软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007676] [CSTR: 32375.14.jos.007676]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

边缘智能中低成本的快速自适应模型部署与更新方法*

滕自怡, 方 娟, 刘雅祺, 张梦媛, 陈慧杰

(北京工业大学 计算机学院, 北京 100124)

通信作者: 方娟, E-mail: fangjuan@bjut.edu.cn

摘 要: 边缘智能通过在网络边缘部署模型, 能有效降低数据传输延迟并缓解中心云的计算压力. 然而, 模型在边缘侧的部署与持续更新, 使部署效率与成本成为评估系统性能的关键指标. 尽管已有研究提出多种策略以权衡模型精度与部署成本, 但在服务请求多变且资源受限的实际环境中, 现有方法仍难以实现高效、自适应的部署与更新. 针对这一问题, 提出一种面向边缘智能的低成本快速自适应模型部署与更新策略 PACE, 旨在动态环境中以更低的部署成本实现对服务需求的高效响应, 从而提升边缘智能系统的持续服务能力与性能. 具体而言, 该方法通过挖掘模型间的共享特性, 对不同阶段的部署成本进行精细化建模, 并结合模型信息年龄 (age of information, AOI) 构建性能评估模型, 从而实现更高成本效益的部署决策. 在此基础上, 引入元强化学习以学习任务变化的先验知识, 并结合变分贝叶斯网络实现在线优化, 从而增强模型部署策略在动态环境下的快速自适应能力. 最后, 从泛化性与收敛性两个方面验证 PACE 的理论有效性. 基于合成的 Zipf 请求分布对所提出的 PACE 方法进行评估. 实验结果表明, PACE 在任务频繁变化的场景中能够实现高效的自适应部署, 在显著降低模型部署成本的同时, 仍可维持可接受的模型精度与系统响应效率.

关键词: 边缘智能; 模型部署; 模型更新; 任务自适应; 部署成本

中图法分类号: TP309

中文引用格式: 滕自怡, 方娟, 刘雅祺, 张梦媛, 陈慧杰. 边缘智能中低成本的快速自适应模型部署与更新方法. 软件学报. http://www.jos.org.cn/1000-9825/7676.htm

英文引用格式: Teng ZY, Fang J, Liu YQ, Zhang MY, Chen HJ. Low-cost and Rapidly Adaptive Model Deployment and Update Method for Edge Intelligence. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7676.htm

Low-cost and Rapidly Adaptive Model Deployment and Update Method for Edge Intelligence

TENG Zi-Yi, FANG Juan, LIU Ya-Qi, ZHANG Meng-Yuan, CHEN Hui-Jie

(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract: Edge intelligence places models at the network edge, effectively reducing data transmission latency and alleviating the computational load on the central cloud. However, the deployment and continuous updating of models at the edge make deployment efficiency and cost key criteria for evaluating system performance. Although prior work has proposed various strategies to balance model accuracy and deployment costs, in practical settings with volatile service demands and constrained resources, existing methods still struggle to achieve efficient and adaptive deployment and updates. To address this challenge, this study proposes PACE, a low-cost and rapidly adaptive model deployment and update strategy for edge intelligence. PACE aims to achieve efficient responses to dynamic service demands with lower deployment costs, thus enhancing the sustained service capability and performance of edge intelligence systems. Specifically, shared characteristics among models are exploited to perform fine-grained modeling of deployment costs at different stages, and the age of information (AOI) is incorporated to construct a performance evaluation model, thus enabling more cost-effective deployment decisions. On this basis, meta-reinforcement learning is introduced to learn prior knowledge of task variations, and a

* 基金项目: 国家自然科学基金 (62202019); 北京市自然科学基金 (4262021)
收稿时间: 2025-10-21; 修改时间: 2025-12-19, 2026-03-03; 采用时间: 2026-03-23; jos 在线出版时间: 2026-07-08

variational Bayesian network is combined to achieve online optimization, thus enhancing the rapid adaptability of the deployment strategy in dynamic environments. Finally, the effectiveness of PACE is theoretically validated from the perspectives of generalization and convergence. The evaluation is conducted using a synthetic Zipf request distribution. Experimental results show that, in scenarios with frequently changing tasks, PACE achieves efficient adaptive deployment, significantly reducing model deployment costs while maintaining acceptable model accuracy and system response efficiency.

Key words: edge intelligence; model deployment; model update; task adaptation; deployment cost

随着人工智能 (artificial intelligence, AI) 在计算机视觉、自然语言处理、智能驾驶和医疗诊断等终端应用中广泛发展, 这些应用通常对低延迟和高性能具有严格需求[1]. 边缘智能通过将 AI 应用的数据处理和模型推理服务从云中心迁移至靠近终端设备的边缘侧, 有效降低服务处理时延并增强数据隐私保护[2]. 模型精度作为智能应用服务响应的重要指标, 需要通过在线更新或从云端获取最新模型来保持性能. 模型精度下降的主要原因包括: 模型漂移 (例如边缘侧模型压缩导致的精度下降[3])、数据漂移 (特征或标签分布随时间变化[4]) 以及任务漂移 (模型在微调或迁移任务中的精度变化[5]). 在边缘设备资源受限的场景下, 这些漂移现象尤为显著, 容易导致模型精度下降. 因此, 在边缘智能环境中, 联合优化模型的部署与更新策略是保障 AI 应用低延迟和高性能的关键.

尽管通过模型间参数块共享可以降低模型部署的成本, 但在动态环境下以及从联合部署和更新的角度来看, 仍存在 3 个亟须解决的挑战: (1) 存储资源容量与缓存需求存在矛盾. 异构模型在大小与参数量上差异显著, 参数共享只能缓解部分存储资源争用, 大规模模型库中各模型的特有部分仍占用大量存储空间. 因此, 在容量受限条件下如何持续满足动态任务需求仍具挑战; (2) 模型更新成本与响应效率存在矛盾. 模型更新需要从云端获取最新权重并产生更新成本, 若更新不及时则可能降低边缘侧响应效率, 因此需要在动态任务下对更新开销与服务质量进行合理权衡; (3) 非平稳请求下的联合部署与更新难度较高. 系统需要快速适应新请求并保持在线推理效率, 而现有方法适应性不足, 难以在有限计算与更新预算下快速获得接近最优的联合策略, 从而可能引发性能波动. 基于此, 如何在动态请求下实现高效的联合部署与更新并保持整体性能稳定仍是核心问题.

为解决上述问题, 我们提出面向边缘智能的低成本快速自适应模型部署与更新策略 PACE. 首先, PACE 基于模型参数块共享刻画异构模型在容量约束下的缓存与调用过程, 并通过对部署决策变化施加平滑约束抑制频繁重配置, 从而提升有限资源条件下对动态任务需求的满足能力. 其次, PACE 引入信息年龄刻画模型新鲜度, 将更新成本与精度退化显式建模, 并与切换、通信与推理等成本共同构成统一的部署成本目标, 以刻画更新开销与服务质量之间的权衡关系. 最后, PACE 在同一框架内协同建模部署与更新决策, 通过离线元强化学习从历史非平稳请求中学习快速适应的初始化, 并在在线阶段利用变分贝叶斯推断实现联合优化, 从而增强系统在动态环境下的自适应能力并保持整体性能稳定. 综上所述, 本文的主要贡献如下.

(1) 针对边缘智能环境中多变的服务请求与资源约束, 分析模型部署过程中不同成本的影响, 包括模型切换、更新、精度和通信成本, 结果表明切换成本和精度成本是最主要的影响因素.

(2) 基于参数块共享, 对切换、通信、推理、更新与精度退化等成本进行统一建模, 提出联合部署与更新的优化问题, 以最小化部署成本.

(3) 提出快速适应的联合模型部署与更新算法 PACE. 该算法通过连续化处理缓解时间耦合带来的求解困难, 并结合元强化学习从历史非平稳请求中学习快速适应的初始化; 在线阶段利用变分贝叶斯推断实现联合部署与更新优化, 从而增强系统在动态环境下的自适应能力.

(4) 从泛化性与收敛性两个方面验证 PACE 的理论有效性, 并在非平稳的 Zipf 请求分布下对算法性能进行评估. 结果表明, 与其他基准算法相比, PACE 能够在显著降低部署成本的同时, 维持可接受的模型精度与系统响应效率, 从而体现更优的成本与服务质量折中.

本文第 1 节介绍研究动机. 第 2 节介绍相关工作并分析局限性. 第 3 节详细介绍系统模型与成本建模. 在第 4 节中介绍问题建模与转化. 第 5 节介绍所提出的低成本快速自适应模型部署与更新策略. 第 6 节介绍实验与评估方法. 第 7 节总结局限性与未来工作.

1 研究动机

在边缘网络中, 模型任务的服务请求呈现高度动态性, 不同区域的任务流行度存在显著差异. 例如, 城市中心在早晚通勤时段会出现实时交通分析任务高峰[6], 而住宅区域在夜间则集中出现智能家居视频分析任务请求[7]. 这种动态特性要求模型部署策略具备较强的适应能力, 能够随着任务分布的变化迅速调整. 当前模型部署研究中, 决策策略主要采用在线算法 (如 LRU 缓存策略[8]) 或启发式算法 (如贪婪算法[9]), 并常结合机器学习方法进行优化. 我们在 Zipf 合成的多任务变化数据集上评估了 3 种代表性的在线和启发式算法的部署成本, 包括切换成本、推理成本、精度成本、更新成本和通信成本 (数据集和部署成本的详细介绍见第 4.1 节和第 2.3 节), 其中部署算法包括贪婪算法 (Greedy)、LRU 算法和随机算法 (Random). 不同部署策略中各类成本在总成本中所占比例如图 1 所示.

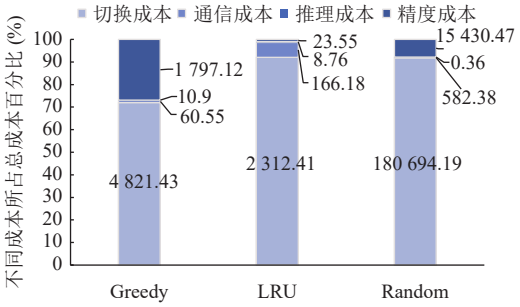


图 1 不同部署策略中成本占总成本的百分比

从图 1 可以看出, 无论采用传统在线算法还是启发式算法, 切换成本在总成本中均占比最大, 其次是精度成本, 而通信成本相对较低. 这主要是因为在动态环境下, 这类算法难以及时适应模型请求的变化, 往往需要频繁将所需模型加载到边缘设备有限的 GPU 显存中, 从而产生较大的切换成本. 另一方面, 由于边缘设备的计算和存储能力有限, 若模型长期不更新, 其精度会逐渐下降, 难以满足服务质量要求, 从而带来精度成本. 值得注意的是, 相比切换成本与精度成本, 通信成本较小, 因此可以通过将更新后的模型从云端拉取至边缘侧来保证模型服务精度, 同时降低精度成本. 由此可见, 在动态环境下, 如何快速适应模型请求以降低切换成本并保证模型精度, 是优化边缘智能服务过程中的关键.

值得注意的是, 模型的切换成本和更新成本与其规模密切相关. 模型的参数量和架构越大, 在边缘设备上加载与切换时占用的显存越多, 从而导致延迟和资源压力也越显著. 随着深度神经网络层数和参数量的持续增长, 这一问题愈发突出[10]. 在模型训练中, 迁移学习、多任务学习和参数高效微调方法 (如 LoRA[11]) 通常依赖于预训练模型中可共享的底层参数, 这些参数一般保持冻结, 仅需针对下游任务微调少量新增参数, 规模约为原始参数的 1%–10%[12,13]. 这一机制表明, 模型之间存在广泛的参数共享特性. 基于此, 我们进一步分析了同一变体或不同模型间的架构相似性, 重点考察 3 类模型的参数共享潜力, 主要包括同一模型的多个变体实例、同一簇内的不同模型 (如不同规模的 ResNet 变体), 以及不同模型家族之间的差异. 图 2 展示了这些模型对的代表性结果.

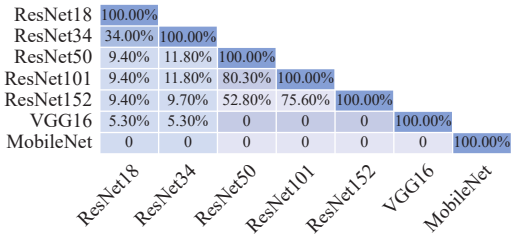


图 2 不同模型对的结构相同层的百分比

图 2 的结果显示, 同一家族中的不同模型架构相似度最高可达 80.3%, 而不同家族之间也存在一定程度的参数共享, 其共享比例为 5.3%. 这种高共享性主要源于结构继承关系, 即较大变体通常是基础模型的扩展版本. 例如, ResNet 系列模型共享由重复模块 (每组包含 2–3 个卷积层) 构成的网络结构, 其中第 1 个卷积层和最后的全连接层在所有变体中保持一致[14]. 受此启发, 在边缘侧部署时可以通过共享相同的参数块来降低内存占用, 仅保留一份权重副本, 从而在模型执行过程中减少所需内存并降低切换和更新成本. 上述现象共同表明, 动态请求导致的高切换成本与模型精度维护需求是主要瓶颈, 而模型间的参数共享为降低部署与更新开销提供了可利用的结构性机会, 这促使我们进一步研究面向动态环境的联合部署与更新策略.

2 相关工作

尽管边缘智能中的模型部署问题已从多个角度得到广泛研究, 但在动态边缘网络和多变请求特性条件下, 如何结合模型特征实现资源节约, 并协同优化模型的部署与更新策略, 仍有待深入探讨. 现有研究主要围绕模型部署与更新机制, 以及动态环境下的在线学习与服务缓存优化两个方向展开. 下面将对相关研究进行回顾, 并进一步分析现有工作的局限性, 以引出本文的研究内容.

2.1 模型部署与更新机制

在边缘智能系统中, 模型部署策略直接影响推理时延、资源利用率以及服务质量, 因此已有研究围绕异构部署与资源协同开展了大量探索. 文献 [15] 面向异构推理服务环境, 联合考虑 DNN 模型部署、模型选择与配置调整, 在模型精度、推理时延和资源开销之间实现权衡. 文献 [16] 进一步研究协同边缘推理场景下模型部署与模型划分的联合优化问题, 通过动态调整模型部署位置与划分策略, 在满足隐私约束条件下降低推理时延并提高资源利用效率. 文献 [17] 则从移动边缘智能网络角度出发, 将模型缓存与应用卸载纳入统一优化框架, 实现模型缓存、任务卸载和资源分配之间的协同优化. 上述研究表明, 边缘模型服务已由静态放置逐步转向与任务分配和资源调度协同的部署优化.

随着边缘侧模型规模不断增大, 模型共享与参数复用开始成为降低部署成本的重要途径. 文献 [18] 面向移动边缘计算环境中的大模型服务, 研究了基于参数共享的联合模型选择与参数拉取问题, 通过利用微调的基础模型之间的共享结构来降低模型拉取时延与资源消耗. 除此之外, 已有研究也从参数共享缓存和模型合并等角度探索模型复用机制, 以降低重复存储与切换代价[12,13]. 这表明, 利用模型间共享结构开展细粒度管理, 已成为提升边缘模型服务效率的重要思路.

另一方面, 随着任务分布和数据分布持续变化, 模型更新与性能维护问题也逐渐受到关注. 文献 [19] 研究了数字孪生辅助边缘计算场景下模型重训练与推理服务的联合优化, 指出训练数据漂移会导致推理精度随时间衰减, 因此需要统筹平衡重训练收益与推理服务开销. 文献 [20] 则从数据新鲜度角度出发, 研究云边网络中的缓存替换与内容更新问题, 以缓解数据陈旧带来的性能下降. 总体来看, 现有研究已在异构部署、参数共享、模型缓存、重训练协同以及新鲜度维护等方面取得了重要进展, 但多数方法仍主要针对部署、共享或更新中的单一侧面展开优化, 尚未充分刻画部署状态变化、更新成本以及模型性能演化之间的动态耦合关系.

2.2 动态环境下的在线学习与服务优化

在动态环境中, 请求分布和系统状态通常具有明显的非平稳特性, 因此研究者开始从在线学习角度设计自适应缓存与服务决策方法, 以提升系统对未知环境变化的适应能力. 文献 [21] 提出基于推荐信息的乐观在线学习缓存方法, 在预测较为准确时能够获得更优性能, 在预测误差较大时仍保持稳定的性能保证. 文献 [22] 进一步研究个性化需求动态变化条件下的边缘缓存问题, 从用户需求随时间变化的角度刻画其对缓存决策的影响. 文献 [23] 则从动态内容的主动推送与按需获取机制出发, 将内容时变性和更新代价显式纳入边缘缓存分析. 文献 [24] 进一步研究未知缓存未命中代价条件下的遗憾最小化在线学习方法, 用于降低边缘缓存服务的长期运行成本. 上述工作从预测辅助决策、需求动态变化、内容更新机制以及遗憾最小化学习等角度, 为非平稳环境下的在线缓存优化提

供了较为系统的理论基础.

在此基础上, 近期研究进一步将在线学习方法扩展到边缘服务缓存与服务部署场景. 文献 [25] 针对动态环境中的服务缓存问题提出近最优在线学习方法, 并在真实边缘网络环境中验证了算法的鲁棒性. 文献 [26] 将动态遗憾分析引入边缘服务缓存问题, 用于刻画时变请求条件下在线决策的性能表现. 文献 [27] 将移动边缘计算中的服务缓存问题建模为受约束的多臂赌博机优化问题, 在优化用户感知时延的同时控制任务在边缘侧的处理比例. 文献 [28] 进一步面向突发业务流量和不确定处理时延条件下的移动边缘计算网络, 研究动态服务缓存与任务卸载问题, 并利用学习驱动方法提升系统在复杂时变环境中的适应能力. 总体来看, 在线学习方法已逐步由一般网络缓存扩展到边缘服务缓存场景, 并在适应时变请求、降低长期决策损失以及增强环境鲁棒性等方面展现出较好效果.

综合来看, 现有研究已在模型部署、共享复用、持续更新以及在线自适应等方面取得了重要进展, 但仍存在两方面不足. 一方面, 模型部署与更新通常被分阶段处理, 难以刻画二者在资源占用、性能演化与服务响应方面的动态耦合关系. 另一方面, 在线学习类方法虽然能够适应非平稳环境, 但其研究对象主要仍是内容缓存或服务缓存, 难以直接支持模型级对象的共享管理与持续更新优化. 与上述研究不同, 本文关注非平稳请求环境下模型部署与更新的联合动态优化问题, 通过利用模型之间的共享特性实现细粒度缓存管理, 并引入信息年龄刻画模型新鲜度对系统性能的影响, 在统一优化框架下协同考虑模型部署、更新成本、通信开销以及推理性能, 从而实现动态环境中的高效模型部署与更新策略.

3 系统模型与成本建模

3.1 系统模型

本文所考虑的边缘协同模型部署与更新系统框架如图 3 所示. 图中包括负责托管人工智能模型的云服务数据中心、负责部署和更新模型子集的边缘服务器, 以及发起服务请求的智能终端设备. 当边缘服务器收到终端设备请求时, 若所请求的模型已部署, 则直接进行推理并返回结果; 否则, 边缘系统将启动部署选择流程, 决策模型的部署与更新, 随后再响应服务请求. 边缘服务器集合表示为 $\mathcal{B} := \{1,2,\ldots,b,\ldots,B\}$, 终端设备集合表示为 $\mathcal{M} := \{1,2,\ldots,m,\ldots,M\}$. 云服务计算中心维护一个注册模型库, 其中所有模型服务的集合表示为 $\mathcal{I} := \{1,2,\ldots,i,\ldots,I\}$. 本文定义边缘服务器集合为 $\mathcal{B}$, 请求与参数块级部署、更新决策均以服务器索引 b 表示. 本文默认终端与边缘服务器的关联已由上层机制确定, 不再额外建模服务器间的负载均衡与请求迁移. 若需支持跨节点模型共享与协同更新, 可在成本模型中进一步加入边缘节点间的传输开销与协同约束进行扩展. 为便于建模与分析, 本文采用离散时隙进行系统演化与决策更新, 并假设所有边缘服务器具有相同的计算与存储能力. 图 3 为云边协同的模型部署与更新框架示意图.

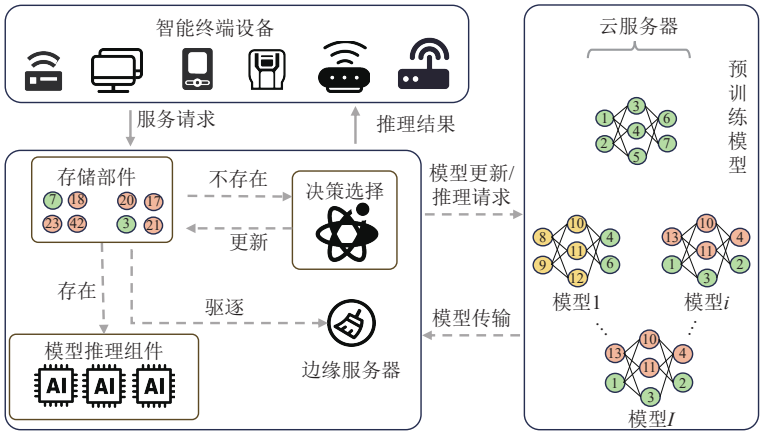


图 3 云边协同的模型部署和更新框架

模型库: 为应对边缘设备的容量限制并利用模型间的共享特性[12,13,18,19], 本文采用参数块共享的模型库架构. 具体而言, 模型库中的每个模型由若干参数块组成, 包括共享模块与特定模块. 共享模块源自预训练模型, 可在多个模型间复用, 承担基础特征提取与通用表示的功能; 特定模块用于模型的个性化任务, 仅属于单个模型, 用以捕捉特定任务所需的信息. 基于上述定义, 模型服务集合为$\mathcal{I}$. 令参数块索引集合为$\mathcal{J}=\{1,2,\ldots,j,\ldots,J\}$. 每个模型$i\in\mathcal{I}$由若干参数块组成, 其参数块集合记为$\mathcal{J}_i\in\mathcal{J}$. 对任意参数块$j\in\mathcal{J}$, 定义包含该参数块的模型集合为$\mathcal{I}_j=\{i\in\mathcal{I}\mid j\in\mathcal{J}_i\}$. 若$|\mathcal{I}_j|=1$, 则$j$为特定参数块; 若$|\mathcal{I}_j|>1$, 则$j$为共享参数块. 考虑边缘服务器$b$的存储约束, 设其容量为$C_b$, 参数块$j$的大小为$c_j>0$. 若$\mathcal{J}_b\subseteq\mathcal{J}$表示服务器$b$实际缓存的参数块集合, 则应满足容量约束$\sum_{j\in\mathcal{J}_b}c_j\leqslant C_b$.

模型服务请求: 为了表示系统的时变性, 我们将观测区间离散化为时隙集合$\mathcal{T}=\{1,2,\ldots,t,\ldots,T\}$. 在每个时隙$t\in\mathcal{T}$内, 仅包含一个请求, 记为$r_{b,i}^t$, 表示在服务器$b$上对模型$i$的请求. 这一单请求假设用于简化离散时间建模与仿真. 对于高并发场景, 可将时隙t视为决策更新窗口, 在窗口内聚合多个到达请求, 并用$\bar{r}_{b,i}^t$表示窗口内对模型i的请求计数或归一化频率, 从而不必对每个请求单独离散化, 相关成本可按窗口内请求量累加或加权计算. 本文将全部请求按到达顺序划分为N个块, 每个块包含R个请求. 记第n个块的任务为$\mathcal{Q}_n$, 并用$\mathcal{P}_{\mathcal{Q}_n}$表示其数据分布, 即$\mathcal{Q}_n\sim\mathcal{P}(\mathcal{Q})$. 全部任务组成任务集合$S=\{\mathcal{Q}_1,\mathcal{Q}_2,\ldots,\mathcal{Q}_N\}$. 这种任务的定义方式常用于元强化学习, 旨在捕捉不同任务之间的相关性, 从而使系统能够在分布变化时快速适应不同的服务请求[29].

3.2 模型部署和更新机制

为满足延迟敏感型服务的实时推理需求, 需要将模型参数合理部署至边缘服务器. 由于模型间存在参数共享特性, 我们以参数块为最小部署与更新单位进行建模.

模型部署决策: 参数块级的模型部署决策定义为$X=\{x_{b,j}^t\in\{0,1\}\mid t\in\mathcal{T},b\in\mathcal{B},j\in\mathcal{J}\}$, 其中$x_{b,j}^t=1$表示在时隙$t$参数块$j$已部署在边缘服务器$b$上, 否则$x_{b,j}^t=0$. 由于模型$i$由参数块集合$\mathcal{J}_i\subseteq\mathcal{J}$构成, 若在时隙$t$时满足$\forall j\in\mathcal{J}_i$, 均有$x_{b,j}^t=1$, 则模型$i$可在节点$b$完成推理; 否则需从云端获取缺失参数块$\mathcal{J}_{b,i}^{\text{miss}}(t)$, 其中$\mathcal{J}_{b,i}^{\text{miss}}(t)=\{j\in\mathcal{J}_i\mid x_{b,j}^t=0\}$.

信息年龄 (AOI): 由于模型会随时间推移而性能下降, 引入信息年龄 (age of information, AOI) 来表示部署在边缘节点的模型参数与云服务器最新模型参数之间差距[30]. 定义在节点b上参数块j的 AOI 为$a_{b,j}^t$. 其随时间的演化为:

$$a_{b,j}^{t+1}=\begin{cases}0, & \text{若在时隙}t\text{对块}j\text{执行同步或更新}\\ \min\{a_{b,j}^t+1,a_{\max}\}, & \text{若}x_{b,j}^t=1\text{且在时隙}t\text{未更新}\\ \text{未定义}, & \text{若}x_{b,j}^t=0\text{ (未部署, AOI无意义)}\end{cases}\tag{1}$$

其中, $a_{\max}>0$为可容忍的 AOI 上限. 若块j被多个模型共享, 则同一个节点b上的$a_{b,j}^t$是全局唯一的, 任一模型触发的更新都会将该块 AOI 置为 0. 考虑到推理依赖于模型的完整参数集合, 为简化建模, 本文采用最旧块近似模型 AOI. 具体地, 当模型i在节点b上可完成本地推理时, 其 AOI 定义为$a_{b,i}^t=\max_{j\in\mathcal{J}_i\cap\{j\mid x_{b,j}^t=1\}}a_{b,j}^t$, 即以模型$i$在节点$b$上所有已部署参数块中最旧者的 AOI 作为模型 AOI. 若$\mathcal{J}_i$中部分块与其他模型共享, 且这些共享块被更新, 则对应块级 AOI 被重置为 0, 从而降低整体$a_{b,i}^t$.

基于 AOI 的精度衰减: 考虑数据漂移与概念漂移导致的性能退化, 受文献 [20,21] 的启发, 构建渐进式的 AOI 驱动精度衰减模型:

$$A_{b,i}^t=A_{b,i}^{t-1}\mathrm{e}^{\frac{-\lambda a_{b,i}^t}{a_{\max}}}\tag{2}$$

其中, $A_{b,i}^t\in[0,1]$表示时隙t模型i在节点b的精度, $\lambda>0$为衰减系数. AOI 越大, $\mathrm{e}^{\frac{-\lambda a_{b,i}^t}{a_{\max}}}$越小, 精度下降越明显.

服务更新: 当云端接收到新的数据时, 会对模型进行更新以保持其性能. 边缘服务器可主动向云端拉取块级更新以保持新鲜度. 定义的块级更新决策张量:

$$Y=\{y_{b,j}^t\in\{0,1\}\mid b\in B,j\in\mathcal{J},t\in\mathcal{T}\}\tag{3}$$

其中, $y_{b,j}^t=1$表示在时隙t对节点b上的参数块j执行更新; 否则不更新. 为简化更新过程并保证决策可行性, 本文约束更新操作仅针对已部署的参数块进行, 即满足$y_{b,j}^t\leqslant x_{b,j}^t$. 在更新决策作用下, AOI 的演化与公式 (1) 一致, 可

写为:

$$a_{b,j}^{t+1}=\begin{cases}0, & x_{b,j}^{t}=1,\ y_{b,j}^{t}=1\\ \min(a_{b,j}^{t}+1,a_{\max}), & x_{b,j}^{t}=1,\ y_{b,j}^{t}=0\\ \text{未定义}, & x_{b,j}^{t}=0\end{cases} \tag{4}$$

3.3 成本模型

基于前述模型部署与更新决策定义, 模型的推理服务可在边缘侧本地执行, 也可通过回程链路在云端执行. 具体的执行方式依赖于模型的部署状态与更新决策. 系统在部署、更新与执行过程中会产生不同类型的成本, 包括切换成本、通信成本、更新成本、精度成本与推理成本. 下面对各部分进行定义.

切换成本: 切换成本表示在边缘服务器上部署或移除参数块时所产生的瞬时资源消耗 (如带宽、能耗、传输延迟等). 该成本仅在部署状态发生变化时产生, 切换成本定义为:

$$P_{\text{switch}}^{t}=\sum\nolimits_{b\in B}\sum\nolimits_{j\in\mathcal{J}}[x_{b,j}^{t}-x_{b,j}^{t-1}]^{+}\cdot C_{b,j}^{\text{switch}} \tag{5}$$

其中, $C_{b,j}^{\text{switch}}$ 表示在边缘节点 b 上部署或迁移参数块 j 所产生的切换成本. 上述各项成本由相应物理度量与单位成本系数相乘得到, 从而将不同量纲的开销映射到统一成本尺度.

通信成本: 当用户请求到达边缘节点 b 时, 若目标模型 i 所需的参数块未被部署, 则需从云端传输缺失参数块. 根据命中情况, 通信成本可分为完全命中、部分命中以及完全未命中这 3 种情况. 完全命中是模型 $\mathcal{J}_i$ 中所有参数块均部署在边缘, 无需通信, 仅产生本地延迟; 部分命中是未部署的部分参数块需从云端获取, 产生部分通信成本; 完全未命中是指模型 $\mathcal{J}_i$ 所有参数块均缺失, 推理需完全在云端完成. 因此, 通信成本定义为:

$$P_{\text{comm}}^{t}=\sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{I}}r_{b,i}^{t}\sum_{j\in\mathcal{J}_i}(1-x_{b,j}^{t})\cdot C_{b,j}^{\text{comm}} \tag{6}$$

其中, $r_{b,i}^{t}$ 表示时隙 t 时节点 b 接收到的服务请求, $C_{b,j}^{\text{comm}}$ 为将参数块 j 从云传至节点 b 单位请求通信成本. 当模型间共享的某些块在 b 上部署时 (即 $x_{b,j}^{t}=1$), 对应项不产生通信成本, 从而降低 P_{comm}^{t}.

更新成本: 当云端模型发生更新时, 边缘节点仅需对已部署的参数块进行同步, 如公式 (3) 所示. 则时隙 t 的更新成本为:

$$P_{\text{update}}^{t}=\sum_{b\in\mathcal{B}}\sum_{j\in\mathcal{J}}y_{b,j}^{t}C_{b,j}^{\text{update}} \tag{7}$$

其中, $y_{b,j}^{t}=1$ 表示时隙 t 对节点 b 的块执行更新; $C_{b,j}^{\text{update}}$ 为该块一次更新的资源成本, 更新成本通常低于部署和切换成本.

精度成本: 精度成本反映由于模型陈旧或未更新导致推理性能下降而产生的系统损失. AOI 越大, 模型越陈旧, 推理精度越低, 由此引起的业务损失越大, 因此精度成本越高. 精度成本的计算公式定义如下:

$$P_{\text{acc}}^{t}=\sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{I}}r_{b,i}^{t}(1-A_{b,i}^{t})C_{b,i}^{\text{acc}} \tag{8}$$

其中, $A_{b,i}^{t}$ 为模型 i 在边缘服务器 b 的当前精度 (由公式 (2) 计算). $C_{b,i}^{\text{acc}}>0$ 为模型精度不足导致的性能损失权重, 该成本体现了模型精度退化与业务性能损失之间的量化关系.

推理成本: 推理成本描述为边缘节点执行模型推理所产生的计算资源消耗. 推理的输入数据量为 $d_{b,i}^{t}$, 每单位输入数据所需计算量为 ρ_i, 则推理成本定义为:

$$P_{\text{inf}}^{t}=\sum_{b\in\mathcal{B}}\sum_{i\in\mathcal{I}}r_{b,i}^{t}\cdot 1_{b,i}\cdot\rho_i d_{b,i}C_{b}^{\text{comp}} \tag{9}$$

此时我们仅考虑全部本地/全部云端的二元情形, 令 $1_{b,i}=\prod_{j\in\mathcal{J}_i}x_{b,j}^{t}\in\{0,1\}$, 其中 $x_{b,j}^{t}\in\{0,1\}$ 表示块 j 是否部署在节点 b 上, $\mathcal{J}_i$ 表示模型 i 所需的参数块集合. 当且仅当模型 i 的全部参数均已部署时, $1_{b,i}=1$, 否则为 0. C_{b}^{comp} 表示节点 b 的单位计算资源成本.

综上所述, 在时隙 t 系统的部署成本可表示为:

$$P^t_{\text{total}} = P^t_{\text{switch}} + P^t_{\text{comm}} + P^t_{\text{update}} + P^t_{\text{acc}} + P^t_{\text{inf}} \tag{10}$$

4 问题建模与转化

4.1 问题定义

基于上述描述, 本文以最小化系统部署成本为目标, 在边缘节点存储与推理精度约束下, 建立优化问题如下:

$$\begin{cases} P_1:\ \min \dfrac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} P^t_{\text{total}} \\ \text{s.t.} \begin{cases} \sum_{j\in\mathcal{J}} x^t_{b,j}\cdot c_j \leqslant C_b, & \forall b\in\mathcal{B}, t\in\mathcal{T} \\ y^t_{b,j} \leqslant x^t_{b,j}, & \forall b\in\mathcal{B}, j\in\mathcal{J}, t\in\mathcal{T} \\ 1 \leqslant a^t_{b,j} \leqslant a_{\max}, & \forall (b,j)\text{且}x^t_{b,j}=1 \\ x^t_{b,j}, y^t_{b,j}\in\{0,1\},\ A^t_{b,i}\in[0,1] \end{cases} \end{cases} \tag{11}$$

在上述约束条件中, 约束 (11b) 确保部署的参数块总大小不超过边缘服务器的存储限制; 约束 (11c) 表明更新操作只能针对已部署的参数块进行, 从而保证更新决策的可行性; 约束 (11d) 限制参数块 AOI 的范围并进行递推, 确保模型参数不过时; 约束 (11e) 定义变量取值范围及初始状态.

4.2 问题转化

为了获得更稳定的服务决策并降低跨时段的部署波动, 我们在原问题 P_1 的基础上引入切换正则项, 并将耦合的整数优化放松为连续可解的模型. 具体的, 引入辅助变量 $z^t_{b,j}$ 来表示相邻时隙间的参数块 j 在边缘节点 b 的部署状态变化, 即 $z^t_{b,j} = [x^t_{b,j} - x^{t-1}_{b,j}]^+$. 同时, 将整数部署决策 $X=\{x^t_{b,j}\}$ 和更新决策 $Y=\{y^t_{b,j}\}$ 松弛为连续变量, 从而得到连续可微的优化问题 P_2.

$$\begin{cases} P_2:\ \min \dfrac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(P^t_{\text{comm}} + P^t_{\text{update}} + P^t_{\text{inf}} + P^t_{\text{acc}} + \sum_{b\in B}\sum_{j\in\mathcal{J}} z^t_{b,j} C^{\text{switch}}_{b,j}\right) \\ \text{s.t.} \begin{cases} \text{约束(11b)}-\text{(11d)} \\ z^t_{b,j} \geqslant x^t_{b,j} - x^{t-1}_{b,j},\ \forall b\in\mathcal{B}, j\in\mathcal{J}, t\in\mathcal{T} \\ 0 \leqslant x^t_{b,j}, y^t_{b,j}, z^t_{b,j} \leqslant 1 \end{cases} \end{cases} \tag{12}$$

正则化方法设计: 受文献 [20] 启发, 原问题中相邻时隙 t 和 $t-1$ 的部署变化会引入切换成本 $[x^t_{b,j} - x^{t-1}_{b,j}]^+ \cdot C^{\text{switch}}_{b,j}$. 为获得可微且更稳定的在线优化目标, 本文在连续松弛后, 进一步对相邻时隙部署变量的变化幅度引入 Pseudo-Huber 正则项[31], 以平滑惩罚频繁重配置行为, 从而抑制部署决策在相邻时隙上的频繁变化并提升策略稳定性. 具体地, Pseudo-Huber 正则化损失可表示为 $L = \delta^2\left(\sqrt{1+\left(\dfrac{x^t_{b,j}-x^{t-1}_{b,j}}{\delta}\right)^2}-1\right)$. 从其函数形式可以看出, 当切换差异 $diff = x^t_{b,j} - x^{t-1}_{b,j}$ 较小时, 即 $|diff| \ll \delta$, 该损失近似于二次函数 $L \approx \dfrac{1}{2}\cdot diff^2$, 此时有助于平滑小幅调整; 当切换差异较大时, 即 $|diff| \gg \delta$, 该损失近似于线性函数 $L \approx \delta\cdot|diff| - \delta^2$, 从而避免对大幅变化施加过强惩罚. 参数 δ 用于控制二次区与线性区之间的过渡范围. δ 较小时, 模型对小幅变化更敏感, 惩罚力度大; δ 较大时, 仅对较大变化施加显著惩罚, 因而具有更强的宽容性. 因此, 本文采用如下 Pseudo-Huber 正则项来平滑刻画相邻时隙部署决策的变化幅度:

$$[x^t_{b,j} - x^{t-1}_{b,j}]^+ = \delta^2\left(\sqrt{1+\left(\frac{x^t_{b,j}-x^{t-1}_{b,j}}{\delta}\right)^2}-1\right) \tag{13}$$

接下来, 我们对问题 P_1 通过连续化松弛变量 X 和 Y 得到了问题 P_2, 然后将问题 P_2 中的切换成本替换为公式 (13) 得到问题 P_3, 具体表示如下:

$$\begin{cases} P_3:\ \min \dfrac{1}{|\mathcal{T}|}\displaystyle\sum_{t\in\mathcal{T}}\left(P^t_{\text{comm}}+P^t_{\text{update}}+P^t_{\text{inf}}+P^t_{\text{acc}}+\sum_{b\in B}\sum\nolimits_{j\in\mathcal{J}} C^{\text{switch}}_{b,j}\delta^2\left(\sqrt{1+\left(\dfrac{x^t_{b,j}-x^{t-1}_{b,j}}{\delta}\right)^2}-1\right)\right) \\ \text{s.t.}\begin{cases} \text{约束(11b)}-\text{(11d)} \\ z^t_{b,j}\geqslant x^t_{b,j}-x^{t-1}_{b,j},\ \forall b\in\mathcal{B}, j\in\mathcal{J}, t\in\mathcal{T} \\ 0\leqslant x^t_{b,j}, y^t_{b,j}, z^t_{b,j}\leqslant 1 \end{cases} \end{cases} \tag{14}$$

需要说明的是, P_3 并非与原整数问题 P_1 严格等价, 而是为支持在线梯度优化构造的可微代理目标. P_1 经连续放松得到 P_2 后, 进一步采用 Pseudo-Huber 损失对切换项进行平滑近似以获得 P_3. 该近似在部署变化幅度上保持与原切换项一致的单调惩罚关系, 并由参数 δ 控制近似紧度, 从而在不改变抑制频繁切换这一优化方向的前提下提升优化稳定性.

5 低成本快速自适应模型部署与更新策略

为了解决上述问题, 我们提出了一种自适应联合部署与更新策略 PACE. PACE 框架结合了元学习与变分贝叶斯优化的思想, 用于在动态环境中实现高效的模型部署与更新决策. 具体而言, 首先利用历史服务请求数据, 通过元强化学习训练得到优化的元参数, 以捕获环境变化规律和任务间共性. 随后, 基于学到的元参数先验, 采用变分贝叶斯方法在在线阶段自适应地更新模型部署与参数状态, 从而在不确定性条件下实现高效推理与资源利用. 理论上, 我们进一步证明了 PACE 框架满足 PAC-Bayes 泛化界和次线性遗憾界, 从而保证算法在精度与收敛性能上的稳健性.

5.1 基于元强化学习的离线学习算法

在本节, 我们基于前面转化得到的目标函数 P_3, 提出了一种基于元强化学习的离线学习框架. 我们先概述所采用的元学习框架; 随后将问题 P_3 重构为马尔可夫过程 (Markov decision process, MDP), 并说明如何在该框架下应用元学习以快速适应新任务.

元学习框架: 为了应对非平稳请求分布下、具有参数共享约束的模型部署和更新问题 (问题 P_3), 本文采用双层优化结构的元强化学习框架. 该框架包括内层和外层. 内层是任务内适应, 即在单个任务内, 通过少量经验快速调整策略以适应当前请求模型; 外层则是跨任务元更新, 跨任务学习一个良好初始化的元策略, 使其在遇到新任务时以极少的更新步数实现快速收敛, 从而在服务精度、资源消耗与系统成本之间实现有效权衡.

元学习的核心目标是在任务分布 $p(\mathcal{Q})$ 上, 学习到一个能够快速适应新任务的策略初始化参数 θ. 在强化学习场景下, 这可形式化为:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{Q}_i\sim p(\mathcal{Q})}[\mathcal{L}_{\mathcal{Q}_i}(\phi_i)] \tag{15}$$

其中, $\mathcal{Q}_i$ 为第 i 个任务, ϕ_i 表示在任务 $\mathcal{Q}_i$ 上以少量梯度步从初始参数 θ 适应后的参数 (即内层更新后的参数), $\mathcal{L}_{\mathcal{Q}_i}$ 是任务 $\mathcal{Q}_i$ 上的损失.

任务 $\mathcal{Q}_i$ 的定义: 根据第 2. 1 节文件请求模型的描述, 每个任务 $\mathcal{Q}_i$ 对应一个特定时间段内 R 个请求, 具有不同的请求模式. 为了适用于元学习双层优化结构, 将单个任务 $\mathcal{Q}_i$ 内的请求数据划分为支持集 $\mathcal{D}_i^{\text{support}}$ 和查询集 $\mathcal{D}_i^{\text{query}}$, 定义为:

$$\mathcal{Q}_i = \{\mathcal{D}_i^{\text{support}}, \mathcal{D}_i^{\text{query}}, t_i^{\text{start}}, t_i^{\text{end}}\} \tag{16}$$

其中, $\mathcal{D}_i^{\text{support}}$ 用于内层快速适应, $\mathcal{D}_i^{\text{query}}$ 用于外层策略评估, t_i^{start} 和 t_i^{end} 表示任务的时间范围. 在该划分下, 任务内请求分布相对稳定而任务间发生漂移, 因此策略需在任务切换后用少量样本快速适配.

原问题重构: 为在元强化学习框架下求解问题 P_3, 将其重构为马尔可夫过程 MDP, 定义的状态、动作和奖赏函数如下.

状态 (State) s^t: 状态向量包含环境的相关信息, 用于描述系统在时隙 t 的状况. 具体包括每个边缘节点 b 的状

态 $s_b^t = \{c_b^t, a_{b,j}^t, acc_{b,j}^t, c_j | j \in \mathcal{J}\}$, 其中 c_b^t 表示节点 b 的缓存状态, $a_{b,j}^t$ 表示参数块 j 的 AOI, $acc_{b,j}^t$ 表示精度状态, c_j 为参数块大小.

动作 (Action) a^t: 在每个时隙 t, 动作由联合部署和更新决策组成, 即 $a^t = [X_t^{\text{deploy}}, Y_t^{\text{update}}]$, 其中 $X_t^{\text{deploy}} = \{x_{b,j}^t \in [0,1]\}$, $Y_t^{\text{update}} = \{y_{b,i,a}^t \in [0,1]\}$. 动作表示每个节点对参数块的部署和更新倾向程度. 为了保证合理性, 引入以下约束:

存储约束: 节点缓存存储容量有限, 已部署参数块总大小不得超过节点存储上限.

更新约束: 只有已部署的参数块可执行更新, 更新动作受部署状态限制.

依赖关系约束: 如果模型的特定参数块需要更新, 其依赖的共享参数块也必须更新, 以保证模型完整性.

奖励函数 (Reward) r^t: 在时隙 t 执行动作 a^t 后, 系统返回即时奖励, 定义为负的总成本 $r^t = -(P_{\text{comm}}^t + P_{\text{update}}^t + P_{\text{inf}}^t + P_{\text{acc}}^t + L_{\text{switch}}^t)$, 其中 L_{switch}^t 是 Pseudo-Huber 正则化的切换成本项. 该奖励设计的目的是最小化多目标成本, 在保证服务精度的同时, 平衡通信、更新和切换成本, 实现整体系统性能优化.

基于元强化学习的离线学习框架: 为了快速适应动态变化的边缘请求环境, 我们希望利用前 M_{tr} 个任务的历史数据及其最优配置, 学习一个能够在新任务上快速调整的元策略 θ^*. 令前 M_{tr} 个任务的经验分布记为 $\hat{p}_{M_{\text{tr}}}(\mathcal{Q})$. 因此元强化学习的目标是:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\mathcal{Q}_i \sim \hat{p}_{M_{\text{tr}}}(\mathcal{T})}[\mathcal{L}_{\mathcal{Q}_i}^{\text{query}}(\phi_i(\theta))] \tag{17}$$

其中, $\phi_i(\theta)$ 是在任务 $\mathcal{T}_i$ 内基于支持集 $\mathcal{D}_i^{\text{support}}$ 通过少量梯度更新得到的任务特定参数:

$$\phi_i = \theta - \beta \nabla_\theta \left(\frac{1}{|\mathcal{D}_i^{\text{support}}|} \sum_{t \in \mathcal{D}_i^{\text{support}}} C_t^{\mathcal{Q}_i}(s_t, \pi_\theta(s_t)) \right) \tag{18}$$

其中, β 为内循环学习率. $C_t^{\mathcal{Q}_i}$ 为当前任务 $\mathcal{Q}_i$ 上的成本值. 外循环在查询集 $\mathcal{D}_i^{\text{query}}$ 上计算损失并更新元参数:

$$\theta \leftarrow \theta - \gamma \nabla_\theta \sum_{\mathcal{Q}_i} \frac{1}{|\mathcal{D}_i^{\text{query}}|} \sum_{t \in \mathcal{D}_i^{\text{query}}} C_t^{\mathcal{Q}_i}(s_t, \pi_{\phi_i}(s_t)) \tag{19}$$

其中, γ 为元学习率. 对应的一步内层更新下的元梯度为:

$$\nabla_\theta \mathcal{L}_{\mathcal{Q}_i}^{\text{query}}(\phi_i(\theta)) = \nabla_{\phi_i} \mathcal{L}_{\mathcal{Q}_i}^{\text{query}} \cdot (1 - \beta \nabla_\theta^2 \mathcal{L}_{\mathcal{Q}_i}^{\text{support}}(\theta)) \tag{20}$$

其中, $\mathcal{L}_{\mathcal{Q}_i}^{\text{support}}$ 和 $\mathcal{L}_{\mathcal{Q}_i}^{\text{query}}$ 分别对应公式 (18) 与 (19) 中的经验平均损失. 由于部署动作作为二值约束, 而神经网络输出连续输出, 我们采用直通估计器 [32]进行近似, 使梯度能够端到端传播:

$$x^{\text{deploy}} = \sigma(a^{\text{deploy}}) + (h^{\text{deploy}} - \sigma(a^{\text{deploy}})) \odot 1_{\text{stop}_{\text{gradient}}} \tag{21}$$

其中, h^{deploy} 是满足容量和依赖约束的二值部署策略, 并满足容量与依赖约束, 因此实际执行动作始终落在 P_1 的可行域内. P_3 的连续优化仅用于提供可微学习信号与稳定的在线更新. $\sigma(\cdot)$ 用于反向传播计算梯度, $\odot$ 表示逐元素乘法, $1_{\text{stop}_{\text{gradient}}}$ 在前向传播时为 1, 在反向传播时梯度为 0.

通过离线训练得到的元策略 θ^* 作为在线阶段的初始策略, 结合新任务请求数据, 通过内循环快速适应, 实现参数块级部署与更新的最优决策, 具体的元训练离线训练算法如算法 1 所示.

算法 1. 基于元学习的离线训练算法.

输入: 序列请求数据 D, 元策略网络初始化参数 θ;
输出: 训练好的元策略参数 θ^*.

1. 将每个任务请求数据划分为支持数据集 $\mathcal{D}_i^{\text{support}}$ 和查询数据集 $\mathcal{D}_i^{\text{query}}$
2. for epoch =1 to E do
3. 从任务分布 $\mathcal{P}(\mathcal{Q})$ 中采样一批任务 $\{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_N\}$
4. for 每个任务 $\mathcal{Q}_i$ do
5. $\phi_i \leftarrow \theta$ //复制元参数

6. //内循环: 任务特定适应
7. for 每个 $t \in \mathcal{D}_i^{\text{support}}$ do
8. 使用当前状态 s^t, 通过策略网络生成动作 $a^t = [X_t^{\text{deploy}}, Y_t^{\text{update}}]$
9. 使用 STE 方法公式 (21) 保证动作满足容量、更新和依赖约束
10. 应用动作到环境, 获得即时成本 $C_t(s_t, a_t)$
11. 累积计算损失值 $\mathcal{L}^{\text{support}}$
12. end for
13. 使用内循环梯度公式 (18) 更新 ϕ_i
14. //外循环: 元更新
15. for 每个 $\mathcal{D}_i^{\text{requery}}$ 中每个请求 do
16. 根据 ϕ_i 生成动作, 保证约束
17. 应用到环境获得即时成本, 累积外循环损失 $\mathcal{L}^{\text{query}}$
18. end for
19. end for
20. 使用外循环梯度公式 (19) 更新元参数 θ
21. end for

5.2 基于元知识先验的变分贝叶斯在线联合优化

我们已通过元学习方法获得具备快速适应能力的元策略参数 θ^*, 该策略在多种边缘计算场景中表现出良好的泛化能力. 然而, 传统的元学习方法多基于点估计参数, 难以在动态环境下有效刻画决策不确定性. 为此, 本文在离线元学习得到的优化参数基础上, 进一步提出一种基于元知识先验的变分贝叶斯在线联合优化方法. 该方法将离线阶段学习到的元知识视为参数的先验分布, 并在在线阶段结合新观测数据进行贝叶斯更新, 从而实现对环境变化的快速适应与不确定性感知的联合优化. 接下来, 我们将详细介绍所提出的贝叶斯在线部署流程.

变分贝叶斯在线学习阶段: 在离线阶段基于元强化学习公式 (17) 优化得到的元策略参数 θ^*, 作为贝叶斯模型的高斯先验分布均值. 由此可初始化先验为:

$$p(w) = \mathcal{N}(w \mid \theta^*, \tau_p^2 I) \tag{22}$$

其中, θ^* 为离线元训练得到的策略网络初始化参数, 即跨任务迁移的元知识. 先验方差 τ_p^2 用于刻画对该元知识的置信度, 从而在新任务到来时减少从零探索并加速适配. 该先验反映了模型对元学习知识的置信度, 其中较小的 τ_p^2 表示对元学习知识的强置信, 从而鼓励在线学习保持接近离线学习所得的元知识. 这样便将元学习获得的点估计转换为概率形式, 并为后续的贝叶斯推断奠定基础. 在在线决策初始阶段, 后验分布与先验近似相同:

$$q_{\phi_0}(w) \approx p(w) = \mathcal{N}(w \mid \theta^*, \tau_p^2 I) \tag{23}$$

其中, $q_{\phi_0}(w)$ 表示时隙 $t = 0$ 时的后验分布, 参数为 $\phi_0 = \{\mu_0, \tau_0\}$.

为了通过先验知识和观测数据来更新对未知参数的信念, 需要结合对环境决策的先验知识和观测数据来计算后验分布, 但由于真实后验分布 $p(w \mid \mathcal{D})$ 难以解析求解, 在线阶段采用变分推理方法, 用近似分布 $q_\phi(w)$ 来近似真实后验分布. 为了简化优化过程, 假设 $q_\phi(w)$ 服从对角高斯分布, 其协方差矩阵为对角阵:

$$q_\phi(w) = \mathcal{N}(w \mid \mu, \text{diag}(\tau^2)) \tag{24}$$

其中, $\phi = \{\mu, \tau\}$ 为变分参数. 变分推理的目标是优化 ϕ, 使得近似分布与真实后验之间的 Kullback-Leibler (KL) 散度最小化, 其等价于最大化证据下界 (evidence lower bound, ELBO)[33], 具体 ELBO 定义为:

$$\mathcal{L}(\phi) = \mathbb{E}_{q_\phi(w)}[\log p(\mathcal{D}|w)] - \delta \cdot KL(q_\phi(w) \| p(w)) \tag{25}$$

在线部署场景中, ELBO 可具体化为期望累积成本最小化问题:

$$\mathcal{L}(\phi)=\mathbb{E}_{w\sim q_\phi(w)}\left[\sum_{t=1}^{T}C_t(s_t,\pi_w(s_t))\right]+\delta\cdot KL(q_\phi(w)\|p(w)) \tag{26}$$

其中, 等号右侧第 1 项是期望累积成本 (对应联合部署与更新成本), 鼓励策略在长期内实现最优部署. 第 2 项为正则项, 防止后验过度偏离先验. δ 是权衡系数, 用于平衡先验与适应速度. 较大的 δ 强化先验约束, 提高稳定性. 较小的 δ 则增强模型的自适应性.

在每个时隙 t, 系统通过随机梯度变分贝叶斯算法不断更新参数:

$$\phi_{t+1}=\phi_t-\eta\nabla_\phi\mathcal{L}(\phi_t) \tag{27}$$

此时后验分布逐步偏离先验并吸收在线经验:

$$q_{\phi_t}(w)=\mathcal{N}(w\mid\mu_t,\mathrm{diag}(\tau_t^2)) \tag{28}$$

其中, 均值 μ_t 逼近当前最优策略, 方差 τ_t^2 随学习而逐渐减小, 表示决策置信度的提升, 最终系统收敛到反映当前环境状态的最优后验.

$$q_{\phi^*}(w)=\mathcal{N}(w\mid\mu^*,\mathrm{diag}(\tau^{*2})) \tag{29}$$

在线决策过程中, 当新的请求到达时, 系统根据当前后验均值生成部署和更新动作:

$$a_t=\pi_{\mu_t}(s_t) \tag{30}$$

其中, μ_t 是后验分布的均值, 后验方差 σ^2 提供了不确定性信息. 同时, 为提升探索能力, 可基于后验不确定性进行采样:

$$a_t=\pi_{\tilde{w}}(s_t),\ \tilde{w}=\mu_t+\epsilon\cdot\tau_t,\ \epsilon\sim\mathcal{N}(0,1) \tag{31}$$

该策略在不确定性较高的参数方向上增强探索, 从而提高模型在动态环境中的适应性与学习效率. 具体的基于元学习先验的变分贝叶斯框架如算法 2 所示.

算法 2. 基于元学习先验的变分贝叶斯框架.

输入: 元学习参数 θ^* (来自算法 1)、当前请求数据 $\mathcal{D}_c$, 初始先验方差 τ_p^2, 变分学习率, *KL* 权重 δ, 更新间隔, MC 采样次数 (用于估计期望成本);

输出: 每个时隙的在线模型部署 $X_t=\{x_{b,j}^t\}$ 和更新决策 $Y_t=\{y_{b,j}^t\}$.

1. 使用算法 1 获得的元参数 θ^* 初始化先验分布, 见公式 (22)
2. 使用公式 (23) 初始化后验 $q_{\phi_0}(w)$
3. for 每个在线时隙 t=1, 2,…, T do
4. 使用公式 (30) 生成联合决策, 得到 X_t 和 Y_t
5. 与环境交互, 获得即时成本
6. if $t\%\,update_{\text{interval}}==0$ then
7. 基于公式 (26) 与 (27) 的变分贝叶斯更新
8. 使用 MC 采样近似估计 ELBO
9. 计算梯度并用公式 (27) 更新后验参数
10. end if
11. end for
12. 输出返回部署和更新决策

复杂度与运行开销: PACE 的计算由离线与在线两部分组成. 离线阶段通过元强化学习得到元策略参数, 并用于初始化在线变分贝叶斯先验, 该阶段在云端完成. 在线阶段在每个时隙生成一次参数块级联合部署与更新决策, 并进行可行化处理以满足容量约束; 决策变量以边缘服务器集合与参数块集合为索引, 因此在线阶段的主要计算与存储开销随边缘服务器数量与参数块数量线性或近线性增长. 在线贝叶斯更新采用递推形式, 并按预设间隔触

发, 从而避免在边缘侧进行高开销的在线重训练. 作为对比, 规则型基线方法主要由简单统计更新或排序构成, 在线开销较低; 深度强化学习基线主要开销为策略网络前向推断, 若包含在线训练则还需额外梯度更新. 总体而言, PACE 将高开销训练移至云端, 并在边缘侧保持轻量推断与间歇更新, 适用于资源受限的边缘智能部署场景.

5.3 理论验证与分析

为了进一步验证提出的 PACE 算法的理论有效性, 我们从泛化性与收敛性两个方面进行分析. 具体地, 首先给出该方法在新任务场景下的 PAC-Bayes 泛化界[32], 以说明模型在未见请求模式与节点配置下的性能之间的差距; 随后给出其在长期在线学习过程中的次线性遗憾界, 以证明该算法在累积决策成本上的收敛与稳定性.

定理 1. PAC-Bayes 泛化界. 对于问题 P_3, 在线学习过程的期望未来成本满足:

$$\mathbb{E}[C_{\text{future}}] \leqslant \mathbb{E}[C_{\text{historical}}] + \kappa \cdot KL(q_\phi(w)\|p(w)) + \frac{\log n\delta}{2n} \tag{32}$$

该界说明, 在变分贝叶斯框架下学习到的策略在新环境配置、不同请求模式及节点分布时, 其期望成本变化是有界且可预测的.

证明: 首先建立变分框架, 令 $\mathcal{F}(q_\phi) = \mathbb{E}_{h\sim q(\phi)}[L_D(h) - L_S(h)]$, 其中 $L_D(h)$ 是期望风险 (对应于问题 P_3 中的期望成本), $L_S(h)$ 是经验风险 (对应历史观测成本的平均). 根据 Donsker-Varadhan 变分公式, 对于任意 $\xi > 0$, 有:

$$\mathcal{F}(q_\phi) \leqslant \frac{KL(q_\phi(w)\|p(w)) + \log \mathbb{E}_{h\sim q(\phi)}[\mathrm{e}^{\lambda(L_D(h)-L_S(h))}]}{\xi} \tag{33}$$

其次, 在问题 P_3 中, 由于缓存容量及请求频率均有限, 成本函数有界, 因此存在常数 κ 使得 $L_D(h) - L_S(h) \in [-\kappa, \kappa]$, 其中 κ 为成本差的上界常数, 应用 Hoeffding 引理:

$$\log \mathbb{E}_{h\sim q(\phi)}[\mathrm{e}^{\lambda(L_D(h)-L_S(h))}] \leqslant \frac{\xi^2 B^2}{8n} \tag{34}$$

取 $\lambda = \sqrt{\dfrac{8n}{B^2}\log\dfrac{n}{\omega}}$, 代入可得:

$$\mathcal{F}(q_\phi) \leqslant B \cdot \sqrt{\frac{KL(q_\phi(w)\|p(w)) + \log\dfrac{n}{\omega}}{2n}} \tag{35}$$

证毕.

定理 2. 次线性遗憾界. 贝叶斯元学习框架的期望遗憾满足:

$$\mathbb{E}[Regret(T)] \leqslant \tilde{O}(\sqrt{dT\log T}) \tag{36}$$

证明: 首先引入信息比 $\Gamma_t = \dfrac{\mathbb{E}[C_t(\pi_{w_t}) - C_t(\pi_{w^*})]^2}{I(w^*; (s_t, a_t, c_t)|\mathcal{H}_{t-1})}$, 其中 $I(\cdot;\cdot)$ 表示互信息, $\mathcal{H}_{t-1}$ 为时刻 $t-1$ 的历史信息. 根据链式法则与信息比分析, 可得策略的期望累计遗憾满足:

$$\mathbb{E}[Regret(T)] \leqslant \sqrt{T \cdot \Gamma_{\max} \cdot I(w^*; \mathcal{H}_T)} \tag{37}$$

由于策略参数空间是 d 维的, 根据信息理论, 在高斯先验与有界观测噪声条件下有:

$$I(w^*; \mathcal{H}_T) \leqslant \tilde{O}(dT\log T) \tag{38}$$

并且成本函数有界, 因此 $\Gamma_t \leqslant \Gamma_{\max}$. 综合公式 (37) 和 (38) 可得:

$$\mathbb{E}[Regret(T)] \leqslant \sqrt{T \cdot \Gamma_{\max} \cdot \tilde{O}(dT\log T)} = \tilde{O}(\sqrt{dT\log T}) \tag{39}$$

即算法在长期学习中具有次线性遗憾界, 表明其平均遗憾随时间增长逐渐减小, 从而体现出较好长期稳定性.

6 实验与评估

为验证所提 PACE 算法的有效性与稳定性, 本文通过系统的模拟实验从 3 方面进行评估: (1) 与多种基线方法的总体性能对比; (2) 对离线元学习与在线变分更新过程的验证与分析; (3) 参数共享与成本项设计的消融实验研究.

6.1 实验设置

本研究构建了一个参数共享的模型库. 基于 CIFAR-100 数据集[32], 选取 ResNet 系列 (ResNet-18/34/50/101/152)、VGG16 与 MobileNet 作为骨干网络. 首先对各骨干网络进行通用特征预训练, 得到可共享的骨干参数块; 随后针对每个具体子类进行二分类微调, 仅更新少量子类相关的轻量参数块, 从而实现共享参数与子类特化的分层训练方案. 最终得到 7 组共享参数的基模型以及 700 个子类特定模型, 其中同一骨干网络下的子类特定模型共享相同的骨干参数块. 由于 CIFAR-100 数据集包含 100 个细粒度类别, 并在 7 种骨干网络上分别进行子类微调, 因而模型库理论上共包含 707 个模型.

为构造多样化的服务请求分布, 我们参考文献 [24] 的方法, 按 Zipf 分布生成模型服务请求序列, 以刻画边缘环境中常见的热点访问与长尾特征. 具体而言, 请求分布指数设为 $\alpha = 2.5 + W$, 其中 W 为区间 $[-d, d]$ 上的均匀随机变量. 通过调节 d 可控制任务间的相似度, 本实验取 $d = 1.25$, 以保证分布不发散, 同时在少数情形下产生较大的方差[29]. 本文中的下游任务指边缘侧的图像识别推理服务请求, 请求到达表示对某一子类识别服务的调用. 不同任务在请求到达频率上的差异用于刻画动态、非平稳的服务需求, 并驱动模型部署与更新决策. 此外, 将单个任务长度设为 $T = 100$, 参考网络缓存中元学习的典型设置[29], 该长度对应于边缘环境中的一个准平稳窗口, 既保证了少样本适应所需的数据量, 又反映了请求分布随时间漂移的非平稳特性. 具体的实验参数设置如表 1 所示, 其中缓存容量 C_b 的单位为 MB, 用于表示边缘服务器可用于存储模型参数文件的总容量上限.

表 1 实验参数设置

参数	取值
缓存容量 C_b (MB)	200
内循环学习 β	1E–1
外循环参数 γ	1E–2
单个任务 $\mathcal{Q}_n$ 的请求数	100
AOI的衰减系数 λ	0.1
KL权重 δ	1

仿真中为便于控制到达过程并统一对比, 在高并发场景下可采用窗口聚合并在窗口边界更新决策, 以降低决策频率并提升可实现性. 在部署成本设定中, 我们认为各项成本随模型大小线性增长, 因而不同规模模型在部署成本上存在差异. 除网络传输外, 单位数据量 (MB) 的成本分别设为: 推理 0.002、通信 0.02、精度 0.05、切换 0.02、更新 0.01 (成本为归一化单位). 在服务请求阶段, 若所需服务可在本地直接获得, 则网络传输成本为 0.001/MB; 若需从云端获取, 则为 0.1/MB. 本文各项成本按“物理度量或数据量×单位成本系数”计算, 其中数据量由模型大小与参数块划分确定, 用于刻画一次更新、切换或传输实际涉及的权重规模; 单位成本系数用于将不同量纲的开销映射到统一成本尺度. 另外, 本地获取与云端获取采用不同的单位传输成本, 以区分边缘侧直接命中与跨云边获取在传输与等待开销上的差异. 部署成本由上述单位成本与相应模型大小逐项相乘并求和得到, 用以评价不同模型在缓存与调用过程中的总体开销.

6.2 基 线

在本文中, 我们采用模型部署成本作为评估不同部署算法性能的指标. 将所提出的 PACE 方法与以下几种部署方法进行对比.

(1) Meta_OCO[29]: 一种结合元学习与梯度上升进行模型部署的方法, 利用任务间的相似性加速元学习过程. Meta_OCO 具有动态适应性, 但未考虑请求间的参数共享特性及服务状态的更新.

(2) MPUTA[20]: 一种联合优化方法, 同时处理模型部署和新鲜度敏感任务分配. 该方法通过将联合优化问题分解, 使用随机轮技术解决模型部署子问题, 并采用图匹配技术处理任务匹配子问题. MPUTA 在新鲜度感知方面表现出色, 但缺乏动态适应性.

(3) Meta[34]: 保持本文提出的离线训练策略, 在线阶段直接使用元学习得到的参数进行模型部署.

(4) TD3 (twin delayed deep deterministic policy gradient)[35]: 一种连续动作空间的深度强化学习方法. 我们将其作为深度学习基线, 在与本文相同的状态输入下学习部署与更新策略. TD3 输出连续动作后, 采用与本文一致的可行化映射方式生成满足容量约束的二值部署决策并执行, 以保证约束处理与执行一致.

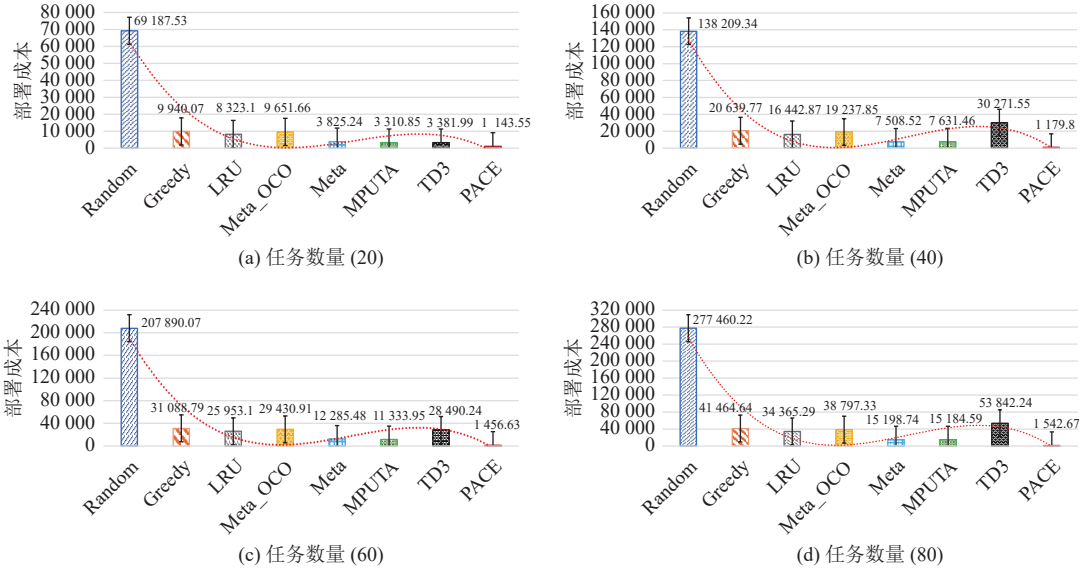
(5) LRU[36]: 一种经典的缓存管理方法, 通过服务请求的先后顺序来选择缓存中部署和驱逐的服务项. 如前所述.

(6) Greedy[37]: 一种贪心策略, 通过维持给定时间窗口内的历史数据, 根据模型大小与请求频率的比例选择最优的进行模型部署, 并随机选择需要更新的模型.

(7) Random[38]: 一种随机部署和更新方法, 每次从模型库中随机选择满足容量约束的模型来处理到来的服务请求.

6.3 性能对比

首先进行模型部署成本策略的实验, 验证 PACE 的自适应部署算法能够以更低的部署成本满足具有多动态特征的服务请求需求. 我们模拟了多个请求分布任务下不同算法的模型部署成本性能. 不同的任务主要有以下不同: (1) 具有不同的访问热度分布 (Zipf 中 α 参数不同); (2) 每个请求所关注的数据集子集不同 (如任务 A 关注 [“猫”, “狗”, “鸟”], 任务 B 则关注 [“汽车”, “飞机”, “船”]); (3) 任务复杂度不同, 简单任务访问更分散, 复杂任务访问更集中. 在本次实验中任务数量设置范围为 [20, 80], 旨在模拟系统在长期运行中经历不同频次的环境变化, 以评估算法的持续适应能力. 图 4 展示 PACE 和基线方法在不同任务数量下的部署成本.



(a) 任务数量 (20)　(b) 任务数量 (40)

(c) 任务数量 (60)　(d) 任务数量 (80)

图 4　不同任务数量请求分布下的部署成本

PACE 的部署成本比基线低 65%–68%. LRU 是一种简单的根据当前请求进行缓存更新的方法, 在多任务动态请求环境下部署成本相对较低; Random 由于随机决策缺乏针对性, 部署成本通常较高. Greedy 算法即使在设置时考虑模型大小和请求频率, 但仍难以应对大模型数据库下模型请求的多样性, 从而引入较高的部署成本. MPUTA 算法考虑了联合优化问题, 但针对多请求变化的情况时, 算法的适应性仍然表现不足, 造成较高的切换等成本, 因此其模型部署成本仍有提升空间. Meta_OCO 可以根据任务请求的相似性动态适应不同的请求分布, 但未显式考虑模型更新带来的更新成本权衡, 因此在边缘场景下的部署成本较高. 此外, 从图 4 可以看出, TD3 的总体部署成本高于 PACE, 且从表 2 的成本分解可见其主要开销来自切换成本, 这表明在任务分布变化更频繁的设置下, 仅依赖常规深度策略学习更难稳定抑制频繁重配置带来的成本. Meta 是本文所提出的离线算法的扩展, 具有较强的动态适应性, 但在线过程中未能充分考虑新的请求分布来持续适应请求需求, 从而表现欠佳. PACE 不仅可以充分利

用历史请求数据获取较好的初始参数, 还可以将新到来的请求转换为联合优化经验, 因此在 Meta 算法的基础上进一步降低了部署成本.

表 2　不同算法在各项细分成本上的性能对比

算法	切换成本	通信成本	更新成本	精度成本	推理成本	部署成本
Random	60 183.39	193.86	8 810.15	0	0.12	69 187.53
Greedy	1 189.75	15.56	8 731.03	0	3.73	9 940.07
LRU	566.92	41.10	7 711.86	0	3.21	8 323.09
Meta_OCO	98.46	182.77	9 370.07	0	0.35	9 651.65
MPUTA	3 122.44	188.02	0	0	0.24	3 310.70
TD3	2 542.39	200.00	639.60	0	0	3 381.99
Meta	3 255.42	179.01	388.61	1.77	0.42	3 825.23
PACE (Ours)	56.80	115.06	969.30	0.67	1.72	1 143.55

为进一步分析公式 (10) 中部署成本各组成项对总体结果的贡献, 我们在与图 4 相同的实验设置下统计了各算法在切换成本、通信成本、更新成本、精度成本与推理成本这 5 项细分指标上的具体数值, 并给出其加和得到的部署成本, 如表 2 所示. 需要说明的是, 表 2 中的精度成本是依据公式 (8) 由模型精度退化折算得到的累计成本项, 其数值不仅取决于模型精度本身, 还受到请求触发情况与成本权重设置的共同影响. 因此, 尽管表 2 中不同算法的模型精度存在一定差异, 但折算后的精度成本在当前实验设定下整体较小, 未构成总部署成本的主要来源.

由表 2 可见, PACE 的总部署成本优势主要来自切换成本与更新成本的降低. 切换成本显著下降表明其能够减少动态环境下参数块在边缘侧的频繁换入换出, 从而降低重配置开销. 更新成本的下降说明其能够避免不必要的过度更新, 减少由权重拉取与替换带来的系统开销. 与此同时, PACE 的通信成本与推理成本保持在较低水平, 表明总体收益并非通过将开销转移到其他成本项获得, 而是源于更合理的部署与更新决策.

此外, 为验证算法服务质量与响应效率, 我们给出不同算法的命中率与模型精度对比结果, 如表 3 所示.

表 3　不同算法的命中率与模型精度对比

算法	命中率	模型精度	部署成本
Random	0.379 0	0.772 4	69 187.52
Greedy	0.771 8	0.771 8	9 940.07
LRU	0.901 2	0.771 9	8 323.09
Meta_OCO	0.379 2	0.772 0	9 651.65
MPUTA	0.518 0	0.757 7	3 310.70
Meta	0.232 8	0.769 7	3 825.23
TD3	0.377 0	0.783 6	3 381.99
PACE (Ours)	0.666 3	0.754 7	1 143.55

由表 3 可见, PACE 在获得更低部署成本的同时, 模型精度并未出现大幅下降. 相较于 Random、Greedy 与 LRU 等基线, 精度差异整体控制在约 3% 以内, 表明部署成本的下降并未带来过大的精度代价. 表 3 同时给出命中率, 用于反映请求在边缘侧直接完成服务的比例. 结果显示, PACE 的命中率高于 Random 与 Meta 等基线, 说明其能够在边缘侧直接响应更大比例的请求, 从而减少潜在的云边交互需求. 虽然 Greedy 与 LRU 的命中率更高, 但对应的部署成本显著更大. 综合部署成本、模型精度与命中率这 3 项结果, PACE 在更低成本与服务质量之间实现了更均衡的权衡.

6.4　参数敏感性分析

为评估关键实验参数变化对结果的影响, 我们补充参数敏感性分析. 除被扰动参数外, 其余实验设置均与主实验保持一致, 并在任务数量为 20 的设置下统计各算法的部署成本. 我们分别考察边缘缓存容量、任务窗口长度以及请求分布强度这 3 类参数, 以验证结论在合理参数范围内的稳健性.

边缘缓存容量用于刻画资源受限程度. 我们在保持其余设置不变的情况下, 缓存容量分别设置为 200、500

与 800, 并对比各算法的部署成本, 如表 4 所示.

表 4　不同缓存容量下的部署成本对比

缓存容量	Random	Greedy	LRU	Meta_OCO	Meta	MPUTA	TD3	PACE
200	69187.53	9940.07	8323.10	9651.66	3825.24	3310.85	3381.99	1143.55
500	43247.87	6589.37	6784.32	5572.95	2253.60	2560.64	2995.42	1004.70
800	17054.80	2613.28	2925.97	2345.38	1036.04	2206.78	1255.91	563.19

任务窗口长度 T 反映任务内统计稳定性与在线适应难度. 我们在保持其余设置不变的情况下, 将 T 取 50、100 与 200, 并对比各算法的总部署成本, 如表 5 所示.

表 5　不同任务窗口长度下的部署成本对比

任务长度	Random	Greedy	LRU	Meta_OCO	Meta	MPUTA	TD3	PACE
50	34770.91	4315.04	2128.78	4296.00	3732.43	1066.26	1363.99	1028.37
100	69274.16	10609.43	8819.14	8695.21	3816.61	4783.87	6061.90	1143.39
200	138499.93	17490.71	8019.18	16990.4	3855.72	2557.01	32152.18	1061.86

请求分布强度用于刻画热点集中程度与分布漂移带来的环境变化. 我们在保持其余设置不变的情况下, 将分布强度参数取 0.5、1.25 与 2, 并对比各算法的总部署成本, 如表 6 所示.

表 6　不同请求分布强度下的部署成本对比

请求分布强度	Random	Greedy	LRU	Meta_OCO	Meta	MPUTA	TD3	PACE
0.5	69434.29	8442.21	3987.56	8451.80	3773.90	1069.61	8783.71	1029.02
1.25	69204.00	10614.95	8785.77	8724.62	3816.61	4783.87	3774.00	1028.92
2	69358.82	8961.07	4225.56	8382.88	3772.42	1632.68	15526.14	1029.66

从表 4–表 6 可以看出, 缓存容量增大能够普遍降低各算法的部署成本, 符合资源更充裕时切换与更新压力减轻的直觉. 任务窗口长度与请求分布强度的变化会使部分基线方法的部署成本出现更明显波动, 说明其对环境统计特性更为敏感. 相比之下, PACE 在 3 类参数的合理取值范围内始终保持较低的部署成本且波动较小, 表明本文方法对关键参数变化具有较好的稳健性, 主实验结论不依赖于特定参数设定.

6.5　离线学习有效性验证

为了探究 PACE 实现的基于离线元学习先验对模型部署成本的有效性, 我们对不同的初始化先验算法进行了测试实验, 图 5 展示了初始算法对模型部署算法的实验结果. 其中 Standard_BO 指的是用标准的正态先验分布初始化变分贝叶斯先验的网络参数; Random_BO 和 Zero_BO 是指分别使用随机和较小的值初始化模型参数. 从图 5 可以看出, 所提出的 PACE 算法在基于元参数先验的基础上进行在线部署时的成本更少, 这主要是由于基于元参数初始化的算法能够从经验有效的参数区域开始, 并能够更加快速地在更优的策略附近进行局部搜索, 因此能够以较少的成本获得更合理的资源分配方案.

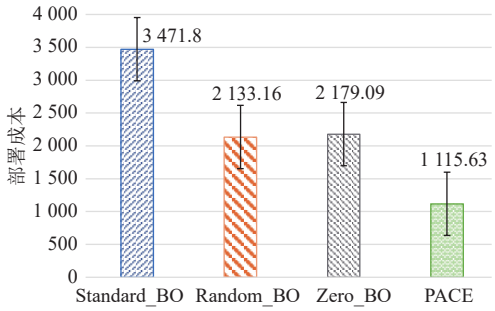


图 5　不同初始化先验算法对模型部署成本的影响

6.6 参数共享对算法的影响

为了进一步探究参数共享与非共享情况下对算法性能的影响, 模型共享被认为会直接影响模型存储大小和成本计算方式. 例如, 共享模型可以减少冗余存储, 同时降低在不同模型间切换和更新参数时的成本. 因此, 基于这些受共享影响的因素, 分析了共享与非共享情况下对算法最终性能的影响. 在实验中, 不同算法在模型共享与非共享条件下的性能进行了详细对比, 重点关注模型部署成本的变化. 图 6 比较了参数共享与非共享条件下各算法的部署成本. 为定量刻画参数共享带来的收益, 我们进一步计算共享相对非共享的部署成本降幅. 结果显示, Random、Greedy、LRU、Meta_OCO、MPUTA、Meta、TD3 与 PACE 的降幅分别为 4.5%、4.5%、10.4%、2.7%、33.2%、0.7%、1.8% 与 1.8%. 总体上, 参数共享能够减少模型间重复缓存的参数块, 从而降低部署与重配置相关开销. 不同算法的降幅存在差异, 主要是由于其部署策略稳定性不同: 对切换或更新更频繁的方法, 共享带来的收益更明显; 对策略更稳定的方法, 共享的收益表现为边际下降. 此外, 共享模型还可能带来间接优化效果, 例如减少内存访问频次、提高缓存利用率, 从而进一步降低系统成本. 具体来看, PACE 算法在共享条件下表现出明显更小的部署成本, 表明其在利用共享模型优化存储和计算成本方面比其他算法更具优势, 同时在实际应用中展现出更高的资源利用效率.

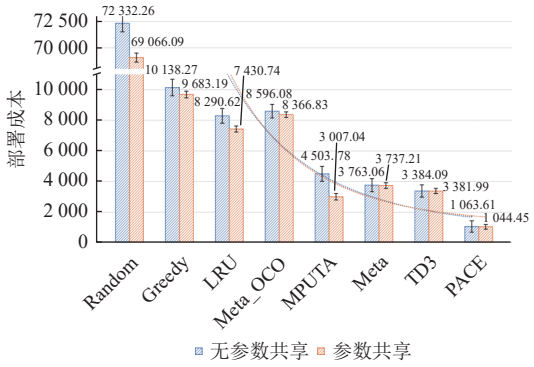


图 6 参数共享与非共享对算法性能的影响

7 局限性与未来工作

本文面向边缘智能场景下大规模异构模型库的低成本部署与更新问题, 针对资源受限条件下的动态任务需求、更新开销与服务响应之间的权衡, 以及非平稳请求分布下联合部署与更新的自适应难点, 提出了联合部署与更新策略 PACE. PACE 基于参数块共享并结合信息年龄刻画模型新鲜度与性能衰减, 将切换、通信、更新、推理与精度损失等开销统一纳入部署成本目标, 并通过连续化处理降低在线联合优化的求解难度. 实验结果表明, 与多种基线方法相比, PACE 能够有效降低部署成本.

尽管如此, 本文仍存在以下局限性: (1) 本文的优化目标为降低多请求分布情况下的部署成本, 但部署成本中各组成部分的定义仍需进一步深化. 例如, 模型缺失带来的通信成本受网络状况影响, 模型切换成本则可能受 GPU 加载性能的制约. (2) 本文虽考虑了模型间参数块的共享特性, 但在模型更新时仅对当前模型进行更新, 未同时考虑共享同一参数块的其他模型的同步更新问题. (3) 本文对边缘部署场景进行了简化, 假设所有边缘服务器具有相同的计算与存储能力. 然而在实际应用中, 不同边缘服务器的资源使用情况存在差异, 可用于部署与计算的资源亦不尽相同. 此外, 当前实验主要基于 ResNet、VGG16 和 MobileNet 等视觉模型展开验证, 其在更大规模模型场景下的适用性仍有待进一步评估. 同时, 本文实验中的服务请求通过 Zipf 分布进行建模, 虽能较好地刻画边缘服务中常见的热点访问与长尾特征, 但整体仍属于受控仿真环境. 在未来工作中, 我们将结合更大规模模型及更接近真实边缘应用场景的公开请求数据, 对本文方法进行进一步验证, 以更全面地评估其在实际部署环境中的适用性与实用价值.

References

[1] Yu SH, Yi MJ, Wu Z, Shen FR, Zhao J. Neuromorphic computing: From spiking neural network to edge deployment. Ruan Jian Xue Bao/Journal of Software, 2025, 36(4): 1758–1795 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7298.htm [doi: 10.13328/j.cnki.jos.007298]

[2] Lin PY, Shi ZC, Xiao Z, Chen C, Li KL. Latency-driven model placement for efficient edge intelligence service. IEEE Trans. on Services Computing, 2022, 15(2): 591–601. [doi: 10.1109/TSC.2021.3109094]

[3] Chen JS, Ran XK. Deep learning with edge computing: A review. Proc. of the IEEE, 2019, 107(8): 1655–1674. [doi: 10.1109/JPROC.2019.2921977]

[4] Niu SC, Wu JX, Zhang YF, Chen YF, Zheng SJ, Zhao PL, Tan MK. Efficient test-time model adaptation without forgetting. In: Proc. of the 39th Int'l Conf. on Machine Learning. Baltimore: PMLR, 2022. 16888–16905.

[5] Tajbakhsh N, Shin JY, Gurudu SR, Hurst RT, Kendall CB, Gotway MB, Liang JM. Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Trans. on Medical Imaging, 2016, 35(5): 1299–1312. [doi: 10.1109/TMI.2016.2535302]

[6] Bai LY, Cao JN, Zhang MJ, Li B. Collaborative edge intelligence for autonomous vehicles: Opportunities and challenges. IEEE Network, 2025, 39(2): 52–60. [doi: 10.1109/MNET.2024.3520166]

[7] Liu HT, Cao GH. Deep learning video analytics through online learning based edge computing. IEEE Trans. on Wireless Communications, 2022, 21(10): 8193–8204. [doi: 10.1109/TWC.2022.3164598]

[8] Wang P, Jiang H, Liu Y, Zhao ZL, Zhou K, Huang ZH. Beyond belady to attain a seemingly unattainable byte miss ratio for content delivery networks. IEEE Trans. on Parallel and Distributed Systems, 2024, 35(11): 1949–1963. [doi: 10.1109/TPDS.2024.3452096]

[9] Zhang JS, Yang X, Ming R, Luo HB, Wang XD, Piran MJ. A time-driven UAV deployment and content caching approach for multi-UAV-enabled consumer electronics network. IEEE Trans. on Consumer Electronics, 2025, 71(2): 3941–3951. [doi: 10.1109/TCE.2025.3560129]

[10] Sevilla J, Heim L, Ho A, Besiroglu T, Hobbhahn M, Villalobos P. Compute trends across three eras of machine learning. In: Proc. of the 2022 Int'l Joint Conf. on Neural Networks (IJCNN). Padua: IEEE, 2022. 1–8. [doi: 10.1109/IJCNN55064.2022.9891914]

[11] Hu EJ, Shen YL, Wallis P, Zhu AZ, Li YZ, Wang SA, Wang L, Chen WZ. LoRA: Low-rank adaptation of large language models. In: Proc. of the 10th Int'l Conf. on Learning Representations. OpenReview.net, 2022.

[12] Qu GQ, Lin Z, Liu FM, Chen XH, Huang KB. TrimCaching: Parameter-sharing AI model caching in wireless edge networks. In: Proc. of the 44th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). Jersey City: IEEE, 2024. 36–46. [doi: 10.1109/ICDCS60910.2024.00013]

[13] Padmanabhan A, Agarwal N, Iyer AP, Ananthanarayanan G, Shu YC, Karianakis N, Xu GH, Netravali R. Gemel: Model merging for memory-efficient, real-time video analytics at the edge. In: Proc. of the 2023 USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2023. 973–994.

[14] Zhou FY, Jin LP, Dong J. Review of convolutional neural network. Chinese Journal of Computers, 2017, 40(6): 1229–1251 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2017.01229]

[15] Huang HB, Liang JB, Min GY. Joint DNN model deployment, selection, and configuration for heterogeneous inference services toward edge intelligence. IEEE Trans. on Mobile Computing, 2025, 24(11): 12726–12741. [doi: 10.1109/TMC.2025.3586793]

[16] Cheng ZP, Xia XY, Wang H, Liwang M, Chen N, Fan XW, Wang XB. Privacy-aware joint DNN model deployment and partitioning optimization for collaborative edge inference services. IEEE Trans. on Services Computing, 2025, 18(5): 3079–3092. [doi: 10.1109/TSC.2025.3607117]

[17] Peng ZY, Qiu Y, Wang GC. Model caching and application offloading for mobile edge intelligence network with learning-and-optimization approach. IEEE Trans. on Services Computing, 2025, 18(5): 3022–3037. [doi: 10.1109/TSC.2025.3592381]

[18] Zhou LZ, Xu ZC, Xia QF, Xu Z, Ren WH, Qi WB, Ma JJ, Yan S, Yang Y. Chasing common knowledge: Joint large model selection and pulling in MEC with parameter sharing. IEEE Trans. on Parallel and Distributed Systems, 2025, 36(3): 437–454. [doi: 10.1109/TPDS.2025.3527649]

[19] Ai X, Liang WF, Liu CY. Joint optimization of model retraining and inference services in DT-assisted edge computing. IEEE Trans. on Networking, 2026, 34: 1804–1819. [doi: 10.1109/TON.2025.3632228]

[20] Yun S, Kim D, Lee S, Lee J. i-CU: Intelligent cache replacement and content update for data freshness in cloud-edge networks. IEEE Trans. on Mobile Computing, 2025, 24(12): 12742–12755. [doi: 10.1109/TMC.2025.3589609]

[21] Mhaisen N, Iosifidis G, Leith D. Online caching with no regret: Optimistic learning via recommendations. IEEE Trans. on Mobile Computing, 2024, 23(5): 5949–5965. [doi: 10.1109/TMC.2023.3317943]

[22] Quan GC, Eryilmaz A, Shroff NB. Optimal edge caching for individualized demand dynamics. IEEE/ACM Trans. on Networking, 2024,

32(4): 2826–2841. [doi: 10.1109/TNET.2024.3369611]
[23] Abolhassani B, Tadrous J, Eryilmaz A, Yüksel S. Optimal push and pull-based edge caching for dynamic content. IEEE/ACM Trans. on Networking, 2024, 32(4): 2765–2777. [doi: 10.1109/TNET.2024.3352029]
[24] Quan GC, Eryilmaz A, Shroff NB. Minimizing edge caching service costs through regret-optimal online learning. IEEE/ACM Trans. on Networking, 2024, 32(5): 4349–4364. [doi: 10.1109/TNET.2024.3420758]
[25] Zhou SJ, Wang Z, Hu CH, Mao YN, Yan HP, Zhang SH, Wu C, Zhu WW. Caching in dynamic environments: A near-optimal online learning approach. IEEE Trans. on Multimedia, 2023, 25: 792–804. [doi: 10.1109/TMM.2021.3132156]
[26] Fan S, Hou IH, Mai VS. Dynamic regret of randomized online service caching in edge computing. In: Proc. of the 2023 IEEE Conf. on Computer Communications. New York City: IEEE, 2023. 1–10. [doi: 10.1109/INFOCOM53939.2023.10229044]
[27] Blasco P, Gündüz D. Learning-based optimization of cache content in a small cell base station. In: Proc. of the 2014 IEEE Int'l Conf. on Communications (ICC). Sydney: IEEE, 2014. 1897–1903. [doi: 10.1109/ICC.2014.6883600]
[28] Ren WH, Xu ZC, Liang WF, Dai HP, Rana OF, Zhou P, Xia QF, Ren HZ, Li MC, Wu GW. Learning-driven service caching in MEC networks with bursty data traffic and uncertain delays. Computer Networks, 2024, 250: 110575. [doi: 10.1016/j.comnet.2024.110575]
[29] Narasimha D, Kalathil D, Shakkottai S. Meta-learning for fast adaption in caching networks. IEEE Trans. on Networking, 2025, 33(1): 65–77. [doi: 10.1109/TNET.2024.3478853]
[30] Yates RD, Ciblat P, Yener A, Wigger M. Age-optimal constrained cache updating. In: Proc. of the 2017 IEEE Int'l Symp. on Information Theory (ISIT). Aachen: IEEE, 2017. 141–145. [doi: 10.1109/ISIT.2017.8006506]
[31] Charbonnier P, Blanc-Feraud L, Aubert G, Barlaud M. Deterministic edge-preserving regularization in computed imaging. IEEE Trans. on Image Processing, 1997, 6(2): 298–311. [doi: 10.1109/83.551699]
[32] Viallard P, Germain P, Habrard A, Morvant E. A general framework for the practical disintegration of PAC-Bayesian bounds. Machine Learning, 2024, 113(2): 519–604. [doi: 10.1007/s10994-023-06391-0]
[33] Blei DM, Kucukelbir A, McAuliffe JD. Variational inference: A review for statisticians. Journal of the American Statistical Association, 2017, 112(518): 859–877. [doi: 10.1080/01621459.2017.1285773]
[34] Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proc. of the 34th Int'l Conf. on Machine Learning (ICML). PMLR, 2017. 1126–1135.
[35] Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proc. of the 35th Int'l Conf. on Machine Learning (ICML). PMLR, 2018. 1587–1596.
[36] O'Neil EJ, O'Neil PE, Weikum G. The LRU-K page replacement algorithm for database disk buffering. In: Proc. of the 1993 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 1993. 297–306. [doi: 10.1145/170035.170081]
[37] Cao P, Irani S. Cost-aware WWW proxy caching algorithms. In: Proc. of the 1997 USENIX Symp. on Internet Technologies and Systems (USITS 1997). USENIX Association, 1997. 193–206.
[38] Psounis K, Prabhakar B. Efficient randomized Web-cache replacement schemes using samples from past eviction times. IEEE/ACM Trans. on Networking, 2002, 10(4): 441–455. [doi: 10.1109/TNET.2002.1031757]

附中文参考文献

[1] 俞诗航, 易梦军, 吴洲, 申富饶, 赵健. 神经形态计算: 从脉冲神经网络到边缘部署. 软件学报, 2025, 36(4): 1758–1795. http://www.jos.org.cn/1000-9825/7298.htm [doi: 10.13328/j.cnki.jos.007298]
[14] 周飞燕, 金林鹏, 董军. 卷积神经网络研究综述. 计算机学报, 2017, 40(6): 1229–1251. [doi: 10.11897/SP.J.1016.2017.01229]

作者简介

滕自怡, 博士生, 主要研究领域为边缘智能, 云边协同, 模型部署, 推理优化.
方娟, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为高性能计算, 智能计算, 边缘智能, 云边协同, 推理优化.
刘雅祺, 博士生, 主要研究领域为边缘智能, 云边协同, 模型部署, 推理优化.
张梦媛, 博士生, 主要研究领域为边缘智能, 云边协同, 模型部署, 推理优化.
陈慧杰, 博士, 讲师, CCF 专业会员, 主要研究领域为智能物联网, 泛在感知, 边缘计算, 物联网系统安全, 隐私保护.