arXiv:2305.01519v1 [cs.LG] 1 May 2023

BCEdge：边缘平台上采用自适应批处理的 SLO 感知型 DNN 推理服务

Ziyang Zhang，*IEEE* 学生会员，Huan Li，*IEEE* 高级会员，Yang Zhao，*IEEE* 高级会员，Changyao Lin 和 Jie Liu，*IEEE* 会士

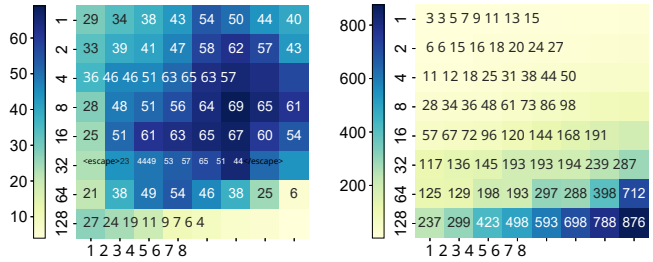
摘要——**随着深度神经网络（DNN）被应用于各种边缘智能应用，边缘推理平台同时实现高吞吐量和低延迟至关重要。此类运行多个深度神经网络模型的边缘平台给调度器设计带来了新的挑战。首先，每个请求可能具有不同的服务等级目标（SLO），以提高服务质量（QoS）。其次，边缘平台应能够高效调度多个异构 DNN 模型，从而提高系统利用率。为实现这两个目标，本文提出了 BCEdge，这是一种新颖的基于学习的调度框架，采用自适应批处理，并在边缘平台上并发执行 DNN 推理服务。我们定义了一种效用函数，用于评估吞吐量与延迟之间的权衡。BCEdge 中的调度器利用基于最大熵的深度强化学习（DRL），通过以下两种方式最大化效用：1）联合优化批大小；2）自动确定并发模型的数量。在不同边缘平台上实现的原型系统表明，与最先进的解决方案相比，所提出的 BCEdge 在满足 SLO 的同时，平均可将效用提高至多 37.6%。**

索引词—**边缘计算、推理服务、调度、强化学习、服务等级目标（服务级别目标）。**

I. 引言

Model 部署在云服务器上的推理服务系统通常会向用户提供多个经过训练的深度神经网络（DNN）。这些系统通常采用多租户架构，即为每个深度神经网络模型托管一个或多个模型实例，以服务于多个推理应用，同时更充分地利用服务器丰富的计算资源。例如，NVIDIA Ampere 架构中的多实例 GPU（MIG）技术能够将单个 NVIDIA A100 GPU 划分为最多七个可独立运行且能够并发执行的 GPU 实例。通过这种方式，GPU 在保证服务质量（QoS）的同时，利用率最高可达 7×。此外，单个推理请求往往会导致资源利用效率低下。因此，以往的研究采用 [1], [2], [3] 批量请求，以更好地发挥 GPU 的并行性。批处理是指

本研究部分受中国国家重点研发计划资助，资助编号 2021ZD0110905；并受中国黑龙江省智慧农业决策平台关键技术研究与应用开放竞赛项目资助，资助编号 2021ZXJ05A03。Ziyang Zhang 就职于哈尔滨工业大学，地址：中国黑龙江省哈尔滨市 150006（电子邮箱：zhangzy@stu.hit.edu.cn）。Huan Li 就职于哈尔滨工业大学（深圳），地址：中国广东省深圳市 518071（电子邮箱：huanli@hit.edu.cn）。Yang Zhao 就职于哈尔滨工业大学（深圳），地址：中国广东省深圳市 518071（电子邮箱：yang.zhao@hit.edu.cn）。ChangyaoLiniswiththeHarbinInstituteofTechnology,Harbin, Heilongjiang 150006 China（电子邮箱：lincy@stu.hit.edu.cn）。Jie Liu 就职于哈尔滨工业大学（深圳），地址：中国广东省深圳市 518071（电子邮箱：jieliu@hit.edu.cn）。



(a) 吞吐量（请求/秒）　　(b) 延迟（毫秒）

图 1. 批处理和并发推理对 (a) 系统吞吐量和 (b) 端到端延迟的影响。吞吐量以每秒请求数 (rps) 衡量。x 轴表示并发模型的数量，y 轴表示批大小。我们在配备 8 吉字节内存的 NVIDIA Xavier NX 边缘平台上使用 YOLO-v5 [9] 。

将到达的请求在给定的时间窗口内聚合成一个批次，DNN 服务系统再于特定时间处理整个批次，从而提高吞吐量（例如，每秒请求数，rps）。在批处理系统中，可以通过增大批大小来提高吞吐量。批次中的请求越多，等待处理的时间就越长，因此延迟不可避免地会增加。

不断增强的计算能力和内存能力为在边缘加速器（例如图形处理器（GPU）、张量处理器（TPU）和视觉处理器（VPU）等）上部署模型推理系统创造了新的机遇。这种新兴的计算范式为具有低延迟要求的边缘智能应用 [4] 提供了保障，包括自动驾驶 [5],中的目标检测、智能手机 [6], 中的推荐系统，以及可穿戴设备 [7], 中的元宇宙应用等。另一方面，针对深度神经网络模型的各种轻量化技术 [8] （如剪枝、压缩、量化、知识蒸馏等）使得在边缘端能够对多个模型实例进行批处理和并发推理。

为了更好地了解批处理和并发推理对性能的影响，我们使用 YOLO-v5 [9] 在 NVIDIA Xavier NX 边缘平台上进行了实验研究。由于边缘平台存在资源约束，我们采用 TensorRT [10] 来加速原始深度神经网络模型。图 1 展示了不同批大小和并发模型数量下的吞吐量与延迟。我们发现了以下启发本研究的关键现象：批大小和并发模型数量都会影响吞吐量和延迟，但增大批大小或模型数量并不总是更好。图1 显示，较高的吞吐量

在适中的批大小和并发模型数量下，延迟较低。由于模型干扰导致的资源争用，过大的批大小和过多的并发模型会显著降低吞吐量、增加延迟，甚至导致内存溢出，尤其是在批大小和并发模型数量都较大时，例如批大小为 128、模型数量为 8。因此，DNN 服务系统中的调度器至关重要，必须做到：1）权衡吞吐量和延迟，以实现最佳性能；2）准确预测模型之间的干扰。

设计此类基于 GPU 的推理服务器，必须应对不同于面向批处理和并发处理的挑战。首先，推理请求需要达到服务级别目标（SLO），以低延迟提供服务质量（QoS）。其次，边缘平台通常会部署多个异构 DNN 模型，以提升吞吐量和资源利用率。因此，设计高效的调度器，对于在边缘平台上优化批处理和并发推理至关重要。传统的基于启发式的方法在多目标优化方面效率低下 [11]。相比之下，深度强化学习（DRL）将深度学习强大的表征能力与强化学习的自适应特性相结合，能够高效地解决上述问题。因此，我们利用 DRL 将多目标（即吞吐量和延迟）问题转化为批大小和并发模型数量的调度问题。

受上述观察启发，我们提出 BCEdge，这是一种面向 SLO 感知型 DNN 推理服务的可学习、自适应多租户调度框架。BCEdge 旨在通过同时调整批大小和并发模型数量，在吞吐量与延迟之间进行权衡，从而自动寻找全局最优解。与以往采用一维（例如批大小）搜索的工作不同，这里的调度搜索空间由批大小和并发模型数量构成，是二维的。批处理-并发调度能够显著提升满足 SLO 的吞吐量。对于每个 DNN 模型，其计算特性都会经过测量并登记到 BCEdge 中。基于各 DNN 模型的性能分析信息，BCEdge 中基于最大熵强化学习的调度器会自动调整批大小和并发模型数量，同时维持 SLO。此外，BCEdge 采用一种轻量级的、基于神经网络（NN）的干扰预测模型，以降低并发推理的影响。

表I 对我们的工作与相关深度神经网络服务框架进行了总结比较。此前的研究都能够在运行时自动或手动调整批大小，以实现更高的吞吐量。随着越来越多的 DNN 推理服务被整合到边缘 GPU 服务器上，尽管一些先前研究支持多个异构 DNN 模型进行多租户部署，但这些工作并未支持在 DNN 服务系统中调度同一模型的多个实例；随着计算资源不断增加，这一能力对于充分利用边缘服务器变得愈发重要。对于多租户之间的模型间资源竞争以及带有服务级别目标（SLO）的请求，准确预测干扰并保证服务级别目标可以

表 I 与既有工作的比较

服务框架	自适应批处理	并发实例	多模型	干扰预测	SLO 感知
TF-Serving [2]	✓	✗	✓	✓	✗
Triton [3]	✓	✓	✓	✗	✗
DeepRT [12]	✓	✗	✓	✗	✓
Clipper [1]	✓	✗	✓	✗	✓
Prema [13]	✓	✗	✓	✗	✗
DVABatch [14]	✓	✓	✓	✗	✗
BCEdge（我们的方法）	✓	✓	✓	✓	✓

帮助调度器更好地自动调整批大小和并发模型数量，从而实现适度的系统吞吐量并降低延迟。只有 TF-Serving [2] 考虑了模型干扰，并通过对冲备份请求缓解由请求间或模型间干扰导致的延迟峰值。另一方面，除 Clipper [1]和 DeepRT [12]，之外，其他先前工作均未考虑具有严格时间约束（即服务级别目标）的请求。重要的是，我们的研究应对了所有这些挑战：并发执行多个实例、保证服务级别目标，并预测多模型之间可能出现的干扰。

我们在配备三种异构边缘 GPU 的边缘平台上，使用表 IV 中的六个深度神经网络模型，对所提出的深度神经网络推理框架进行了评估。这些模型涵盖计算机视觉和自然语言处理应用，包括目标检测、图像分类以及语音识别。评估结果表明，与最先进的解决方案相比，采用批处理和并发模型实例的所提出调度技术能够将 SLO 约束下的权衡改善 37.6%。本文的主要贡献如下：

- 通过一项基于真实场景中边缘平台上的深度神经网络模型批处理和并发推理的激励性案例研究，我们证明了吞吐量与延迟之间的权衡应同时利用批处理和并发推理。

- 我们提出了 BCEdge，这是一种用于边缘平台上推理服务的可学习调度框架，支持自适应批处理和并发模型实例。BCEdge 中的调度器利用最大熵强化学习，自动调整批大小和并发模型数量，以权衡吞吐量与延迟。

- 为实现准确的性能预测，BCEdge 中具有可忽略开销的轻量级基于神经网络的预测模型降低了模型之间相互干扰对深度神经网络并发推理的影响。

本文其余部分的组织结构如下：第II节介绍相关工作。第III节描述系统模型和问题形式化。第IV 节详细阐述我们的框架设计。第V 节报告实验结果。第VI节总结全文。

II. 相关工作

A. 模型级深度神经网络推理服务

以往的工作将深度神经网络模型视为不可分割的整体，并提出了一系列边缘推理服务框架，以提供深度神经网络推理服务 [1], [2], [12],

[14], [15], [16], [17], [18], [19], [20]. Clipper [1], TensorFlow- Serving [2], MArK [15], DeepRT [12], 和 BATCH [20] 采用传统的自适应批处理方法，通过使用时间窗口来高效执行 DNN 推理。这些现有框架均未提供模型实例的并发运行功能，以进一步提高吞吐量。此外，还有一些先前的研究致力于 SLO 感知型 DNN 推理服务。Gpulet [16]利用计算资源的时空共享，为多个异构 DNN 模型提供满足 SLO 约束的服务。Clockwork [19] 利用可预测的执行时间来实现严格的请求级 SLO。INFaaS [18] 通过选择适当的模型变体来降低成本、提高吞吐量并减少 SLO 违规。PSLO [17] 是一种面向边缘平台、基于最小平均预期延迟的抢占式 SLO 感知型调度器，旨在权衡响应时间、系统吞吐量和 SLO。与上述工作不同，我们专注于降低由多模型干扰引起的 SLO 违约率。此外，一些边缘推理框架分别涉及隐私保护 [21], [22] 和边缘云协同 [23],[24], 。这些工作与 BCEdge 正交，可缓解隐私和资源约束问题。

B. 算子级*DNN*推理服务

已有一些研究致力于优化深度神经网络模型的算子调度，以提升模型服务质量 [13], [25], [26], [27]。REEF [25] 采用基于动态内核填充的并行机制来提升整体吞吐量。VELTAIR [27] 提出了自适应的算子级编译与调度方法，以确保资源使用效率，并减少干扰导致的性能损失，从而支持多租户DNN服务。PREMA [13] 是一种面向可抢占神经处理单元的预测性多任务调度算法，旨在满足高吞吐量需求。Abacus [26]利用重叠感知型延迟预测和对重叠DNN算子的确定性调度，在维持多租户DNN服务服务质量的同时提升吞吐量。由于BCEdge通过批处理和并发推理，利用边缘平台上加速器的计算能力，因此这些工作与BCEdge也相互正交，可以结合使用，以实现更高的吞吐量和更低的延迟。

C. 边缘平台上的多租户调度

由于边缘平台上的资源受限，与云计算相比，多租户调度更具挑战性。TVW-RL [28] 基于深度强化学习（DRL）方法，利用时变工作负载的各种时间资源使用模式，以提高真实生产轨迹中的利用率。同样，KaiS [29], A3C-R2N2 [11],MILP [30], A3C-DO [31], 和 MFRL [32] 在边缘云集群中提出了不同的基于多智能体强化学习的调度策略，以优化吞吐量、延迟、能耗、成本等。MCDS [33]使用基于树的搜索策略和基于DNN的预测模型，在边缘云测试平台中优化服务质量。与 MILP [30], 类似，DeEdge [34]提出了 D-Deads，这是一种在边缘计算中带有任务截止期限的分布式贪心调度算法，可最大化在最大限度降低延迟的同时提高吞吐量。需要注意的是，上述工作仅逐一调度单个任务，忽略了批处理和并发推理所带来的优势。受这些工作的启发，BCEdge 也可以扩展为一种边缘-云协同推理框架，以优化特定目标。

III. 系统模型与问题形式化

本节首先建立系统模型，包括请求模型、调度模型、计算模型和网络模型。随后，我们给出优化问题的形式化描述，以展示吞吐量与延迟之间的权衡。

A. 系统模型

1) 请求模型：我们假设物联网设备（例如摄像头、无人机、智能手机等）共享边缘平台的计算资源。在任务调度之前，物联网设备会生成一系列推理请求，这些请求具有不同的深度神经网络模型类型 m_t^i、输入类型 d_t^i （即图像或文本）、输入形状 d_s^i 和服务级别目标 SLO_i。因此，第 i 个请求r_i可表示为 $r_i = \{m_t^i, d_t^i, d_s^i, SLO_i\}$。请注意，请求以泊松分布随机地在线到达 BCEdge。BCEdge 为每个模型维护一个请求队列，并支持动态批处理，将多个使用同一模型的推理请求聚合到相应的请求队列 $seq_b = \{r_1, r_2, \ldots, r_b\}$中，其中 b 为深度神经网络模型的批大小。同时，BCEdge 为每个模型构建多个实例 $m_c(c = 1, 2, \ldots, n)$（即并发实例）。这对于配备 GPU 的边缘平台至关重要，因为批处理和并发推理能够有效提高吞吐量。BCEdge 后端的模型库从请求队列中执行 DNN推理，并返回预测结果 $O_b = \{o_1, o_2, \ldots, o_b\}$。

2）调度模型：由于每个请求的服务级别目标各不相同，固定的调度时隙并不适用。另一方面，为每个请求单独指定调度时隙并不可行，因为这会显著增加系统开销。因此，我们将第 i 个调度时隙 t_i 设为一批请求的服务级别目标$_i$ 总和与并发模型数量的比值，记为

$$t_i = \sum_{i=1}^{b} SLO_i / m_c \tag{1}$$

这样，BCEdge 能够保证每个请求的服务级别目标，并通过批处理和并发推理提供高效的推理服务。此外，BCEdge 在完成当前调度后立即开始下一次调度，以减少 GPU 空闲。

3）计算与网络模型：端到端延迟涉及物联网设备与边缘平台之间的通信，以及模型推理时间，后者由以下部分组成：

- 请求传输时间t_{it}：物联网设备通过网络向边缘平台发送第 i 个推理请求（例如，图像、文本等）所需的时间，该时间取决于通信带宽和输入数据的大小。

- 请求序列化时间 $t_{是}$：在边缘平台上，将具有相同模型的多个推理请求聚合到单个请求队列中，以便对模型实例进行批处理和并发推理所需的时间。
- 请求排队时间 t_{iw}：请求在请求队列中等待调度的时间，这与批大小和并发模型的数量有关。
- ***DNN*推理时间** t_{im}：边缘平台执行模型推理所需的时间。推理完成后，当前请求会从队列中移除。
- 结果传输时间 t_{io}：边缘平台通过网络向物联网设备发送第 i 个推理请求所需的时间，这与网络带宽有关，而与结果大小无关（通常可忽略不计）。

因此，总延迟 t_r^i 可表示为：

$$t_r^i = t_t^i + t_s^i + t_w^i + t_m^i + t_o^i \tag{2}$$

B. 问题形式化

我们的目标是在保证服务级别目标的同时，自动探索批大小和并发模型数量的可行取值范围，从而协同优化每个深度神经网络模型的吞吐量和延迟。受名为 Pollux 的协同自适应调度器启发 [35], 我们提出了一个效用函数 U 来在式（3）中评估吞吐量与延迟之间的权衡。

$$U = \log(T_{t_i}(b, m_c)/\frac{L_{t_i}(b, m_c)}{(\sum_{j=1}^{b} r_j)/m_c}) \tag{3}$$

其中，b 为批大小，m_c 为并发模型的数量。第 i 个调度时隙 t_i中的吞吐量可表示为 $T_{t_i}(b,m_c)$，$L_{t_i}(b, m_c)$ 表示第 i 个请求的实际延迟。$(\sum^{b}{}_{j=1}\, r_j)/m_c$ 表示批量请求的 SLO 总和与并发模型数量之比。值得注意的是，$L^{t_i}(b,m_c)\,(\sum^{b}{}_{j=1}\, r_j)/m_c \in (0, 1]$ 在确保实时性能的同时，尽可能避免请求调度失败。

调度器在对请求进行批处理并发执行时，必须考虑边缘平台的内存容量 M_i 和服务级别目标约束 SLO_i。因此，满足上述要求的优化目标可表述为

$$\begin{aligned} &\min. U \\ &\text{s.t.}\, m_i \le M_i \\ &L_i \le SLO_i \end{aligned} \tag{4}$$

其中，m_i 是第 i 个请求实际使用的内存，L_i 是第 i 个请求的端到端延迟。

表II 主要提供符号定义及其对应说明。

表 II 符号表及其描述

符号	描述
m_t^i	第 i 个请求的深度神经网络模型类型
d_t^i	第 i 个请求所请求的输入类型
d_s^i	第 i 个请求所需的输入形状
SLO_i	第 i 个请求的服务等级目标
seq_b	批大小为 b 的请求队列
m_c	并发模型的数量
O_b	推理结果
t_i	第 i 个调度时隙
t_r^i	第 i 个请求的端到端延迟
u	平均资源利用率
s_i	第 i 个队列的批处理时隙
$T_{t_i}(b,\ m_c)$	吞吐量
$L_{t_i}(b,\ m_c)$	端到端延迟
U	效用函数

IV. BCEdge 设计

A. 系统概述

BCEdge 旨在为多模型 DNN 推理服务设计一种调度框架，通过为每个到达的推理请求分配适当的批大小和并发模型数量，同时维持服务级别目标。

为此，带有服务级别目标要求的深度神经网络推理请求调度服务级别目标要求必须考虑两个方面：批处理和并发模型实例。不同于以往仅考虑这两个维度中部分内容的工作 [1], [12], ，我们提出一种调度器，全面探索这两个维度，以找到最有效的调度方案。

图 2 展示了我们提出的调度框架 BCEdge 的整体架构。该框架由基于学习的调度器（第IV-B节）、动态批处理模块（第 IV-C节）、并发实例模块（第 IV-D节）、性能分析器（第IV-E节）以及 SLO感知干扰预测器（第 IV-F节）组成。BCEdge 首先 ❶ 为每个深度神经网络模型维护一个请求队列。物联网设备生成的、针对不同深度神经网络模型的请求会被合并，并发送到相应的请求队列。性能分析器 ❷ 定期收集每个深度神经网络模型的信息（例如，一对批大小和并发模型数量对应的利用率、SLO、系统吞吐量和端到端延迟）。与此同时，SLO感知干扰预测器 ❸ 分析并发模型实例可能造成的干扰开销，从而指导调度器做出更稳健的决策。随后，基于学习的调度器 ❹ 利用性能分析信息找出最佳批大小和并发模型数量，并分别将其反馈给动态批处理模块和并发实例模块。后端中的执行器最终 ❺ 在边缘平台上使用批处理和指定数量的并发模型执行深度神经网络推理服务。

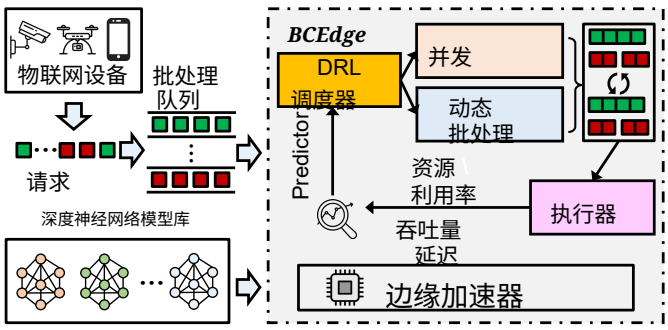


图 2. DNN 服务调度框架概览。

B. 基于学习的调度器

搜索空间挑战：基于学习的调度器是 BCEdge 的关键组件。需要注意的是，与之前的工作（例如 TF-Serving [2], Clipper [1], 和 DeepRT [12]）相比，BCEdge 中的调度更加复杂，因为它同时涉及批处理和并发推理。BCEdge 的二维调度空间（批大小和并发模型数量）所面临的挑战在于，调度决策受到多个相互依赖的变量影响。具体而言，最佳批大小和并发模型数量取决于不同深度神经网络模型的计算需求、输入特性、SLO 约束以及边缘平台可用的计算资源。因此，最优的权衡配置应位于由这两个维度构建的搜索空间中的最佳点，这会产生一个巨大的搜索空间。

为此，我们针对 BCEdge 定制了基于学习的调度算法。与传统启发式方法相比，深度强化学习（DRL）在处理复杂的策略决策方面具有显著优势，适用于高维动作空间。因此，为了权衡吞吐量和延迟，我们设计了一种新颖的基于 DRL 的调度器，通过批处理和并发推理实现高效的深度神经网络推理服务。由于批大小和并发模型数量均为离散值，我们基于离散软演员-评论家 [36]框架，提出了一种具有最大熵特性的可学习在线调度算法。与传统 DRL 方法相比，该算法在最大化奖励的同时，也最大化访问状态的熵。引入熵使我们提出的调度算法具备以下优势：

- 使 DRL 中的智能体能够学习更多接近最优的动作，从而加速训练（即输出为策略分布）；相比之下，基于确定性策略的 DRL 输出的是动作。
- 使智能体具备更强的环境探索能力，避免陷入局部最优。
- 使系统更加稳健。

现在，我们重点介绍 BCEdge 中的调度器如何为每个推理请求确定批大小和并发模型数量，以优化公式（3）中的权衡。具体细节如下：

1）马尔可夫决策过程建模：首先，我们将推理请求的批处理和并发调度建模为马尔可夫决策过程（MDP）。该过程可表示为五元组（$\mathcal{S}$、 $\mathcal{A}$、 π、 p、 r）：

- 状态： $\mathcal{S}$ 是离散状态空间。在每个调度时隙 t_i，深度强化学习中的智能体构建状态 s_t（$s_t \in \mathcal{S}$），以定期收集请求信息和边缘平台的资源利用率。 s_t 由五个部分组成：（I）深度神经网络模型类型 m_t^i。（II）输入类型 d_t^i和输入形状 d_s^i。（III）每个请求的服务级别目标。（IV）边缘平台的可用计算资源 m_i。（V）请求队列的信息 seq_b。
- 动作： $\mathcal{A}$ 是离散动作空间。深度强化学习中智能体的动作是找到最佳批大小 b 和并发模型数量 m_c。调度时隙 $a_t(a_t \in \mathcal{A})$处的动作调度时隙 t_i 可表示为 $a_t = (b, m_c)$。例如，如果一个深度神经网络模型有 $\mathcal{M}$ 种可选批大小和 $\mathcal{N}$ 种可选并发模型数量，则离散动作空间的大小 $\mathcal{A}$ 为 $\mathcal{M} \times \mathcal{N}$。
- 策略：策略 $\pi(a_t \mid s_t)$ 是一个函数，智能体根据时间戳 t 时的环境状态 s_t 来决定下一个动作 a_t。在基于最大熵的深度强化学习算法中，我们同时最大化奖励和访问状态的熵。最优策略 π^* 表示如下：

$$\pi^* = \text{argmax}_\pi \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[\gamma^t r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))] \tag{5}$$

其中， $\gamma \in [0,1]$ 是折扣因子， ρ_π 表示由策略π生成的动作轨迹分布， α 是温度参数，用于表示奖励和熵的相对重要性。 $\mathcal{H}(\pi(\cdot \mid s_t)) = -\log \pi(\cdot \mid s_t)$，即状态 s_t下的熵。

- 状态转移概率： $p(s_t' \mid s_t, a_t)$ 是状态转移概率，表示在时间戳 t 的当前状态 s_t 中执行动作a_t 后转移到下一个状态 s_t' 的概率，满足$\sum_{s'\in S} p(s_t' \mid s_t, a_t) = 1$。
- 奖励： $r: \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ 是奖励函数。DRL 中的智能体旨在最大化累积期望奖励 $\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$，其中 r_t 是智能体在每个调度时隙 t_i选择批大小和并发模型数量时的即时奖励。由于我们的目标是最大化吞吐量与延迟之间的权衡，我们将公式 (3) 中的目标迁移到奖励函数中：

$$r_t = U \tag{6}$$

通过这种方式，传统调度问题被转化为深度强化学习中的奖励最大化问题。我们进一步采用高效的基于学习的算法来降低复杂度。

2) 基于最大熵深度强化学习的调度算法：我们利用软策略迭代 [37] 来最大化奖励和熵。具体而言，软策略迭代由策略评估和策略改进组成，并在训练过程中交替进行。

- 软Q函数（评论家网络）：我们首先在策略评估步骤中计算软Q值 $Q(s_t, a_t)$。软Q函数表示如下 [37]:

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p}[V(\mathbf{s}_{t+1})] \tag{7}$$

其中， $\mathcal{T}^\pi$ 是修改后的贝尔曼备份算子。
离散状态空间 s_t 中策略 π 对应的软状态价值函数 $V(s_t)$为：

$$V(s_t) := \pi(s_t)^T [Q(s_t) - \alpha\log(\pi(s_t))] \tag{8}$$

我们通过最小化软贝尔曼残差来训练式（7）中的软Q值，损失函数为：

$$J_Q(\theta) = E_{(s_t,a_t)\sim D}[\frac{1}{2}(Q_\theta(s_t,a_t) - (r(s_t,a_t) + \gamma E_{s_{t+1}\sim p(s_t,a_t)}[V_{\bar{\theta}}(s_{t+1})]))^2] \tag{9}$$

• 策略（***Actor Network***）：使用策略改进来更新策略网络，表示如下：

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\cdot \mid \mathbf{s}_t) \| \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right) \tag{10}$$

其中，D_{KL} 表示 Kullback-Leible（KL）散度，$Z^{\pi_{\text{old}}}$ 表示配分函数。

我们最小化式（11）中的 KL 散度，以更新策略网络的参数：

$$J_\pi(\phi) = E_{s_t \sim D_{KL}} \left[\pi_t(s_t)^T \left[\alpha \log\left(\pi_\phi(s_t)\right) - Q_\theta(s_t) \right] \right] \tag{11}$$

•温度参数：我们根据 α 式（5）中的 [37]，自动调整温度参数：

$$J(\alpha) = E_{a_t \sim \pi_t} \left[-\alpha \left(\log \pi_t(a_t \mid s_t) + \bar{H} \right) \right] \tag{12}$$

其中，H 是一个常数向量，其值等于目标熵的超参数。

算法1 给出了使用动态批处理调度并发模型实例的整体流程。调度器首先接收深度神经网络模型的信息，以及每个推理请求的资源利用率。在每个调度时隙之前，它分别使用相应参数初始化所有网络，包括软Q网络、目标软Q网络、策略网络和温度网络。请注意，我们使用两个软Q网络，并取其最小值，以缓解软Q值的高估。

算法 1： 基于学习的调度算法

输入： $(m_t^i, d_t^i, d_s^i, SLO_i)$ 的信息
请求 r_i，资源利用率 u
输出： 批大小 b，并发模型数量 m_c

1 初始化所有神经网络 $Q_{\theta_1}, Q_{\theta_2}, Q_{\theta_1}, Q_{\theta_2}, \pi_\phi, T_\alpha$;
2 随机初始化网络参数 $\theta_1, \theta_2, \phi, \alpha$;
3 初始化一个空的经验回放缓冲区 $\mathcal{D} \leftarrow \emptyset$;
4 **对于**每个调度时隙 t_i **执行**
5 **对于**每个环境步**执行**
6 **如果**请求队列 $seq_b == \emptyset$ **则**
7 将请求 r_i 推送到请求队列 seq_b;
8 **end**
9 根据策略 π 执行动作 $a_t(b, m_c)$ 并获取奖励 $r_t(a_t \mid s_t)$ 使用公式（6）;
10 $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$;
11 $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t) \; s_{t+1})\}$;
12 从批处理中提取当前请求队列 seq_b 槽位 s_i;
13 **end**
14 **对于**每个梯度步**执行**
15 更新执行器网络和目标网络 θ_i 以进行 $i \in \{1, 2\}$ 使用式（9）;
16 更新评论家网络 ϕ 使用公式（11）;
17 <escape>更新温度网络 α 使用公式（12）; </escape>
18 **end**
19 **结束**

在每个调度时隙中，调度器首先检查每个请求队列。如果请求队列为空，则将传入请求放入请求队列（第 7 行）。随后，调度器根据式（5）采取动作（例如，确定每个请求的最佳批大小和并发模型数量），而深度强化学习中的智能体会获得作为效用的即时奖励（第 9 行）。与此同时，状态从 s_t 变为 s_{t+1}，当前状态、动作、奖励和下一状态作为一次状态转移存储在经验回放缓冲区 $\mathcal{D}$中。当当前请求序列中的批量请求执行完毕时，调度器从批处理时隙（第IV-C节）中提取请求序列（第 12 行）。最后，调度器更新所有网络的参数，并重复上述过程（第 14~18行），直至迭代结束。

C. 动态批处理

BCEdge 通过允许单个推理请求指定一个输入批次来实现批量推理。一个输入批次中的输入会同时进行推理，这对图形处理器尤为重要，因为这能大幅提升推理吞吐量。如图 3所示，动态批处理会为不同模型的请求分别维护请求队列，而队列中的每个批次大小取决于

BCEdge 中基于学习的调度器。动态创建的批次会分配给为该模型配置的所有模型实例，随后动态批处理模块会并发执行每个模型的多个批次请求队列。具体来说，动态批处理首先按照到达顺序，将每个请求加入相应的请求队列。与此同时，它会根据每个队列中推理请求的服务级别目标对优先级进行排序，服务级别目标越短，优先级越高。然后，动态批处理会将多个请求合并为一个较大的请求，并在运行时将请求队列中的批量请求分配给相应模型的多个槽位。请注意，如果优先级相同，批量请求会按照到达顺序进行调度。

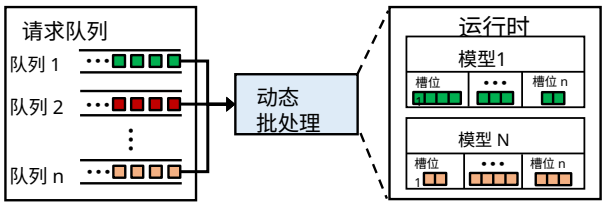


图 3. 动态批处理模块。

D. 模型实例并发

BCEdge 支持在单个或多个 GPU 上并行执行多个模型以及同一模型的多个实例，而由 CPU 执行的深度神经网络模型也由 BCEdge 以类似方式处理。图 4展示了三个深度神经网络模型使用批量请求并行执行模型实例的流水线，其中每个模型分配了两个实例。我们假设 BCEdge 当前未处理任何请求。当最初的三个请求同时到达时，三个模型的每个实例分别处理一个对应请求。随后，BCEdge 立即将这两个请求调度到

GPU，而 GPU 的硬件调度器开始并行处理三个推理。请注意，前三个推理请求会立即并行执行，而最后三个推理请求必须等到前三个请求中的一个完成后才能开始。具体而言，如果同一模型的多个推理请求同时到达，BCEdge 会通过一次只调度一个请求来串行执行它们。

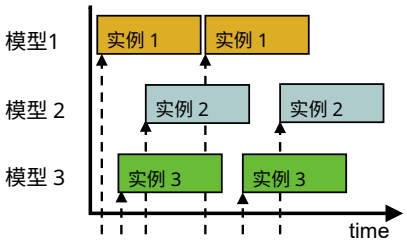


图4. 并发实例模块。

E. 性能分析器

BCEdge 中的性能分析器会定期在线收集性能信息，包括 CPU、GPU、内存的当前利用率，以及系统吞吐量和端到端延迟。具体而言，性能分析器会记录边缘平台上不同输入形状的每个批量请求的性能信息，并将这些信息反馈给调度器。调度器根据上述信息学习，以调度下一个批量请求，即确定最佳批大小和并发模型数量，从而最大化效用。同时，BCEdge 可以通过性能分析器避免系统过载并提高资源利用率，这体现了 BCEdge 在动态资源管理与分配方面的潜在优势。

*F. SLO*感知干扰预测器

多个模型的并发推理，或单个模型多个实例的并发推理，可以同时处理更多请求，从而提高吞吐量。然而，一个重要挑战是单个 GPU 上多个模型并发推理所造成的干扰性能。如图1所示，我们观察到，与独立执行单个模型相比，并发推理会显著增加延迟，因为多个模型会争用边缘平台上的共享资源，尤其是内存。在这种情况下，模型间干扰可能导致调度器作出错误的调度，并违反服务级别目标。

缓解干扰的一项关键挑战，是预测多个推理在同一 GPU 上并发执行时延迟的增加。为了限制干扰效应，我们采用一个开销可忽略的轻量级双层神经网络（NN）作为预测模型，直接学习多个推理在单个 GPU 上并发执行时的干扰延迟。如图5所示，这种简单而有效的、基于 NN 的干扰预测模型利用当前可用的计算资源（即

内存、CPU 和 GPU）以及并发模型的数量

调度器将其学习到的信息作为神经网络的输入。随后，我们根据性能分析器提供的性能反馈，将神经网络输出的估计延迟与实际延迟进行比较，并通过最小化实际值与估计值之间的标准差来训练神经网络。训练后的神经网络旨在提高调度器的稳定性并降低 SLO 违约率。

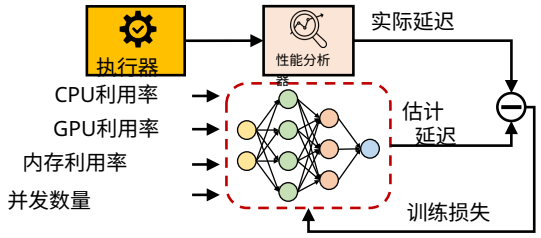


图 5. 基于神经网络的 SLO 感知干扰预测器。

V. 评估

A. 实验设置

BCEdge 原型： 我们将 BCEdge 原型实现为 Triton 的运行时后端 [3], ， Triton 是 NVIDIA 推理服务系统。III详细描述了所评估的推理系统和所用 GPU 规格。该表还列出了操作系统、CUDA、运行时和机器学习框架的版本。如图6所示，我们使用两个 IMX 摄像头和一个麦克风作为物联网设备。请求到达率设为每秒 30 个请求（rps），并遵循泊松随机分布。除非另有说明，所有评估均在 NVIDIA Xavier NX 边缘 GPU 上进行。

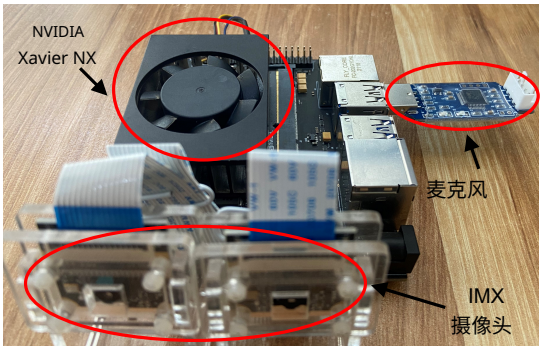


图 6. 在 NVIDIA Xavier NX 边缘平台上实现的 BCEdge 原型。我们使用两个 IMX 摄像头和一个麦克风作为物联网设备。

深度神经网络模型： 我们使用三个常见深度神经网络系列中的六种深度神经网络模型来处理图像和语音数据。具体来说，我们使用 YOLO-v5 [9]执行目标检测任务，使用 MobileNet- v 3 [38], ResNet-18 [39], EfficientNet-B0 [40] 和 Inception- v 3 [41] 执行图像分类任务，尤其是使用 Tiny- BERT [42] 执行语音识别任务。我们使用 TensorRT [10]来减少内存占用，以便更好地进行批处理和并发执行。由于 Xavier NX 的计算能力有限，我们将输入图像下采样至 224×224 分辨率。全部

表 III 评估系统规格

边缘平台	NVIDIA Xavier NX
操作系统	Ubuntu：18.04.6（内核 4.15.0）
软件	CUDA 10.2 和 TensorRT 8.2 [10]
CPU	6 核 Carmel ARMv8.2
GPU	384 核 Volta GPU，配备 48 个张量核心
内存容量	8 吉字节内存
运行时	NVIDIA Triton 推理服务器 2.19.0 [3]
机器学习框架	PyTorch 1.10

表 IV 评估中使用的深度神经网络模型列表

模型	输入形状（维度）	服务级别目标（毫秒）
YOLO-v5（YOLO）	VOC-2012（3x224x224）	138
MobileNet-v3（MOB）	ImageNet-2012（3x224x224）	86
ResNet-18（res）	ImageNet-2012（3x224x224）	58
EfficientNet-B0（eff）	ImageNet-2012（3x224x224）	93
Inception-v3（inc）	ImageNet-2012（3x224x224）	66
TinyBERT（bert）	Speech Commands（1x14）	114

图像是具有 3 个 RGB 通道的彩色帧。每个对应的服务级别目标延迟都列于表 IV。

训练详情： 我们提出的算法 1 基于 SAC [36] 框架。所有网络均使用 Adam 优化器进行训练，学习率为 10^{-3}。每个网络都是一个包含两层的 ReLU 神经网络，隐藏单元数分别为 128 和 64，缓冲区大小固定为 10^6。我们使用四块 NVIDIA GeForce GTX 3080 图形处理器，在一台边缘设备上以 512 的小批量大小离线训练该算法，共训练 500 轮。随后，我们将训练好的算法部署到边缘平台上在线运行。

B. 基线方法

1） 边缘推理服务框架： 我们将 BCEdge 与两个最先进的边缘推理服务框架进行比较：

- ***DeepRT*** [12]：一种软实时调度器，采用动态批处理和最早截止时间优先（EDF）调度算法来执行批量请求。

- 采用 ***Actor-Critic***（***TAC***）的 ***Triton*** [3]：由于 Triton 仅支持手动设置固定的批大小和并发模型数量，我们将 Triton 与不含熵的 Actor-Critic 相结合，以便与 BCEdge 进行比较。

2） 多种调度算法： 我们将 BCEdge 推理服务框架中的传统启发式方法和其他强化学习方法移植过来，与我们的调度算法进行比较，包括：

- 遗传算法（***GA***） [43]：作为一种用于解决优化问题的搜索算法，遗传算法的核心思想源于生物进化理论中的“适者生存”。我们将遗传算法中的适应度函数作为本文提出的效用。

- 近端策略优化（***PPO***） [44]：PPO 是一种基于演员-评论家架构的同策略（智能体在学习过程中的优化策略与行为策略是同一策略）深度强化学习算法。

- 双重深度 ***Q*** 网络（***DDQN***） [45]：作为一种离策略（学习过程中智能体的优化策略与行为策略是不同策略）深度强化学习算法，DDQN 通过将目标 Q 值中动作的选择与目标 Q 值的计算解耦，消除了过估计。

C. 权衡性能

1） 边缘推理框架的比较： 我们首先评估 BCEdge 在吞吐量与延迟之间权衡方面的性能。图 7 报告了表 IV 中六种深度神经网络模型的归一化效用。我们提出的 BCEdge 在所有模型上的表现始终优于 TAC 和 DeepRT。DeepRT 效用较低的原因是缺乏并发推理。尽管 TAC 利用基于学习的方法进行批处理和并发调度，但其智能体缺少能够全面探索环境的熵。相比之下，BCEdge 将熵引入我们的基于学习的调度算法，使深度强化学习中的智能体拥有更强的探索能力，从而获得更高的效用。此外，BCEdge 通过高效调度以及 SLO 感知干扰预测，在吞吐量与延迟之间实现了更好的权衡。具体而言，BCEdge 的效用分别比 DeepRT 和 TAC 平均高出 37% 和 25%。

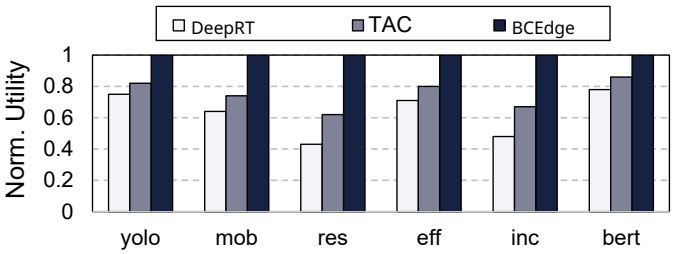


图 7. 与六个深度神经网络模型的归一化效用比较。

接下来，我们分别从吞吐量和延迟方面展示 BCEdge 在 3,000 秒内处理六种深度神经网络模型的表现。图 8 展示了各模型的累计吞吐量堆叠图，图 9 报告了各模型随时间变化的端到端延迟。在 0 至 1,500 秒之间，吞吐量和延迟均渐近增长，这表明 BCEdge 持续优化我们提出的效用函数，以找到每个深度神经网络模型合适的批大小和并发模型数量。从 1,500 秒开始，吞吐量和延迟均趋于饱和，这表明 BCEdge 已在资源和服务级别目标约束下，成功找到最佳批大小和并发模型数量。此外，我们注意到，BCEdge 倾向于牺牲较高的吞吐量来换取较低的延迟，从而实现更好的效用。

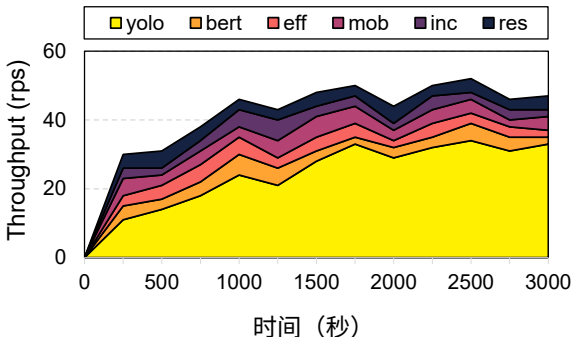


图 8. 六种深度神经网络模型的吞吐量对比。每种模型的调度时长为 3,000 秒。

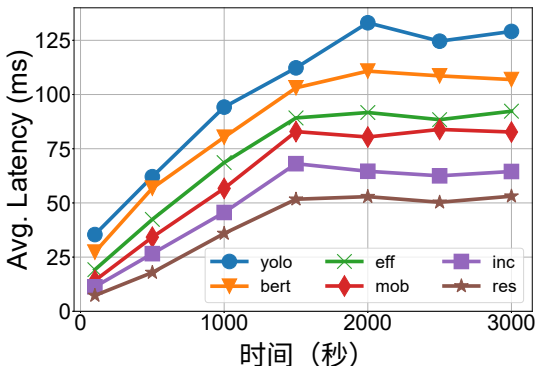


图 9. 六种深度神经网络模型的平均延迟对比。每个模型的调度时长为 3,000 秒。

2) 收敛性能比较： 我们比较了 BCEdge 深度神经网络推理服务框架中三种基于学习的方法（PPO、DDQN 和我们的方法）与一种基于启发式的方法（GA）调度算法的收敛情况。如图 10 所示，与8×∼3。7× 基线方法相比，我们提出的调度算法的收敛速度提升了 1。 这是因为熵使深度强化学习中的智能体能够学习到更多近似最优动作。也就是说，在某些状态下，可能存在多个最优动作，而我们提出的调度算法会使这些动作被选中的概率相同。因此，最大熵可以有效加快学习过程。需要注意的是，遗传算法（GA）的缺点是容易过早收敛，即 GA 探索环境的能力有限，因此不可避免地会收敛到局部最优解。重要的是，GA 涉及大量计算，例如交叉、变异等，导致收敛速度较慢。

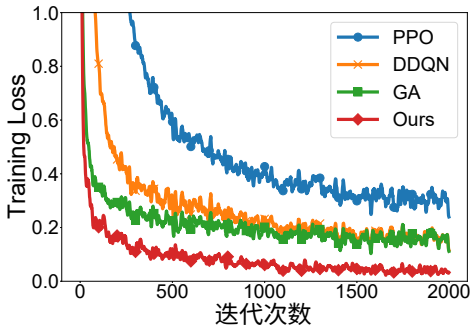


图 10. 基于深度强化学习的方法（PPO、DDQN 和我们的方法）与启发式调度算法（遗传算法）的训练损失对比。

D. 可扩展性评估

为评估表 III 所示平台之外的可扩展性，我们还选择了两种配备 GPU 的边缘平台（例如 NVIDIA Jetson Nano 和 TX2）。表 V 给出了这两种异构边缘平台与 Xavier NX 相比的具体参数。我们分别使用目标检测（YOLO-v5）、图像分类（ResNet-18）和语音识别（TinyBERT）深度神经网络模型来评估 BCEdge 的可扩展性。图11展示了 BCEdge 在

表 V 边缘平台性能参数

边缘平台	算力	内存	CUDA 核心
Jetson Nano	0.47 太 FLOPS（半精度浮点数）	4 GB	128
Jetson TX2	1.33 太 FLOPS（半精度浮点数）	8 GB	256
Xavier NX	21 万亿次运算每秒（8 位整数）	8 GB	384

Jetson Nano/TX2 边缘平台上的效用，并与基线方法进行比较。可以看出，BCEdge 在这两种异构边缘平台上均优于基线方法。由于图像分类深度神经网络模型所需的计算资源最少，也就是说，与 YOLO-v5 和 TinyBERT 相比，ResNet-18 可配置更多的批大小和并发模型数量，因此 BCEdge 对 ResNet-18 实现了更好的权衡。图 7 中也有类似结果。即使是在计算能力最弱的 Jetson Nano 上，BCEdge 的效用也能提升 30%

与 DeepRT 和 TAC 相比，分别高出 19%。由于 Jetson TX2 拥有更多计算资源，可配置更大的批大小和更多并发模型，因此 BCEdge 能够实现更高的性能，其效用分别比 DeepRT 和 TAC 高 39% 和 27%。

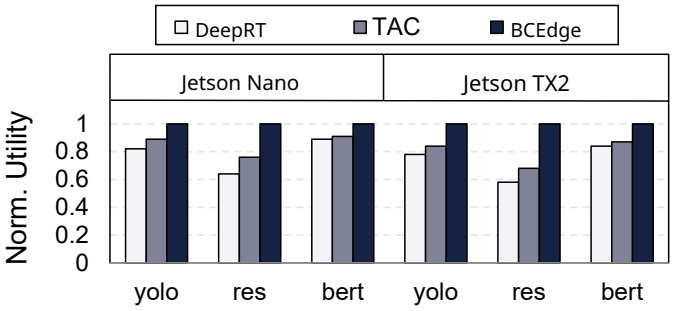


图 11. 异构边缘平台的效用。边缘平台的计算资源越多，效用越高。

图。12 展示了与图11中效用对应的三个边缘平台的吞吐量和延迟。图 12 中的左侧 y 轴表示峰值吞吐量，右侧 y 轴表示相应的平均延迟。正如图 12所示，对于计算资源较少的深度神经网络模型，以及计算资源更充足的边缘平台，BCEdge 的性能提升也更为显著。即使在性能最弱的 Jetson Nano 上，BCEdge 也能充分利用计算资源，为不同的深度神经网络模型优化吞吐量与延迟之间的权衡。总之，BCEdge 展现出灵活的可扩展性，能够适应资源受限的异构边缘平台。

E. 干扰模型评估

在此实验中，我们研究了 BCEdge 中所提出的干扰预测模型在不同每秒请求数（rps）下对 SLO 违约率的影响。对于每个深度神经网络模型，干扰预测模型以 1 秒为周期记录了共 2000 次推理干扰。在收集的 2000 条数据中，我们随机选取 1600 条执行数据作为训练数据，另选 400 条数据用于验证。图 13 展示了预测的累积分布

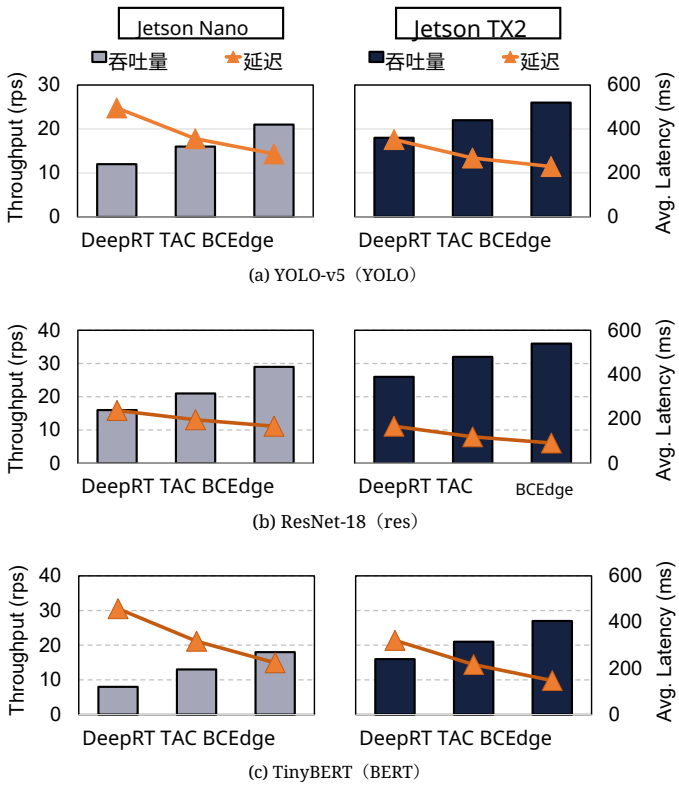


图 12。异构边缘平台的吞吐量和平均延迟。边缘平台的计算资源越多，吞吐量越高，平均延迟越低。

误差，并将我们基于神经网络的干扰模型与线性回归模型进行比较 [16], [46]。所提出的模型在 90% 的情况下可将预测误差控制在 2.69% 以内；若允许 3.25% 的误差，则这一比例可达 95%。与线性回归模型相比，该模型将错误率降低了一半。由于我们在图 1 中观察到的模型干扰并非简单的线性关系，线性回归模型的预测误差较高。相比之下，我们提出的基于神经网络的干扰模型考虑了边缘平台的资源利用率以及深度神经网络模型的实际延迟，因此能够准确预测干扰延迟。

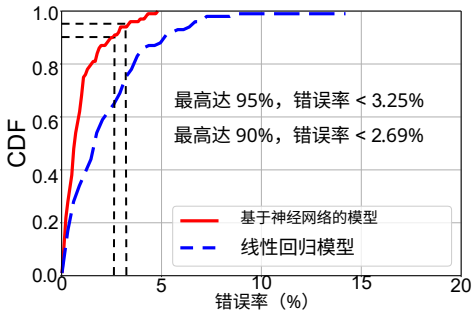


图 13。相对错误率的累积分布。我们提出的基于神经网络的模型能够预测最多 95% 的情况，错误率低于 3.25%。

图 14展示了在 30 rps 下，BCEdge 启用和未启用干扰预测模型时 SLO 违约率的累积分布。我们分析了图 8 和图 9中调度时长为 3000 秒时的 SLO 违约率。

与未启用干扰预测模型的 BCEdge 相比，所提出的模型可将 BCEdge 的 SLO 违约率从 9.2% 降至 4.1%。这表明，干扰预测模型能够提高 BCEdge 的鲁棒性，并有效降低 SLO 违约率。

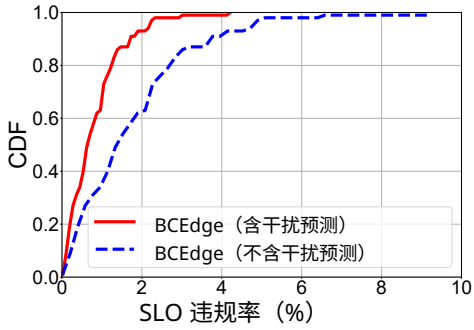


图 14。30rps 下 SLO 违约率的累积分布。与未采用干扰预测模型时最高达 9.2% 的 SLO 违约率相比，我们提出的干扰预测模型能够将 SLO 违约率控制在 4% 以内。

如图 15所示，我们通过逐步提高每秒请求数（rps）来测量 SLO 违约率。可以看出，BCEdge 在所有 rps 下的 SLO 违约率均最低，平均而言分别比 DeepRT 和 TAC 低 53% 和 25%。即使在 40rps 下，BCEdge 的 SLO 违约率也不超过 5%。由于 DeepRT 中的软实时调度器 [12] 仅适用于没有严格 SLO 约束的深度神经网络模型，因此在严格的 SLO 约束下，其 SLO 违约率最高。此外，TAC 未考虑模型干扰的影响，因此其 SLO 违约率高于 BCEdge。

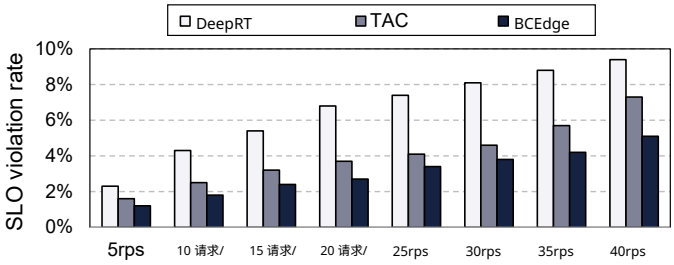


图 15. 在不同 rps 下，使用真实世界的六种深度神经网络模型基准测试对比服务级别目标（SLO）违约率。得益于干扰预测模型，BCEdge 的 SLO 违约率最低。

F. 调度开销

为了衡量调度器带来的运行时开销，我们从平均调度延迟方面将 BCEdge 与 DeepRT 和 TAC 进行比较。图 16 展示了这些调度开销。可以看出，BCEdge 的调度开销较低，这是因为 BCEdge 中的调度器引入了最大熵，能够学习更近似的最优动作，从而加快学习并降低开销。具体而言，BCEdge 的平均调度开销分别比 DeepRT 和 TAC 低 26% 和 43%。这表明 BCEdge 能以极低的开销高效调度批处理和并发请求。需要注意的是，

我们没有评估性能分析器和干扰预测模型的开销，因为这些开销可以忽略不计。

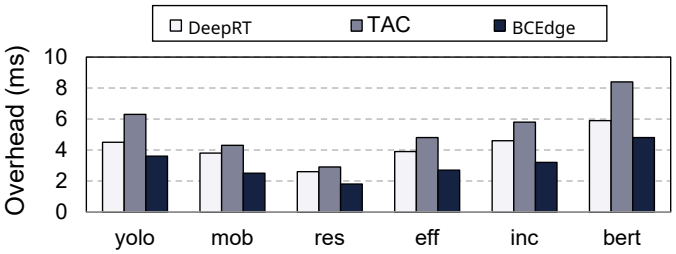


图16. 六种深度神经网络模型的调度开销比较。

VI. 结论

在本研究中，我们提出了 BCEdge，这是一种具备自适应、服务级别目标感知能力的多租户深度神经网络服务调度框架。BCEdge 支持边缘平台上的边缘智能应用采用批处理和并发推理，从而实现高吞吐量与低延迟。BCEdge 的关键在于一种基于最大熵深度强化学习的调度器，它能够自动联合优化批大小和并发模型数量。与最先进的解决方案相比，BCEdge 的平均效用提升最高可达 37.6%，同时满足服务级别目标。

参考文献

[1] D. Crankshaw、X. Wang、G. Zhou、M. J. Franklin、J. E. Gonzalez 和 I. Stoica，“Clipper：低延迟在线预测服务系统”。载于 NSDI，第 17 卷，2017 年，第 613–627 页。[2] C. Olston、N. Fiedel、K. Gorovoy、J. Harmsen、L. Lao、F. Li、V. Ra-jashekhar、S. Ramesh 和 J. Soyke，“TensorFlow-Serving：灵活、高性能的机器学习服务”，arXiv 预印本 arXiv:1712.06139，2017 年。[3] “NVIDIA Triton 推理服务器”，https://developer.nvidia.com/ nvidia-triton-inference-server。[4] S. Deng、H. Zhao、W. Fang、J. Yin、S. Dustdar 和 A. Y. Zomaya，“边缘智能：边缘计算与人工智能的交汇”，IEEE 物联网期刊，第 7 卷，第 8 期，第 7457–7469 页，2020年。[5] D. Feng、C. Haase-Sch¨utz、L. Rosenbaum、H. Hertlein、C. Glaeser、F. Timm、W. Wiesbeck 和 K. Dietmayer，“面向自动驾驶的深度多模态目标检测与语义分割：数据集、方法与挑战”，IEEE Transactions on Intelligent Transportation Systems，第 22 卷，第 3 期，第 1341–1360 页，2020年。[6] X. Xie、F. Sun、Z. Liu、S. Wu、J. Gao、J. Zhang、B. Ding 和 B. Cui，“用于序列推荐的对比学习”，载于《2022年第38届IEEE数据工程国际会议（ICDE）》。IEEE，2022 年，第 1259–1273 页。[7] Z. Liu、G. Lan、J. Stojkovic、Y. Zhang、C. Joe-Wong 和 M. Gorlatova，“Collabar：面向移动增强现实的边缘辅助协同图像识别”，载于《2020年第19届ACM/IEEE传感器网络信息处理国际会议（IPSN）》。IEEE，2020 年，第 301–312 页。[8] S. Han、H. Mao 和 W. J. Dally，“深度压缩：通过剪枝、训练量化和霍夫曼编码压缩深度神经网络”，arXiv 预印本 arXiv:1510.00149，2015 年。[9] “YOLOv5”，https://github.com/ultralytics/yolov5。[10] “NVIDIA TensorRT”，https://developer.nvidia.com/tensorrt。[11] S. Tuli、S. Ilager、K. Ramamohanarao 和 R. Buyya，“利用 A3C学习和残差循环神经网络，为随机边缘云计算环境进行动态调度”，IEEE 移动计算汇刊，2020年。[12] Z. Yang、K. Nahrstedt、H. Guo 和 Q. Zhou，“DeepRT：面向边缘计算机视觉应用的软实时调度器”，载于《2021年IEEE/ACM边缘计算研讨会（SEC）》。IEEE，2021 年，第 271–284 页。

[13] Y. Choi和M. Rhu，“Prema：面向可抢占神经处理单元的预测性多任务调度算法”，载于2020年IEEE国际高性能计算机体系结构研讨会（HPCA）。IEEE，2020年，第220–233页。[14] W. Cui、H. Zhao、Q. Chen、H. Wei、Z. Li、D. Zeng、C. Li和M. Guo，“{DVABatch}：面向多样性感知的 {多入口}{多出口} 批处理，以实现 {DNN} 服务在 {GPU}上的高效处理”，载于2022年USENIX年度技术会议（USENIX ATC 22），2022年，第183–198页。[15] C. Zhang、M. Yu、W. Wang和F. Yan，“MArK：利用云服务实现成本效益型、服务级目标感知的机器学习推理服务”，载于USENIX年度技术会议，2019年，第1049–1062页。[16] S. Choi、S. Lee、Y. Kim、J. Park、Y. Kwon和J. Huh，“在具有 {多GPU} 的服务器上通过{时空} 共享提供异构机器学习模型服务”，载于2022年USENIX年度技术会议（USENIX ATC 22），2022年，第199–216页。[17] W. Seo、S. Cha、Y. Kim、J. Huh和J. Park，“面向边缘平台中异构处理器的服务级目标感知推理调度器”，《架构与代码优化汇刊》（TACO），第18卷，第4期，第1–26页，2021年。[18] F. Romero、Q. Li、N. J. Yadwadkar和C. Kozyrakis，“INFaaS：自动化的无模型推理服务”，载于USENIX年度技术会议，2021年，第397–411页。[19] A. Gujarati、R. Karimi、S. Alzayat、W. Hao、A. Kaufmann、Y. Vigfusson和J. Mace，“像时钟般提供DNN服务：自底向上的性能可预测性”，载于第14届USENIX操作系统设计与实现研讨会（OSDI 20），2020年，第443–462页。[20] A. Ali、R. Pinciroli、F. Yan和E. Smirni，“BATCH：基于自适应批处理的无服务器平台机器学习推理服务”，载于SC20：国际高性能计算、网络、存储与分析会议。IEEE，2020年，第1–15页。[21] X. Liu、B. Wu、X. Yuan和X. Yi，“Leia：边缘上的轻量级密码神经网络推理系统”，《IEEE信息取证与安全汇刊》，第17卷，第237–252页，2021年。[22] J. Hou、H. Liu、Y. Liu、Y. Wang、P.-J. Wan和X.-Y. Li，“模型保护：面向边缘模型隐私的实时隐私保护推理服务”，《IEEE可信与安全计算汇刊》，2021年。[23] Y. G. Kim和C.-J. Wu，“Autoscale：使用强化学习优化随机边缘推理的能效”，载于2020年第53届IEEE/ACM微体系结构国际研讨会（MICRO）。IEEE，2020年，第1082–1096页。[24] W. Zhang、D. Yang、H. Peng、W. Wu、W. Quan、H. Zhang和X. Shen，“基于深度强化学习的工业物联网DNN推理资源管理”，《IEEE车辆技术汇刊》，第70卷，第8期，第7605–7618页，2021年。[25] M. Han、H. Zhang、R. Chen和H. Chen，“并发 {GPU加速}{DNN} 推理的微秒级抢占”，载于第16届USENIX操作系统设计与实现研讨会（OSDI 22），2022年，第539–558页。[26] W. Cui、H. Zhao、Q. Chen、N. Zheng、J. Leng、J. Zhao、Z. Song、T. Ma、Y. Yang、C. Li等，“基于确定性算子重叠和精确延迟预测实现同步DNN服务”，载于国际高性能计算、网络、存储与分析会议论文集，2021年，第1–15页。[27] Z. Liu、J. Leng、Z. Zhang、Q. Chen、C. Li和M. Guo，“VELTAIR：通过自适应编译和调度实现高性能多租户深度学习服务”，载于第27届ACM编程语言与操作系统架构支持国际会议论文集，2022年，第388–401页。[28] S. S. Mondal、N. Sheoran和S. Mitra，“使用强化学习调度时变工作负载”，载于AAAI人工智能会议论文集，第35卷，第10期，2021年，第9000–9008页。[29] Y. Han、S. Shen、X. Wang、S. Wang和V. C. Leung，“面向Kubernetes的边缘云系统的定制化基于学习的调度”，载于IEEE INFOCOM 2021-IEEE计算机通信会议。IEEE，2021年，第1–10页。[30] X. Wang、Z. Ning和S. Guo，“泛在边缘计算的多智能体模仿学习：一种去中心化计算卸载算法”，《IEEE并行与分布式系统汇刊》，第32卷，第2期，第411–425页，2020年。[31] J. Zou、T. Hao、C. Yu和H. Jin，“A3C-DO：边缘场景中基于深度强化学习的区域资源调度框架”，《IEEE计算机汇刊》，第70卷，第2期，第228–239页，2020年。[32] D. Shi、H. Gao、L. Wang、M. Pan、Z. Han和H. V. Poor，“平均场博弈引导的深度强化学习，用于边缘场景中的任务放置”

协同多接入边缘计算，”《*IEEE*物联网期刊》，第 7 卷，第 10 期，第 9330–9340 页，2020年。[33] S. Tuli、G. Casale 和 N. R. Jennings，“ MCDS：移动边缘云计算系统中的 AI 增强型工作流调度”，《*IEEE*并行与分布式系统汇刊》，第 33 卷，第 11 期，第 2794–2807 页，2021年。[34] J. Meng、H. Tan、X.-Y. Li、Z. Han 和 B. Li，“边缘计算中的在线期限感知任务分派与调度”，*IEEE*并行与分布式系统汇刊，第 31 卷，第 1286–1270 页，2019 年第 6 期。[35] A. Qiao、S. K. Choe、S. J. Subramanya、W. Neiswanger、Q. Ho、H. Zhang、G. R. Ganger 和 E. P. Xing，“Pollux：面向吞吐效率优化的深度学习的协同自适应集群调度”，载于 第 *15* 届 {*USENIX*}操作系统设计与实现研讨会（ { } *OSDI*）。 *21* 2021
[36] P. Christodoulou，“离散动作场景下的软演员-评论家”，*arXiv* 预印本 *arXiv:1910.07207*，2019 年。[37] T. Haarnoja、A. Zhou、K. Hartikainen、G. Tucker、S. Ha、J. Tan、V. Kumar、H. Zhu、A. Gupta、P. Abbeel 等，“软演员-评论家算法及其应用”，*arXiv* 预印本 *arXiv:1812.05905*，2018 年。[38] A. G. Howard、M. Zhu、B. Chen、D. Kalenichenko、W. Wang、T. Weyand、M. Andreetto 和 H. Adam，“ MobileNets：面向移动视觉应用的高效卷积神经网络”，*arXiv* 预印本 *arXiv:1704.04861*，2017 年。[39] K. He、X. Zhang、S. Ren 和 J. Sun，“ 用于图像识别的深度残差学习”，载于*IEEE*计算机视觉与模式识别会议论文集，2016 年，第 770–778 页。[40] M. Tan 和 Q. Le，“EfficientNet：重新思考卷积神经网络的模型缩放”，载于国际机器学习会议。PMLR，2019 年，第 6105–6114 页。[41] C. Szegedy、V. Vanhoucke、S. Ioffe、J. Shlens 和 Z. Wojna，“ 重新思考面向计算机视觉的 Inception 架构 ”，载于 *IEEE* 计算机视觉与模式识别会议论文集，2016 年，第 2818–2826 页。[42] X. Jiao、Y. Yin、L. Shang、X. Jiang、X. Chen、L. Li、F. Wang 和 Q. Liu，“ TinyBERT：为自然语言理解蒸馏 BERT”，载于计算语言学协会年会论文集：*EMNLP 2020*，2020 年 11 月，第 4163–4174 页。[43] D. Whitley，“遗传算法教程”，《统计与计算》，第 4 卷，第 2 期，第 65–85 页，1994 年。[44] J. Schulman、F. Wolski、P. Dhariwal、A. Radford 和 O. Klimov，“近端策略优化算法”，*arXiv* 预印本 *arXiv:1707 06347*，2017 年。。[45] H. Van Hasselt、A. Guez 和 D. Silver，“基于双重Q学习的深度强化学习”，载于*AAAI*人工智能会议论文集，第 30 卷，第 1 期，2016 年。[46] Q. Chen、H. Yang、J. Mars 和 L. Tang，“Baymax：提升仓储规模计算机中不可抢占加速器的服务质量感知能力与利用率”，*ACM SIGPLAN*通告，第 51 卷，第 4 期，第 681–696 页，2016 年。



Ziyang Zhang Ziyang Zhang于2020年在中国天津南开大学电子信息与光学工程学院获得理学硕士学位。目前，他正在哈尔滨工业大学（HIT）计算机科学与技术学院攻读博士学位。他的研究兴趣包括边缘计算、机器学习系统和深度学习。



Huan Li Huan Li博士于2006年在美国马萨诸塞大学阿默斯特分校获得计算机科学博士学位。她目前的研究兴趣包括人工智能物联网、边缘智能、分布式实时系统和数据科学。她曾担任多场国际会议的程序委员会委员，包括IEEE RTAS、ICDCS、RTCSA等。她目前是IEEE高级会员。



Yang Zhao Yang Zhao于2003年获得山东大学电气工程理学学士学位，2006年获得北京航空航天大学电气工程硕士学位，并于2012年获得犹他大学电气与计算机工程博士学位。2013年至2021年，他在GE全球研究中心担任首席研究工程师。自2021年起，他任职于哈尔滨工业大学（深圳），担任国际人工智能研究院研究教授。

他的研究兴趣包括无线感知、边缘计算和信息物理系统。他是IEEE高级会员。



Changyao Lin Changyao Lin于2020年和2022年分别在中国哈尔滨哈尔滨工业大学（HIT）计算机科学与技术学院获得学士学位和理学硕士学位。目前，他正在哈尔滨工业大学攻读博士学位。他的研究兴趣包括边缘计算、分布式系统和深度学习。



Jie Liu Jie Liu是中国哈尔滨工业大学深圳校区（HIT Shenzhen）的讲席教授，并担任该校人工智能研究院院长。在加入哈尔滨工业大学之前，他曾在施乐帕洛阿尔托研究中心和微软工作18年。他曾任微软研究院（雷德蒙德）首席研究经理，并担任微软合伙人。他的研究兴趣包括信息物理系统、面向物联网的人工智能和高能效计算。他曾获IEEE TCCPS杰出领导力奖以及顶级会议颁发的6项最佳论文奖。他是IEEE会士和ACM杰出科学家、ACM SIGBED中国分会创始主席。