动作价值。这些算法已在我们先前的工作中得到验证[29]。

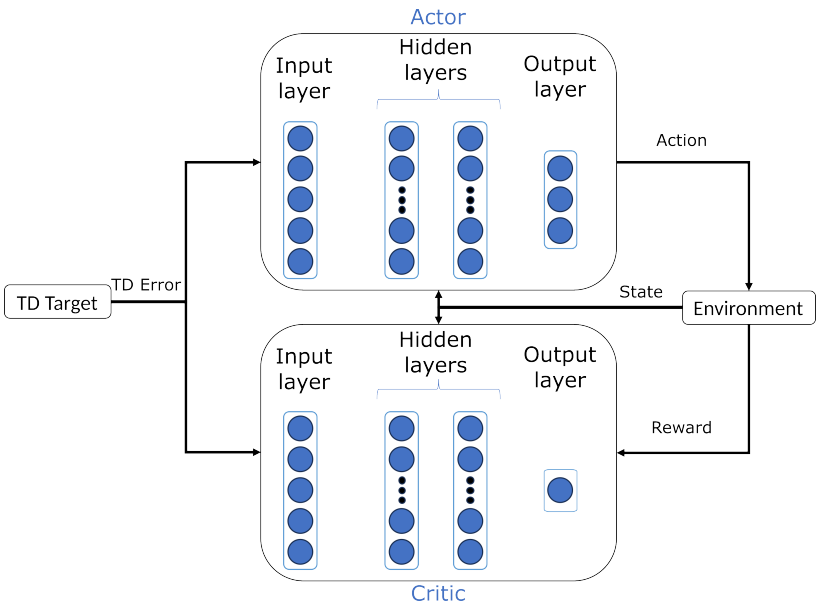


图 2. 所实现的行动者-评论家算法架构

1）演员-评论家：如图 2 所示，AC算法在基于策略的方法与基于价值的方法之间取得了平衡。

该算法包含 2 个结构相似的网络（行动者和评论家）。行动者和评论家网络均接收相同的 5 维状态向量作为输入。行动者和评论家网络分别由以下部分组成：

- **输入层**：接收 5 维状态向量。
- **1第 1 个隐藏层**：包含 256 个神经元并采用双曲正切（TanH）激活的全连接（Dense）层。
- **2第 2 个隐藏层**：另一个包含 256 个神经元并采用 TanH 激活的全连接层。
- **输出层**：对于行动者，该层是一个包含 3 个神经元的全连接层（每个神经元对应一个可能的动作），随后采用 Softmax 激活函数，以生成可能动作上的概率分布。对于评论家，该层是一个包含 1 个神经元的全连接层，因此输出的是当前状态的标量价值估计，且不使用激活函数。

训练通过在线学习进行，智能体与环境交互，并在每一步后根据观测到的奖励和状态转移更新其网络，从而无需在训练过程中使用批次。评论家经过训练以最小化

时序差分（TD）误差，而行动者则使用策略梯度方法进行更新，并以 TD优势加权。两个网络均使用 Adam优化器进行优化，学习率固定为 $1 \cdot 10^{-5}$。

2）深度 *Q* 网络：DQN算法是一种基于价值的强化学习方法，它通过使用深度神经网络近似 Q值函数，扩展了传统的Q学习，从而能够处理更大的状态空间。其配置如图 3 所示。

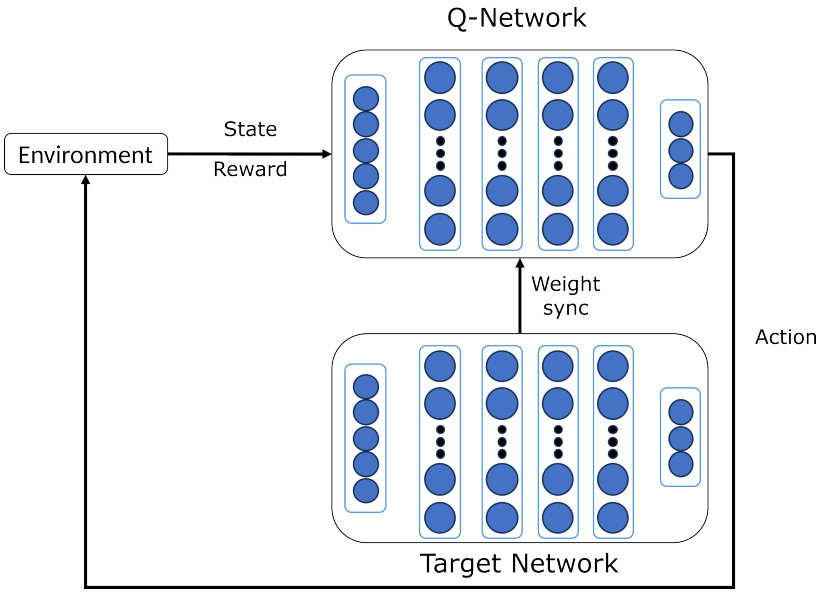


图 3. DQN模型

用于近似 Q值函数的深度神经网络以当前状态作为输入，并输出该状态下每个可能动作对应的 Q 值。在本例中，该神经网络由四个隐藏层组成，每层包含 256 个神经元，并采用修正线性单元（ReLU）激活函数。生成的 Q 值用于根据 ε-贪心策略，确定当前状态下期望奖励最高的动作。该策略通过以 ϵ 的概率随机选择一个动作来平衡探索与利用；否则，选择 Q 值最高的动作。此模型采用 Adam优化器，学习率为 $1 \cdot 10^{-5}$。

两种算法均采用在线持续学习方法进行训练，不使用大型经验回放缓冲区或批量更新。这一设计选择反映了现实的边缘部署场景：此类场景中的内存可用性有限，且需要及时适应。

IV. 实验设置

本节介绍用于开展下一节所述实证实验的测试平台。首先，我们将详细描述该测试平台，随后说明用于分析设备端训练过程性能的评估方法。

A. 测试平台

我们搭建了一个如图 4 所示结构的测试平台，用于开展后续章节所述的实证实验。如图所示，我们通过为各个用户设备分别配备一个支持5G的调制解调器，将不同的用户设备连接到专用5G网络，因为它们不具备集成的5G通信能力。该5G网络由专用基站提供。随后，基站连接到一台移动边缘计算服务器和一台云服务器。为了模拟访问后者所需的网络行为，我们使用了自主开发的网络仿真器。最后，其他模拟的用户设备也连接到同一基站，以生成可能影响网络状态的流量。

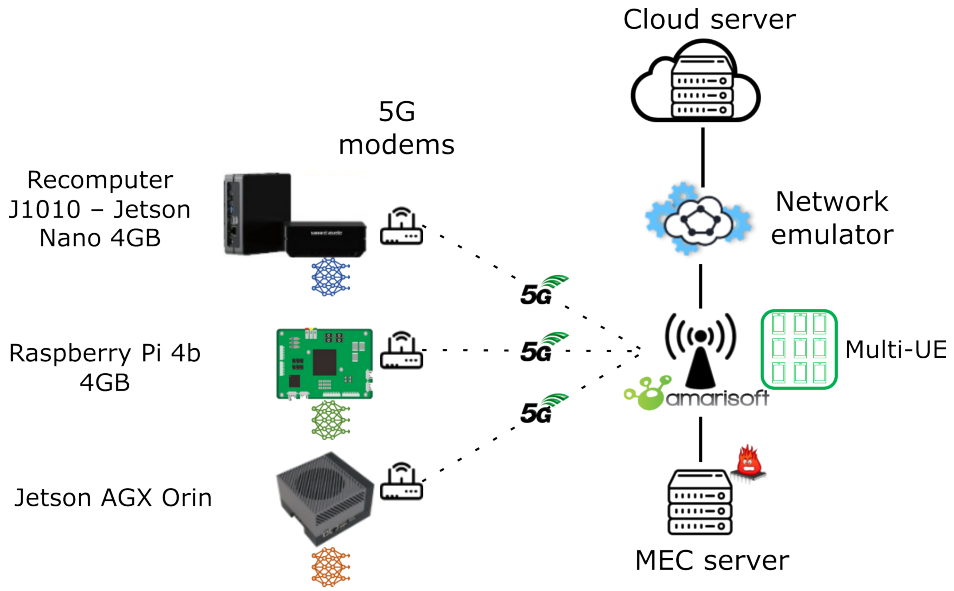


图 4. 评估测试平台的架构

在本研究中，如第 III-A 节所述，用户设备必须运行一个卸载决策代理实例，同时还必须运行一个工具来获取其能耗。我们实现了相应的工具，可根据每台用户设备所能使用的工具提供实际测量值或估算值，具体如下：

- **树莓派 Raspberry Pi 4 型号**：配备 1.5 吉赫兹 64 位四核 *ARMCortex-A72*处理器和 4GB 随机存取存储器，运行 树莓派 OS 11 版。在这种情况下，已安装 *Powerjoular*[30]，用于测量能耗，并可提供可靠的中央处理器能耗估算值。

- **reComputer J1010 / Jetson Nano**：搭载 1.43 吉赫兹四核*ARM Cortex-A57* 中央处理器和 128 核*NVIDIAMaxwell* 图形处理器，配备 4GB LPDDR4 内存和 16GB eMMC 存储，在*JetsonNano*量产模块上运行*JetPack* 4.6.2。能耗数据通过读取 IIO 电源传感器[1]获取。

- **Jetson AGX Orin** 搭载 2.2 吉赫兹 12 核*ARM Cortex-A78AE* 中央处理器和 2048 核 *NVIDIA Ampere* 图形处理器 （含 64 个张量核心），配备 64GB LPDDR5 内存和 64GB eMMC 存储，运行*JetPack* SDK 6.5。通过解析*tegrastats* 实用程序的输出，获取图形处理器和中央处理器的能耗。

为 了向用户设备提供 5G 连接，我们使用了两种不同的 5G 模块 ：

- **Quectel RM520N-GL**。这是一款面向全球 6 GHz 以下频段独立组网部署的 5G NR 模块，在独立组网（SA）模式下，峰值下行速率最高可达 2.4 Gbps，上行速率最高可达 900 Mbps。它支持 3GPP Release 16，非常适用于高吞吐量、低延迟应用。该模块搭配官方评估板套件，通过 USB 连接与树莓派 相连，并通过 M.2 转接器与*Jetson AGX Orin*相连。

- **SIMcom SIM8262E-M2**。这是一款基于*SnapdragonX62*平台的紧凑型 5G 模块，针对 SA 网络进行了优化，支持最高 2.4 Gbps 的下行速率和最高 1 Gbps 的上行速率。该模块符合 3GPP Release 16，专为需要可靠 5G SA 连接的嵌入式应用而设计。其 M.2 外形规格支持 USB 3.1，因此可用于连接*reComputer J1010*。

5G 基站是一个*Amarisoft* 新无线电（NR）SA 基站（版本 2024年-09-04），完全符合 3GPP Release 17，可对 Release 17 功能进行详细测试和部署。

服务器运行一个 Flask[2] 服务，该服务公开了多个 REST 端点。两台服务器都提供一个端点，可通过 POST 请求接收文件、运行文件，并根据执行结果返回成功或错误消息。移动边缘计算服务器上的 Flask 服务还提供

[1]Jetson Nano 配备板载 INA3221 电源监视器，该监视器通过 Linux IIO 子系统（sysfs）提供访问。[2] Flask：https://flask.palletsprojects.com/

一个用于报告当前 CPU 使用率的端点，使用户设备能够在卸载任务前评估服务器状态。表 I 展示了公开方法的说明。

表 I 服务器端点

服务器	特征	HTTP 方法	输入	输出
MEC	运行任务	POST	Task	成功 / 错误
	获取负载	GET	-	CPU 使用率
云端	运行任务	POST	Task	成功 / 错误

服务器的特性如下：

- **移动边缘计算服务器** 运行 Ubuntu 20.04.3 LTS 平台，配备第 4 代英特尔酷睿 i7 中央处理器，主频为 2.00 吉赫兹，并搭载 8 GB 随机存取存储器。它位于用户设备附近数米处，并通过以太网电缆与基站进行物理连接，从而提供低延迟通信。可以使用 *stress-ng* 工具[3]对移动边缘计算服务器施加压力，以模拟用户定义的负载条件。

- **云服务器（CS）**同样运行 Ubuntu 20.04.3 LTS，配备第 4 代英特尔酷睿 i7 中央处理器，主频为 2.50 吉赫兹，并搭载 32 GB 随机存取存储器。通过添加**网络仿真器**，可以改变基站与云服务器之间的网络条件，从而模拟延迟、抖动、丢包、突发丢包相关性、数据损坏、重新排序和数据包重复。

最后，采用 TACLeBench 套件任务集生成器 [31]来生成任务，从而创建具有不同特征的可执行任务，包括不同的计算复杂度和时延约束。该工具生成能够反映现实应用的代表性可执行任务，同时允许调整任务属性，例如计算负载和时间约束。因此，会为每个待卸载任务生成合成数据，并据此以类似于 [32]，中的方法推导时延要求，其中每个任务的延迟要求（δ_i）建模为 C_i/β，且 β 在 $[9, 11]$ 中均匀分布。本文根据表 II 所示的配置生成任务。

[3]*stress-ng*：https://github.com/ColinIanKing/stress-ng

表 II 任务特征

	参数	值
任务集	最小全局利用率	3.9
	最大全局利用率	3.9
	利用率步长	0.1
	任务数	500
	任务集数	1
	最小周期	1e5
	最大周期	1e8
	周期步长	1000
	Seed	50
目标硬件	架构	x86
	CPU 速度（MHz）	1200.048
	核心数	4

B. 训练过程

该系统使用消息队列遥测传输（MQTT）协议作为客户端或用户设备与位于移动边缘计算服务器中的训练服务器之间的轻量级通信层，实现了演员-评论家（AC）或深度 Q 网络（DQN）深度强化学习智能体的分布式训练功能。

因此，每个用户设备都运行一个深度强化学习智能体（使用 *Tensorflow*[4] 和 *Keras*[5]创建），该智能体与环境交互并在设备端采取动作。关于训练过程，模型可以在本地训练，也可以收集经验元组（状态、动作、奖励和下一状态），并通过 MQTT 发送到多接入边缘计算服务器。当服务器发回更新后的神经网络（NN）权重时，客户端会立即更新其本地模型，从而使用更新后的模型为新到达的任务选择动作。

移动边缘计算服务器充当全局学习器，订阅来自所有用户设备的 MQTT 主题，以 *protobuf* 消息的形式接收序列化经验，利用接收到的数据更新相应模型，并将更新后的模型权重发布回相应的用户设备。这样，移动边缘计算服务器便可同时管理多个智能体，这些智能体通过设备 ID D 和模型类型 Π进行标识。

[4]Tensorflow: https://www.tensorflow.org/[5]
Keras: https://keras.io/

考虑到部分用户设备配备了图形处理器，*TensorFlow* 可以在可用时自动利用 GPU 加速。对于配备图形处理器的设备（例如 *Jetson AGX Orin*），可以通过设备配置显式启用或禁用图形处理器。移动边缘计算服务器不包含图形处理器；因此，即使用户设备支持 GPU 加速，在服务器上执行的训练也不会使用该功能。训练过程的伪代码在算法 1 中进行了说明。

算法 1 分布式强化学习框架

需要： 模型类型 Π，设备 *ID* D，代理地址，训练模式

使用模型类型 Π 及参数 θ_A初始化智能体 A
当 训练未完成**时执行**
    观察当前状态 s_i
    选择动作 $\alpha_i \sim \pi(s_i; \theta_A)$
    执行 α_i，观察奖励 r_i，下一状态 s_{i+1}
    如果 训练模式 = $remote$**则**
        将 $(s_i, \alpha_i, r_i, s_{i+1})$ 打包为经验消息
        发布到主题 client/D/experience/M/v1
        对于 每条收到的经验 $(s_i, \alpha_i, r_i, s_{i+1})$ **执行**
            识别智能体类型 M
            在 θ 上执行一个训练步骤
            通过权重消息广播 θ
        结束循环
    else
        如果选择了图形处理器**则**
            使用图形处理器和经验在本地训练智能体 $(s_i, \alpha_i, r_i, s_{i+1})$
        else
            使用中央处理器和经验在本地训练智能体 $(s_i, \alpha_i, r_i, s_{i+1})$
        结束条件判断
    结束条件判断
    如果 收到新的权重消息**则**
        更新本地参数 $\theta_A \leftarrow \theta_{server}$
    结束条件判断
结束循环

其中，主题结构 *client/D/experience/M/v1* 编码了设备 ID *D* 和模型类型*M*，使服务器能够将经验数据路由到正确的学习器实例。版本后缀*v1* 提供了命名空间隔离，便于未来进行模型修订。

每台设备都运行针对其硬件定制的*TensorFlow*版本。*reComputerJ1010*和 *RaspberryPi 4* 使用*Tensorflow* 2.14 版本，而*Jetson AGXOrin*使用*Tensorflow* 2.16 版本。

图 5 展示了工作流模式。设备发布其经验数据，并上传各自的深度强化学习模型。MQTT 代理运行在移动边缘计算服务器上，并监听所有网络接口，通过服务器学习器接收经验数据，并通过模型接收器管理传入的模型。服务器模型接收器订阅来自设备的上传主题，并收集和组装模型分块。对于服务器学习器，每个服务器学习器进程都由服务器管理器在适用于其设备类型的虚拟环境中启动。这样，它便会订阅分配给自己的经验主题，使用收到的经验数据在服务器上本地训练深度强化学习智能体，并通过 MQTT 将更新后的模型权重发布回设备。这种设计使每个学习器能够针对特定设备和模型类型进行专门化，同时集中共享计算资源。

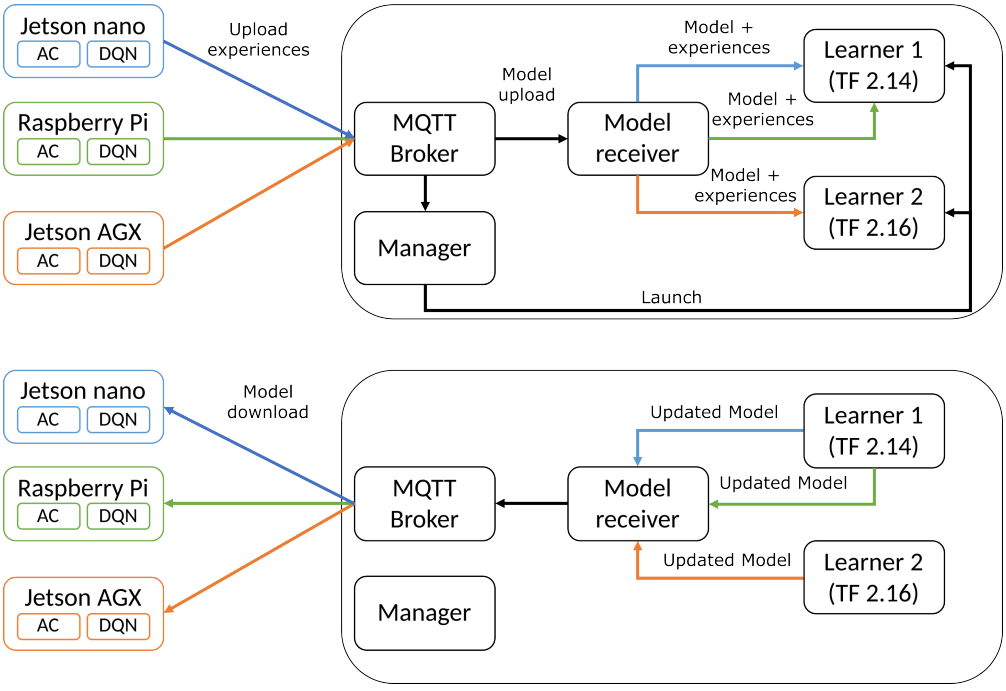


图 5. MQTT 进程

V. 结果

本节全面评估了异构边缘平台上的设备端训练和远程深度强化学习训练。分析结果时重点关注训练时间、能耗和累积奖励，旨在评估现实边缘环境中设备端学习的实际可行性和效率。

我们在持续学习场景中评估设备端深度强化学习，因此每台设备都会针对每个到来的任务即时学习。这样，对于每项任务，设备不仅会使用深度强化学习策略进行推理，还会更新已部署智能体，执行任务特定训练，而非批量训练。

我们首先对设备进行特性分析，以比较其峰值计算能力；测试使用 openBLAS[6] 和 cuBLAS[7], 完成，结果如表 III 所示。深度强化学习在训练期间需要反复执行前向和反向传播，而底层的矩阵运算使这些计算开销很大。为了评估这些运算在所测试平台上的理论性能，我们在中央处理器和图形处理器上分别测量单精度通用矩阵乘法（SGEMM）和双精度通用矩阵乘法（DGEMM）。SGEMM 是典型神经网络推理和训练中的主要运算，而 DGEMM 则能反映需要更高数值精度的任务中的双精度性能。基准测试还表明，无论推理是否可能受中央处理器性能限制，神经网络训练都能从设备上的图形处理器加速中获益。值得注意的是，树莓派 和 r*eComputer J1010/JetsonNano*最多支持 4 个中央处理器线程，而*JetsonAGXOrin*最多支持 8 个线程。在图形处理器能力方面，*Jetson AGX Orin*64GB 配备 2,048 个 CUDA 核心 [8], ，而 *reComputer J1010* 配备 128 个 CUDA 核心 [9]。

我们使用由第 IV 节所述的开源框架生成的多种合成任务组成的任务集，用于基准测试和最坏执行时间（WCET）研究。

最后，关于测试，可以配置不同的场景，以评估系统在各种条件下的性能。这些配置选项如表 IV 所示。

[6]OpenBLAS: http://www.openmathlib.org/OpenBLAS/[7]cuBLAS: https://developer.nvidia.com/cublas[8]*Jetson AGXOrin* 数据手册 : https://nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf[9]*reComputer J1010*数据手册: https://files.seeedstudio.com/wiki/reComputer/reComputer-J1010-datasheet.pdf

表 III 每台设备的每秒操作数

设备			单精度通用矩阵乘法（十亿次浮点运算每秒）		双精度通用矩阵乘法（十亿次浮点运算每秒）	
Name	**HW**	**中央处理器 / GPU**	**Mono 线程**	**多 线程**	**Mono 线程**	**多 线程**
Jetson AGX Orin	12x A78AE 2.2 吉赫兹	CPU	24.03	108.93	12.36	70.19
		GPU	-	435.61	-	14.66
reComputer J1010 - Jetson nano	4xA57 1.4 吉赫兹	CPU	9.11	29.50	1.60	6.10
		GPU	-	28.09	-	6.31
Raspberry Pi 4b	4xA72 1.5 吉赫兹	CPU	8.54	20.06	4.56	12.25

表 IV 设备特定约束下的可配置参数和选项

参数	选项
设备	树莓派 ||Jetson Nano ||Jetson AGX Orin
算法	演员-评论家 || 深度 Q 网络
训练位置	本地 || 多接入边缘计算
多接入边缘计算压力	无压力 ||压力
Mode	中央处理器 ||图形处理器（树莓派上不可用）
多用户设备	无多用户设备 || 多用户设备

因此，我们在两种不同的场景下对系统进行测试：

- **场景1：理想条件**。在此场景中，没有压力，也没有虚拟用户设备与网络交互。

- **场景 2：并发用户设备**。在此场景中，我们模拟 50 个并发用户设备，并将每个设备配置为使用 *iperf*[10] 向移动边缘计算服务器生成流量。总负载

[10]Iperf：https://iperf.fr/

相当于 50 个用户设备各以 10 Mbps 的速率传输数据。此设置还会使移动边缘计算服务器承受压力，其处理能力负载在 80% 至 100% 之间波动，并遵循预定义的分布，从而使我们能够评估高负载、多用户条件下的性能，并评估其在实际部署中的可扩展性。

首先，我们在场景1中（理想条件下）分别在本地运行两个深度强化学习智能体，如图 6 所示。在所有设备上，与深度 Q 网络相比，演员-评论家的训练时间始终更长，但两种算法在各设备上的表现趋势相似。此外，在所评估的边缘场景中，GPU 加速并未提升深度强化学习训练性能，因为额外的数据传输和内核启动开销会导致能耗更高、收敛时间更长，相比之下，基于中央处理器的训练表现更好。此外，缺少大型批次会降低并行化效率，导致图形处理器核心数未被充分利用，并增加每次更新的延迟。相比之下，中央处理器更擅长处理频繁的小规模更新。树莓派的训练时间最长，而*reComputerJ1010* 的训练时间最短。

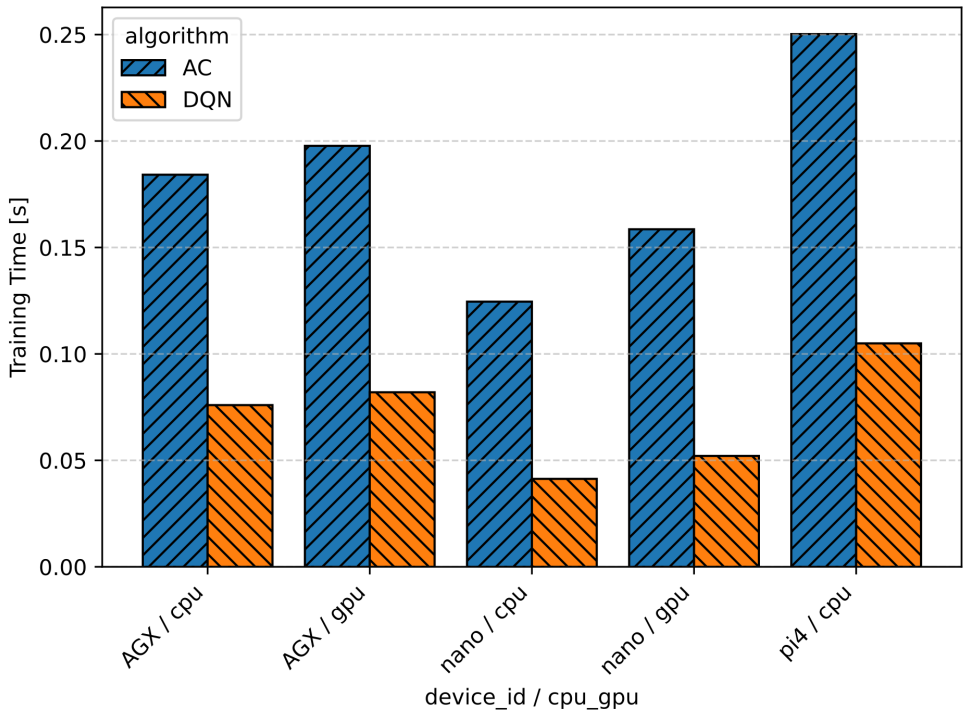


图 6. 场景1中各设备上每种算法的平均本地训练时间

在相同场景下，我们通过本地训练评估了两种深度强化学习算法在每台边缘设备上的性能。最佳结果出现在 *Jetson AGX Orin* 上，因为其更强的

计算能力使算法能够在本地运行更多任务，从而降低延迟。相比之下，在资源受限的平台（如 树莓派 和 *reComputerJ1010*）上，奖励显著下降。它们有限的处理能力导致本地运行时延迟更长，并且对于演员-评论家和深度 Q 网络，它们通常会更频繁地决定进行卸载。图 7 展示了这些算法在奖励方面的性能，该奖励定义见第 III-A 节。

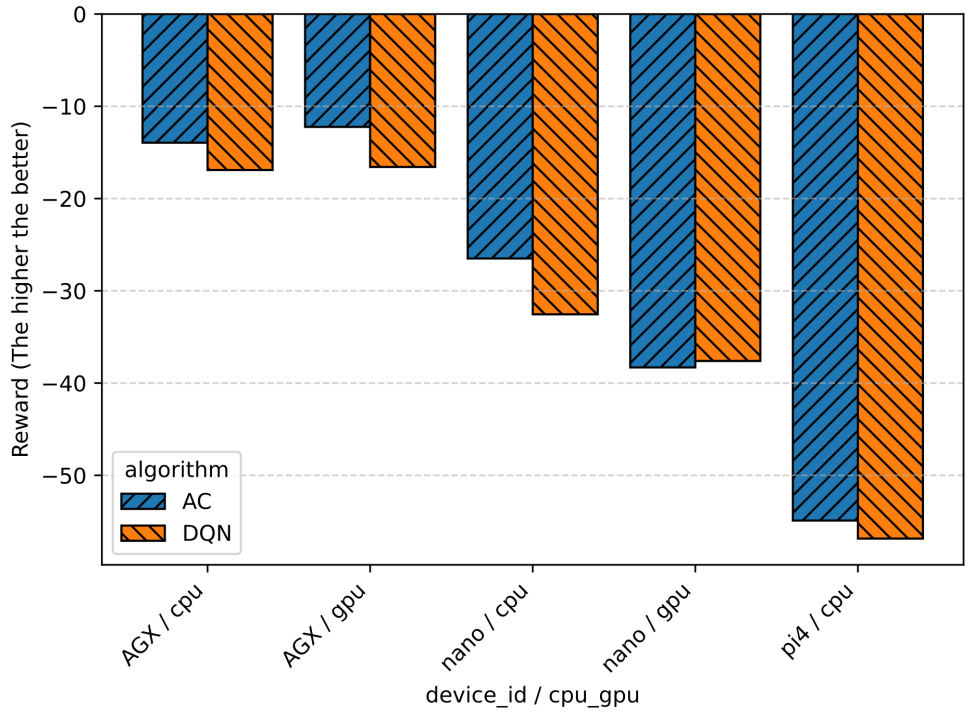


图 7. 场景1中各设备上每种算法的平均奖励

接着，在图 8 中，我们比较了场景1下使用 AC 算法时，在设备本地执行训练与在移动边缘计算服务器上远程执行训练所需的时间。总体而言，由于没有通信开销，本地训练始终更快。在多接入边缘计算上训练时，所有设备的训练时长相对相近，细微差异由网络条件和调制解调器行为造成。需要注意的是，多接入边缘计算训练时间涵盖了整个过程，包括发送模型、执行训练（如绿色柱所示）、接收更新后的参数以及重新组装模型。

图 9 展示了在所有设备上使用 DQN算法时的训练时间，分别是在本地和移动边缘计算服务器上进行训练；观察到的模式与此类似。在

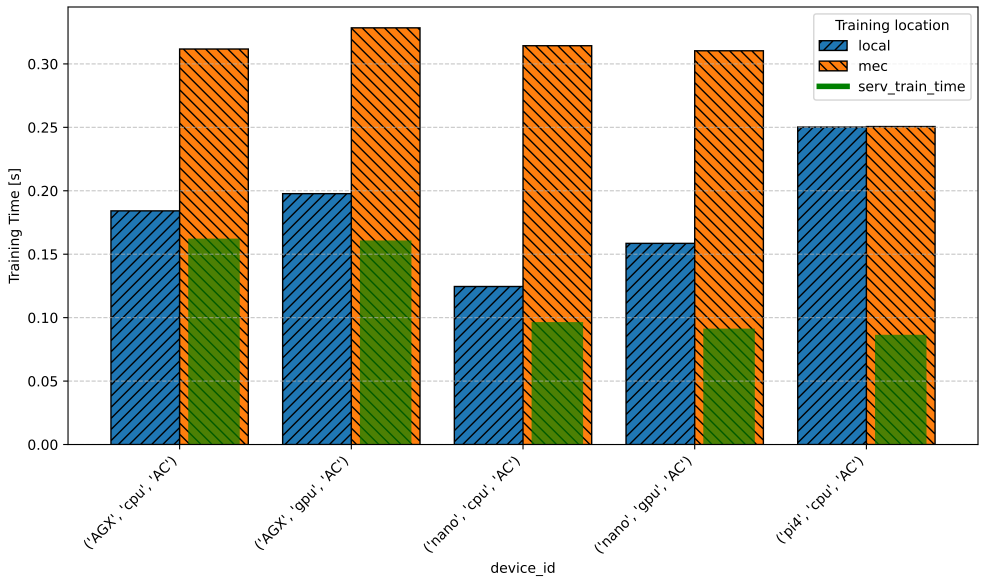


图 8. 场景1中各设备本地和远程执行 AC 训练的平均时间

移动边缘计算服务器上进行训练的时间（包括传输、训练和接收时间）在所有设备上都保持一致，介于 0.25 至 0.35 秒之间。不过，由于 TensorFlow 版本不同，使用 *Jetson AGX Orin*时服务器上的训练过程会稍长一些。相比之下，本地训练速度明显更快，且耗时远少于使用演员-评论家，这反映出 DQN 的计算复杂度较低。该算法的训练计算开销较小，因此本地和远程环境下的训练时间存在显著差异。这些差异凸显了算法复杂度对部署策略的影响。

我们分析了场景1中本地训练及在移动边缘计算服务器上远程训练主动评论家智能体所产生的设备端能耗。由于能耗与训练耗时密切相关，卸载到多接入边缘计算会因额外的通信开销而导致总体功耗更高。这包括等待数据传输和模型更新时的空闲时间。图 10 展示了 *reComputer J1010*、*JetsonAGX Orin* 和 树莓派 的能耗，表明与本地执行相比，远程训练始终会产生更高的能耗。

使用深度 Q 网络时的表现与演员-评论家类似，如图 11 所示。不过，对于深度 Q 网络，本地训练与远程训练时间的差异更加明显，因为其计算复杂度较低，本地训练速度显著更快。

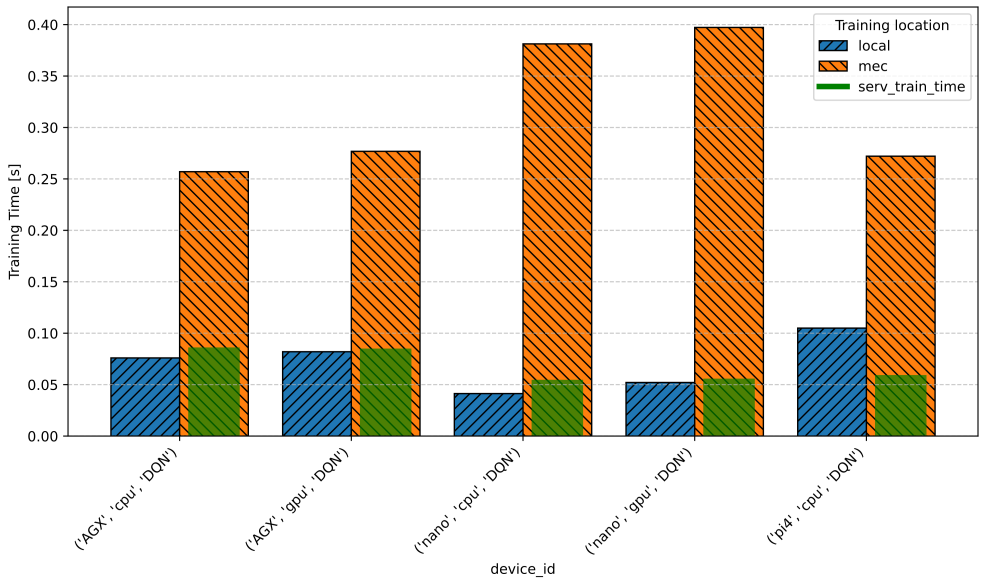


图 9. 场景1中每台设备本地和远程训练深度 Q 网络的平均训练时间

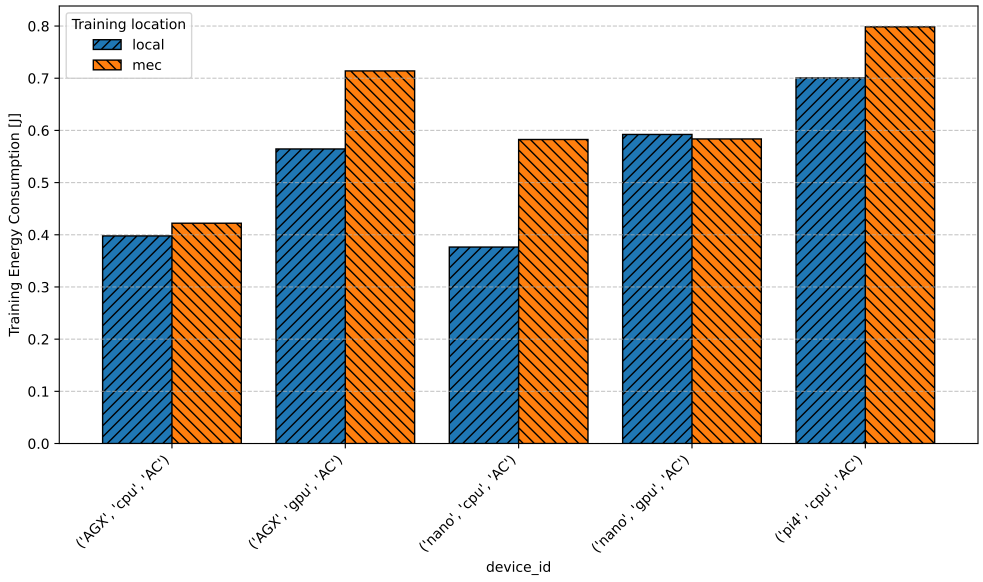


图 10. 场景1中每台设备本地和远程训练演员-评论家的平均能耗

这表明，将深度 Q 网络的训练卸载到多接入边缘计算会引入远高于本地执行的额外开销，因此对于该算法而言，本地训练的效率要高得多。

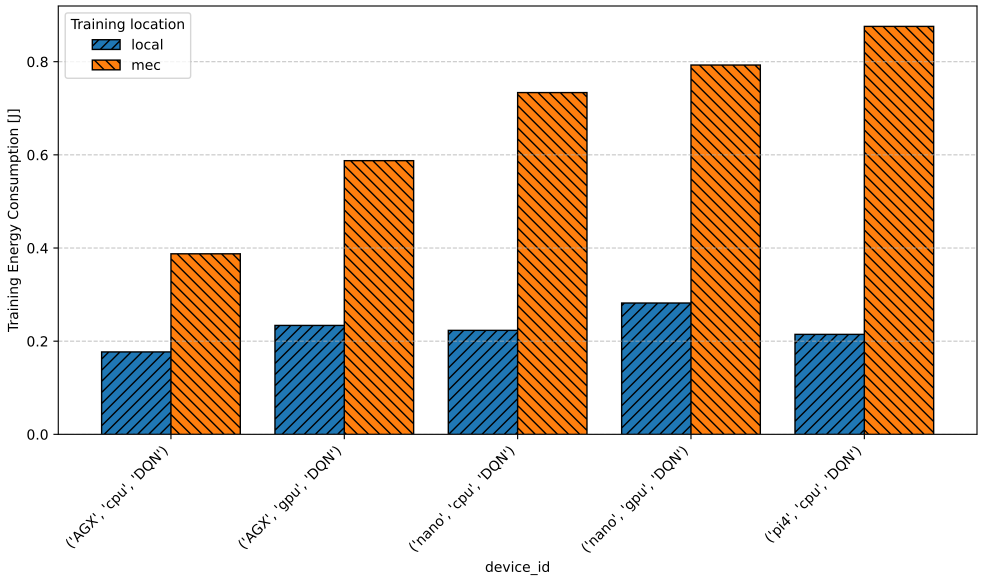


图11. 各设备本地和远程深度 Q 网络训练的平均能耗

最后，总体而言，使用图形处理器时能耗更高，因此如前所述，在此情境下使用图形处理器并不合适。图12展示了在 *Jetson AGX Orin* 上使用演员-评论家和深度 Q 网络，在场景 1 和 2 中训练时的能耗。可以看出，使用图形处理器时，每项测试的能耗都更高，且训练时间也略长于中央处理器执行。因此，对于这些工作负载，使用图形处理器没有任何优势，不建议使用。

随后，我们分析场景 1 和 2 中 AC算法的行为。在场景 2 中，多个并发用户设备的存在会因额外的流量和计算需求，对通信信道和移动边缘计算服务器造成扰动。正如图13所示，这些情况可能会显著降低远程训练性能。在场景 2 中将训练卸载到多接入边缘计算时，与场景1相比，训练时间大幅增加，凸显了网络拥塞和资源争用对学习效率的影响。只有在场景1中使用 树莓派 时，在移动边缘计算服务器上训练才可能与本地训练相媲美。

图14采用DQN算法进行了相同的分析，得出了类似的结果。在这种情况下，不存在使用多接入边缘计算服务器进行训练更为合适的情形

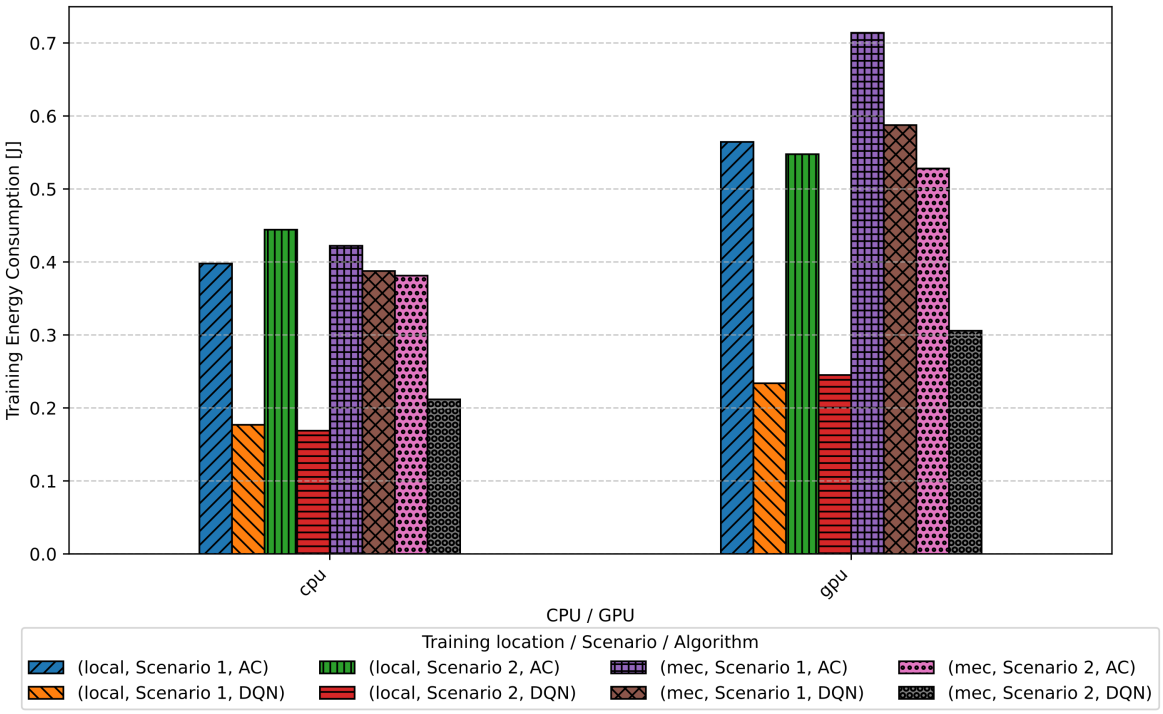


图 12. AGX 在各场景中的能耗

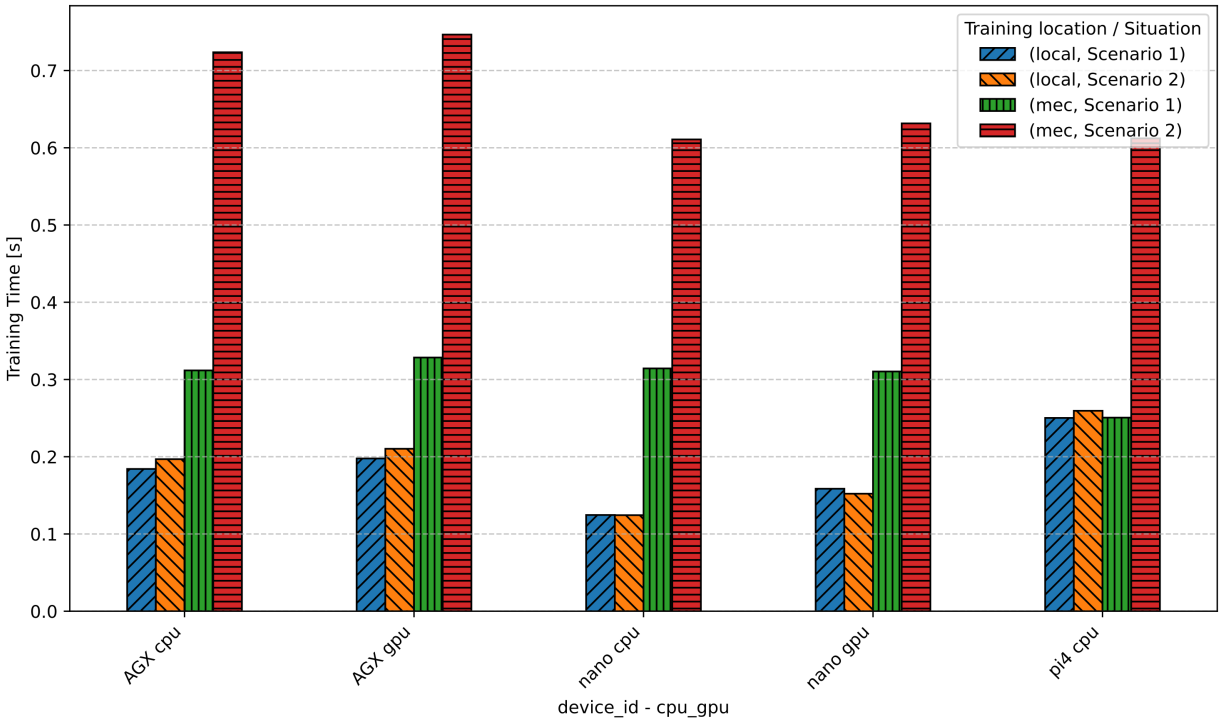


图 13. 场景 1 和 2 中演员-评论家训练时间的比较

与本地训练相比。

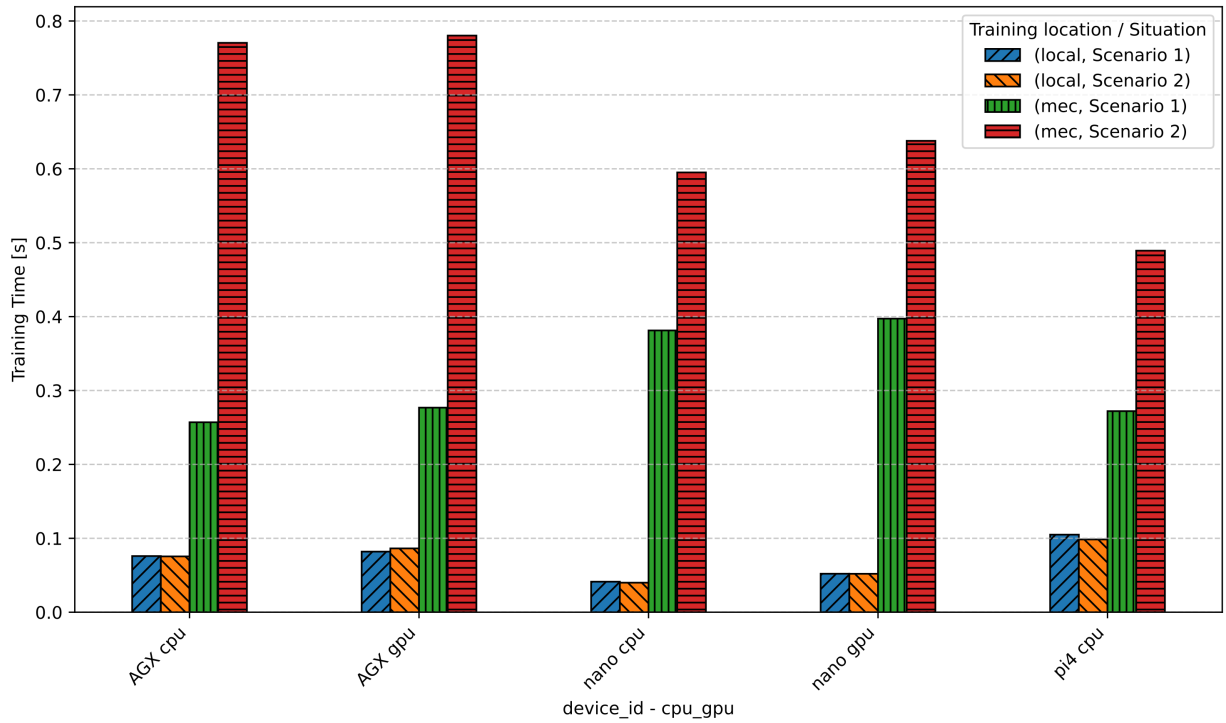


图 14. 使用和不使用多用户设备时深度 Q 网络训练时间的比较

我们分别比较每台设备上的性能，并分析获得的奖励。图 15 展示了在两种场景下，使用 *Jetson AGX Orin*进行本地训练与卸载到多接入边缘计算服务器时，两种算法的性能。鉴于*Jetson AGXOrin*具有更强的计算能力，因此倾向于选择本地执行而非卸载，因为本地训练能够获得更高的奖励，并具有更好的响应能力。这一观察结果进一步说明了避免采用集中式模型或聚合模型的必要性，因为每台设备都具有影响性能的不同特性。在这两种算法中，演员-评论家始终略优于深度 Q 网络，这证实了它更适用于计算能力更强的边缘设备。此外，使用图形处理器获得的结果与使用中央处理器获得的结果相近或更差。

如图16所示，对于*reComputer J1010*，两种算法在场景1中进行本地训练时也都取得了最佳结果。不过，*reComputer J1010* 的计算能力较弱，因此获得的奖励低于使用*Jetson AGX Orin*时的奖励。无论如何，从*Jetson AGX Orin* 得出的结论可以

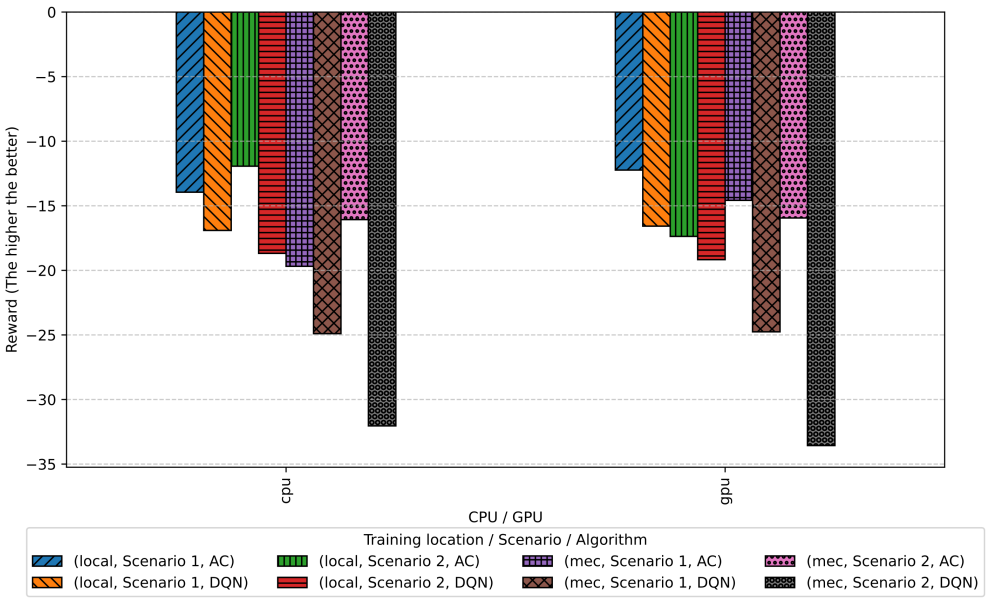


图 15. AGX 在各条件下的奖励

推广到使用*reComputer J1010*时的结果。

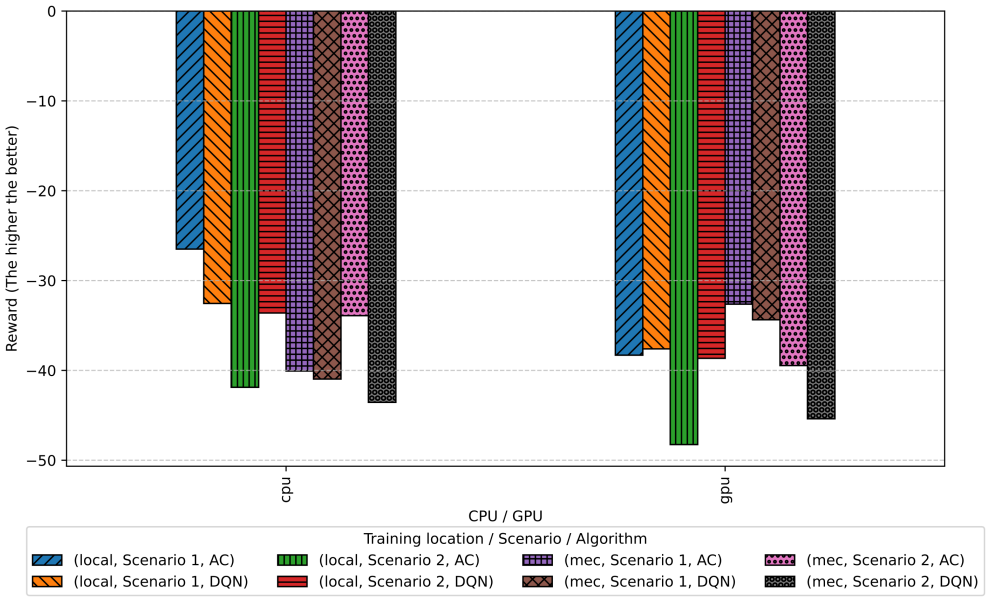


图 16. reComputer J1010 在各条件下的奖励

最后，如图17所示，在评估的设备中，树莓派 获得的奖励最低。在场景1中，AC算法无法选择最优动作，这可能是由于

设备有限的计算资源。

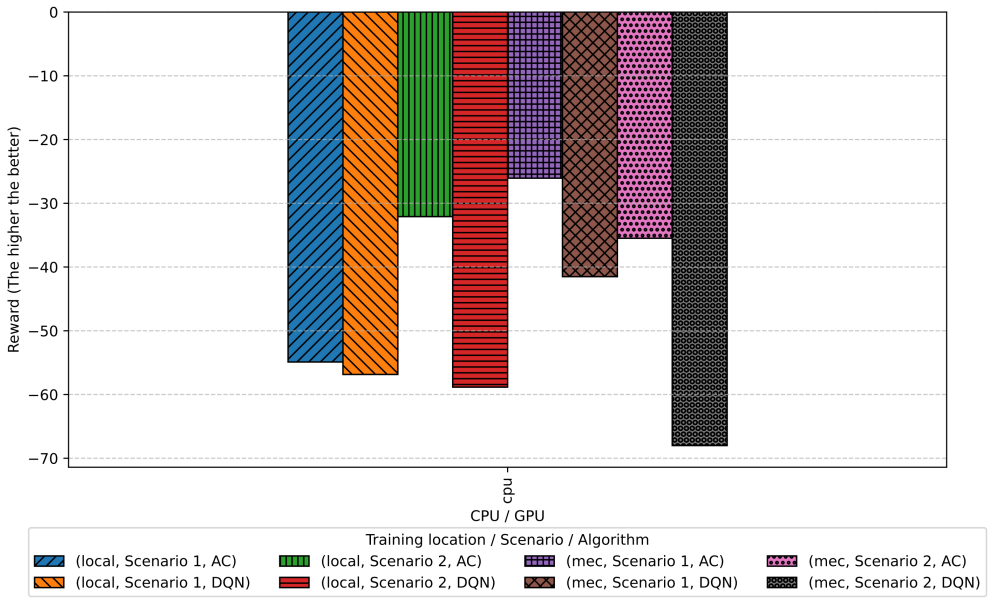


图 17. 树莓派在各条件下的奖励

本节结果表明，所提出的深度强化学习智能体对于边缘环境中的自适应任务卸载既有效又可行。在某些约束条件下，设备端训练是可行的，尤其是采用节能的中央处理器执行，而非图形处理器加速。即使在 Jetson AGX Orin 这样的高端平台上，在不进行批处理的持续学习场景中，使用图形处理器也几乎没有带来任何好处，反而增加了能耗。对于轻量级在线深度强化学习任务，相较于以图形处理器为中心的设计，基于中央处理器的执行或自适应混合架构似乎更为合适。此外，将训练卸载到多接入边缘计算服务器的优势有限。根据环境的变化程度，这样做会带来网络延迟和资源争用方面的风险；这也是采用深度强化学习技术解决计算卸载问题的原因之一。

VI. 结论

本研究为基于深度强化学习的任务卸载在边缘环境中的实际部署提供了若干重要见解。我们展示了在现实场景中运行设备端深度强化学习以解决计算卸载问题的可行性及其优势。我们在设备端实现了不同深度强化学习方法的完整流程，以解决计算

卸载问题，并以完全去中心化的方式展示了其相较于集中式解决方案的优势。

我们采用一个测试平台，可通过改变模拟用户设备的数量及其流量交换，生成各种动态环境，从而改变任务特征和网络条件。我们还通过模拟服务器被不同用户设备或进程使用的情形来改变服务器状态；这也会影响集中式训练，从而凸显设备端学习的必要性。

结果表明，在线设备端深度强化学习训练不仅能够在现代边缘平台上实现，而且比远程训练方法更加节能、响应更快。此外，GPU 加速并不能为边缘侧的轻量级深度强化学习训练带来优势；在某些情况下，由于额外开销，反而会降低性能。

这些发现对物联网人工智能（AIoT）系统的设计具有直接影响。系统设计人员不应默认依赖远程训练或 GPU 加速，而应优先采用在本地训练的轻量级、基于 CPU 的深度强化学习智能体，尤其是在网络条件多变且能源约束严格的场景中。

总体而言，该解决方案具有很强的可行性。所提出的深度强化学习智能体在测试的各类场景中表现出色，证明了其在预期应用中的有效性。此外，本地训练的计算开销相对较低，这表明该方法可以在无需大量资源的情况下部署。这些发现支持了所提方法在现实场景中的实用性。

在未来的工作中，我们将探索更多策略，进一步提升设备端学习和可扩展性。值得关注的方向包括针对资源受限设备优化轻量级深度强化学习模型。我们还将深入分析联邦学习（FL）的应用，因为联邦深度强化学习方法能够在无需集中式聚合的情况下实现协同学习，从而加快深度强化学习模型的学习。在持续学习环境中应对概念漂移，对于长期保持性能也至关重要，因为训练可能成为此类环境中的瓶颈。最后，混合计算策略（例如使用中央处理器进行在线适应，并使用图形处理器定期进行批量整合）能够在动态边缘环境中实现效率与准确性的平衡。

参考文献

[1] A. R. Al-Ali, R. Gupta, I. Zualkernan, and S. K. Das, “物联网技术在大数据管理系统中的作用：综述及智能电网案例研究，” *Pervasive and Mobile Computing*，第100卷，文章编号101905，2024年5月。[2] J. Cook, S. U. Rehman, and M. A. Khan, “5G及后续网络中低功耗物联网设备的安全与隐私：挑战与未来方向，” *IEEE Access*，第11卷，第39 295–39317页，2023年。[3] R. T. Himani and Kumar，物联网云计算。Springer，2020年，第2章，第25–41页。[4] M. I. Khattak, H. Yuan, A. Khan, A. Ahmad, I. Ullah, and M. Ahmed, “面向多样化泛在资源利用的多接入边缘计算（MEC）演进：综述，” *Telecommunication Systems*，第88卷，文章编号71，2025年。[5] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D’Andria, “物联网-边缘-云连续体系统综述：现状、挑战、用例与未决问题，” *Future Internet 2023, Vol. 15, Page 383*，第15卷，文章编号383，2023年11月。[6] H. Xiao, C. Xu, Y. Ma, S. Yang, L. Zhong, and G. M. Muntean, “边缘智能：一种面向依赖型物联网应用的计算任务卸载方案，” *IEEE Transactions on Wireless Communications*，第21卷，第7222–7237页，2022年9月。[7] Z. Zabihi, A. Masoud, E. Moghadam, M. H. Rezvani, Z. Zabihi, A. M. E. Moghadam, and M. H. Rezvani, “计算卸载的强化学习方法：系统性综述，” *ACM Computing Surveys*，第56卷，文章编号17，2023年8月。[8] L. Zheng and L. Tan, “基于动态定价的多用户边缘计算任务卸载去中心化方案，”*Peer-to-Peer NetworkingandApplications*，第18卷，第1–22页，2025年4月。[9] S. Anwar, M. T. Vega, M. S. Pathan, A. Elhanashi, P. Dini, S. Saponara, and Q. Zheng, “微型机器学习的进展：在物联网设备中的应用、局限性及影响，” *Electronics 2024, Vol. 13, Page 3562*，第13卷，文章编号3562，2024年9月。[10] C. Wang, Y. Li, and D. Jin, “移动性辅助机会式计算卸载，” *IEEE CommunicationsLetters*，第18卷，第1779–1782页，2014年10月。[11] J. Mei, Z. Tong, K. Li, L. Zhang, and K. Li, “移动边缘计算中满足时延约束的节能启发式计算卸载，” *IEEE Transactions on Services Computing*，第16卷，第4404–4417页，2023年11月。[12] A. Lakhan, T. M. Groenli, G. Muhammad, and P. Tiwari, “赋能工业信息物理系统的进化元启发式卸载与调度方案，” *IEEE Systems Journal*，第18卷，第826–835页，2024年6月。[13] B. Ling, X. Deng, Y. Huang, J. Zhang, J. S. Gui, and Y. Qian, “多层协同任务卸载优化：平衡本地边缘与云资源之间的竞争与合作，”*Journal of Supercomputing*，第80卷，第26 483–26511页，2024年12月。[14] X. Xu, Q. Jiang, P. Zhang, X. Cao, M. R. Khosravi, L. T. Alex, L. Qi, and W. Dou, “边缘计算中基于模糊神经网络的分布式车联网任务卸载博弈论方法，” *IEEETransactions onFuzzy Systems*，第30卷，第4593–4604页，2022年11月。[15] S. Yu, X. Wang, and R. Langar, “移动边缘计算的计算卸载：一种深度学习方法，” *IEEE*个人、室内和移动无线电通信国际研讨会，*PIMRC*，第2017-October卷，第1–6页，2017年7月。[16] K. Moghaddasi, S. Rajabi, F. S. Gharehchopogh, and A. Ghaffari, “面向物联网高效任务卸载的三层设备到设备-边缘-云计算架构深度强化学习高级算法，”*SustainableComputing:Informatics andSystems*，第43卷，文章编号100992，2024年9月。[17] P. Choppara and S. S. Mangalampalli, “一种在雾计算中结合模糊逻辑与深度强化学习的混合任务调度技术，” *IEEE Access*，第12卷，第176 363–176388页，2024年。[18] K. Bian and R. Priyadarshi, “机器学习优化技术：综述、分类、挑战与未来研究议题，” *Archives of Computational Methods in Engineering*，第31卷，第4209–4233页，2024年9月。

[19] L. Wang、Z. Pan 和 J. Wang，《基于强化学习的制造调度智能优化综述》，复杂系统建模与仿真，第1卷，第257–270页，2021年12月。[20] F. C. Pedrosa、J. C. Nereu 和 J. B. do Val，《控制与状态变化如何加剧不确定性：离散时间下的建模与随机控制》，*Automatica*，第123卷，文章编号109341，2021年1月。[21] J. Vainer 和 J. Kukacka，《霍特林模型中的纳什Q学习智能体：重建均衡》，非线性科学与数值模拟通讯，第99卷，文章编号105805，2021年8月。[22] H. Dong、H. Dong、Z. Ding、S. Zhang 和 T. Chang，深度强化学习。Springer，2020年，第24卷。[23] L. W. Shao、L. P. Qian、M. Q. Li、W. Jiang 和 W. Jia，《Sc-drl：一种由状态校正赋能的依赖感知应用卸载深度强化学习算法》，*IEEE Transactions on ServicesComputing*，第1–14页，2025年。[24] M. Hevesli、A. M. Seid、A. Erbad 和 M. Abdallah，《分层MEC赋能空地网络中的队列感知任务卸载多智能体深度强化学习》，*IEEE*认知通信与网络汇刊，2025年。[25] N. Provatas、I. Konstantinou 和 N. Koziris，《参数服务器架构综述：优化分布式集中式学习的方法》，*IEEE Access*，第13卷，第30 993–31015页，2025年。[26] X. Qiu、W. Zhang、W. Chen 和 Z. Zheng，《面向计算卸载的分布式协同深度强化学习：实践视角》，*IEEE*并行与分布式系统汇刊，第32卷，第1085–1101页，2021年5月。[27] S. Zhu、T. Voigt、F. Rahimian 和 J. Ko，《设备端训练：现有系统初览》，*ACM*传感器网络汇刊，第20卷，文章编号39，2024年10月。[28] M. F. A. Sayeedi、J. F. Deepti、A. M. I. M. Osmani、T. Rahman、S. S. Islam 和 M. M. Islam，《物联网设备机器学习模型部署优化的比较分析》，*Applied Sciences*（瑞士），第14卷，2024年。[29] G. Nieto、I. de la Iglesia、U. Lopez-Novoa 和 C. Perfecto，《面向5G边缘云连续体中动态任务卸载的深度强化学习技术》，云计算期刊，第13卷，第1–24页，2024年12月。[30] A. Noureddine，《Powerjoular和Joularjx：多平台软件功耗监控工具》，*2022*年第*18*届智能环境国际会议（*IE 2022*）论文集，2022年。[31] Y. D. Bock、S. Altmeyer、T. Huybrechts、J. Broeckhove 和 P. Hellinckx，《用于基于TACLeBench基准测试套件进行可调度性分析的任务集生成器》，*ACM SIGBED*评论，第15卷，第22–28页，2018年3月。[32] Y. Zhang、X. Chen、Y. Chen、Z. Li 和 J. Huang，《边缘计算系统中时延敏感任务的高成本效益调度》，论文集——*2018*年*IEEE*服务计算国际会议（*SCC 2018*），*2018*年*IEEE*服务世界大会组成部分，第73–80页，2018年9月。