计算机科学
COMPUTER SCIENCE

基于负载特征的边缘智能系统性能优化

胡朝霞，胡海周，蒋从锋，万健

引用本文

胡朝霞，胡海周，蒋从锋，万健. 基于负载特征的边缘智能系统性能优化[J]. 计算机科学, 2022, 49(11): 266-276.

HU Zhao-xia, HU Hai-zhou, JIANG Cong-fengand WAN Jian. Workload Characteristics Based Performance Optimization for Edge Intelligence[J]. Computer Science, 2022, 49(11): 266-276.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000067

基于负载特征的边缘智能系统性能优化

胡朝霞[1]　胡海周[1]　蒋从锋[1]　万　健[2]
1 杭州电子科技大学计算机学院　杭州 310018
2 浙江科技学院信息与电子工程学院　杭州 310023
(hltz37@hdu.edu.cn)

摘　要　边缘智能指利用人工智能算法为网络边缘设备提供数据分析能力的一种服务形式。然而,边缘计算环境比云计算更加复杂和多变。在构建边缘智能的过程中存在很多问题,例如缺乏量化的评价标准、异构计算平台、复杂的网络拓扑、不断变化的用户需求等,其中比较突出的是算法模型的高资源需求与边缘设备资源储备低之间的矛盾。机器学习是边缘智能的主要工作负载,它需要大量的计算资源,然而边缘设备的计算资源有限,两者的供求关系并不匹配,边缘智能负载的部署和优化成为了一个难题。因此,针对边缘智能负载性能优化问题,文中提出了基于负载特征的边缘智能性能优化CECI(Cloud-Edge Collaborative Inference)策略,从模型选择、批量自适应调整和云边协同方面对不同机器学习负载进行了优化。在模型选择方面,使用基于目标权重的模型自适应选择策略,实现在多个条件约束下,综合权衡多个性能优化目标的效果。在批量自适应调整方面,提出了基于开销反馈的批量自适应调整算法,使得模型在运行时能够达到更好的性能。在云边协同方面,通过结合网络状态和用户时延要求设计出了云边协同策略,进而达到了动态利用云端计算资源的效果。实验结果表明,与云智能相比,所提出的基于负载特征的边缘智能能够缩短50.79%的程序运行时间,降低了42.46%的系统能耗,并提升了4.52%的模型准确率。

关键词:边缘智能;云边协同;边缘计算;负载识别;模型选择
中图法分类号　TP391

Workload Characteristics Based Performance Optimization for Edge Intelligence

HU Zhao-xia[1],HU Hai-zhou[1],JIANG Cong-feng[1] and WAN Jian[2]
1 School of Computer Science and Technology,Hangzhou Dianzi University,Hangzhou 310018,China
2 School of Information and Electronic Engineering,Zhejiang University of Science and Technology,Hangzhou 310023,China

Abstract　Edge intelligence refers to a form of service that uses artificial intelligence algorithms to provide data analysis capabilities for network edge devices. However,the edge computing environment is more complex and changeable than cloud computing. There are many problems in the process of building edge intelligence,such as the lack of quantitative evaluation standards,heterogeneous computing platforms,complex network topologies,and changing user needs. Among them,the more prominent is the contradiction between the high resource demand of the algorithm model and the low resource reserve of edge devices. Machine learning is the main workload of edge intelligence. It requires a lot of computing resources. However,the computing resources of edge devices are limited,and the supply and demand between the two do not match. The deployment and optimization of edge intelligent load has become a problem. Therefore,in response to the problem of edge intelligent load performance optimization,this paper proposes cloud-edge collaborative inference(CECI) based on load characteristics,which is optimized for different machine learning loads in terms of model selection,batch adaptive adjustment and cloud-side collaboration. In terms of model selection,a model adaptive selection strategy based on target weights is used to comprehensively weigh the effects of multiple performance optimization targets under multiple constraints. In the aspect of batch adaptive adjustment,a batch adaptive adjustment algorithm based on overhead feedback is proposed,so that the model can achieve better performance at runtime. In terms of cloud-side collaboration,a cloud-side collaboration strategy is designed by combining network status and user delay requirements to achieve the effect of dynamic utilization of cloud computing resources. Experimental results show that compared with cloud intelligence,the edge intelligence based on load characteristics proposed in this paper can reduce program running time by 50.79%,reduce system energy consumption by 42.46%,and improve model accuracy by 4.52%.

Keywords　Edge intelligence,Cloud-edge collaborative,Edge computing,Workload recognition,Model selection

到稿日期:2021-10-10　返修日期:2022-06-22
基金项目:国家自然科学基金面上项目(61972118,61972358)
This work was supported by the General Program of National Natural Science Foundation of China(61972118,61972358).
通信作者:蒋从锋(cjiang@hdu.edu.cn)

1 引言

物联网和5G通信技术的快速发展，促使终端边缘设备连接到互联网的成本逐渐降低，网络边缘接入的电子设备数量呈现指数级增长。以云计算为代表的集中式数据处理方式在处理海量边缘数据时，会导致较长的服务响应时延并消耗大量网络带宽，因此云端提供的人工智能服务（后文简称云智能）很难保证实时性。而边缘计算的出现为优化云智能中存在的性能问题提供了一种新的解决思路，它的核心思想是在产生数据的网络边缘端消费数据，进而降低服务的响应时延和能耗，并加强数据隐私保护。此外，移动互联时代的到来使得在边缘节点上部署人工智能算法的需求愈加强烈，将人工智能算法与新兴的边缘计算范式相结合成为了新的研究热点，边缘智能的概念应运而生。

边缘智能指利用人工智能算法为网络边缘设备提供数据分析能力的服务形式，它在性能表现上优于云智能。虽然边缘智能可以为用户带来更好的人工智能服务体验，但在具体实现和应用中需要解决很多技术难题，如边缘智能部署问题、边缘节点管理问题、云边协同问题等，其中亟待解决的问题是算法模型的高资源需求与边缘设备低资源储备之间的矛盾。而机器学习作为边缘智能中的主要工作负载，其计算量的供需关系更是成为了边缘智能机器学习负载（简称边缘智能负载）的部署与优化的难题。

受此问题的启发，本文聚焦于根据边缘智能负载的性能优化问题，设计了云边协同智能推理框架。首先，从问题分析入手，将这个问题分解为模型选择、批量自适应和云边协同3个子问题，并介绍了CECI框架的相关功能模块。其次，设计了基于目标权重的模型自适应选择策略，通过预选和优选结合的方式为边缘设备匹配最合适的算法模型，进而达到优化模型相对性能的效果。接着，提出了基于开销反馈的批量自适应调整算法，为模型提供最佳的运行时批量，优化模型运行时的性能。然后，提出了基于服务级别协议（Service Level Agreement，SLA）感知的云边协同策略，致力于在满足SLA约束的条件下，使用云端的计算资源来提升准确率。最后，针对提出的算法和策略分别设计实验来进行有效性评估。

2 相关工作

目前边缘智能的性能优化一般从模型设计阶段、模型训练阶段和模型推理阶段切入。

（1）模型设计阶段。在模型设计阶段一般会通过各种方法减少人工智能模型所需要的资源，以解决或缓解边缘节点资源受限的问题。常见的实现方案有两种：1）依据边缘节点的特征设计出的轻量级模型[1]；2）将已有的模型进行再训练和压缩，进而得到一个资源需求小的模型，常见的方法有模型裁剪、知识蒸馏、提前退出和模型量化等。

（2）模型训练阶段。在云智能范式中，各终端设备将数据集发送到云端，云端利用收集到的数据集对人工智能模型进行训练。该训练模式可以将数据保存在边缘，并且充分利用边缘设备的资源。然而，当终端数量较多、终端数据体量较大时，数据传输到云端会有较高的时延（如视频数据）。与此同时，终端用户并不想暴露自己的数据隐私，不愿意把数据分享到云中心。因此，使用云边协同的训练模式可以提供更好的服务以满足用户需求[2-5]。该训练模式主要有两种，一种是针对大量数据的分布式训练；另一种是针对数据隐私保护的联邦学习。

（3）模型推理阶段。由于边缘节点资源受限，使得人工智能推理阶段成为边缘智能优化的方向之一。目前，与之相关的研究工作主要可以分为两类：一方面，相关文献[6]设计了根据任务特征为边缘节点选择模型的方法，这些特征包括数据集的特征、用户的性能需求等。另一方面，任务划分方法成为了边缘智能性能优化的另外一种手段。在任务划分方法中，推理任务被划分为云阶段计算任务和边阶段计算任务，该方法将数据依赖强的任务划分到本地执行，将计算资源需求强的任务推动到云端计算。通过合理利用边缘端靠近数据、云端计算资源丰富的特性为边缘智能带来性能上的提升。

表1列出了常见的各阶段的部分优化方法。此外还有其他研究[7-12]也做了相关的优化研究。

表1 边缘智能各阶段优化工作的对比

Table1 Comparison of edge intelligent optimization work

优化阶段	方法	作者	模型	边缘设备	评估指标
模型设计阶段	轻量级模型	Sheng 等[13]	MobileNets	—	时延、准确率
		Snatos 等[14]	SqueezeNet	—	时延、模型大小
	模型压缩	Cerutti 等[15]	CNN	物联网设备	准确率、模型大小
		Yao 等[16]	LeNet5	Intel Edison	时延、能耗、内存
	多策略结合	Han 等[17]	LSTM	XCKU060 FPGA	时延、能耗
		Liu 等[18]	LeNet，AlexNet	手机、穿戴设备	内存、时延、能耗
模型训练阶段	分布式机器学习	Hardy 等[19]	CNN	—	准确率
		Dean 等[20]	DNN	—	节点数量、训练时间
	联邦学习	Yu 等[21]	—	—	缓存效率、通信次数
		Wang 等[22]	—	—	缓存命中率、资源利用率
模型推理阶段	模型选择	Fang 等[23]	VGG16，ResNet-50	Samsung Galaxy S8，LG Nexus 5	能耗、准确率
		Taylor 等[24]	CNN	NVIDIA Jetson TX2	时延、准确率
	模型划分	Kang 等[25]	AlexNet，VGG	NVIDIA Jetson TK1	时延、能耗吞吐量
		Mao 等[26]	VGG	LG Nexus5	时延、数据传输

3 性能优化方案整体设计

3.1 优化目标

终端用户对边缘智能的性能期望是不同的，如有些希望结果尽可能地精确，而有些则希望请求尽快被满足，允许结果存在一定偏差。因此，边缘智能负载在边缘设备上的真实运行性能不能仅根据准确率指标来评估，而是应结合多个维度的性能指标综合考虑。将服务体验（Quality of Experience，

QoE)作为边缘智能负载评估的方式是更合适的,它取决于推理时间、吞吐量、准确率、内存占用、系统能耗等性能指标。本研究将上述指标用于边缘智能负载性能优化的评估工作,力争以更少的资源获得更快的推理时间和更高的准确率。

3.2 问题分解

云智能存在隐私保护缺失、高计算成本和高时延等问题,但边缘智能的出现为其提供了一种可行的解决思路。然而,边缘智能技术仍然面临许多挑战,尤其是将其应用至实践的过程:1) 供需矛盾成为了阻碍边缘智能发展的瓶颈,尤其是边缘设备计算资源不满足人工智能的算力资源需求的矛盾;2)算法模型在不同平台上以不同批量运行会影响边缘智能的负载性能;3)云中心和边缘节点之间复杂的网络数据交互为云边协同计算带来了巨大的挑战。

针对上述问题,本研究将边缘智能负载性能优化问题分解为若干子问题,致力于简化该问题的求解过程。

(1)模型选择问题。边缘设备的多样性带来了模型选择这一复杂的问题,即如何在有限的资源下选择最适配硬件资源的模型结构,进而满足用户的性能需求。具体而言,参数量和计算量较小的算法模型会占据更少的系统资源,推理时间更短,但往往以降低准确率为代价;而参数量和计算量较大的算法模型具有较高的准确率,但往往以增加运行时间和系统能耗为代价,甚至部分模型无法在资源受限的边缘设备上运行。平衡准确率、推理时间和系统能耗等多个指标是模型选择算法的意义。

(2)批量自适应问题。边缘智能负载单次加载的数据量规模(批量)会对模型性能产生影响。因此,考虑硬件资源约束下的批量自适应调整方案对边缘智能的性能优化具有重大意义。举例而言,通过增加批量可以提升负载的每秒传输帧数(Frames Per Second, FPS),但会增加算法的计算资源需求;而减少批量虽然可以降低计算资源需求,适配更多边缘设备,但是会减少负载的 FPS,增加任务处理的整体耗时。

(3)云边协同计算问题。云边协同计算旨在为异构的边缘设备和云中心之间建立一种任务协调机制。通过利用边缘节点靠近数据和云中心计算资源丰富的特点,设置合理的云边协同策略,进而为边缘智能负载带来性能的提升。

为了最大限度地提升边缘智能负载的性能,一种有效的方式是综合多种技术和优化方法协作,边缘智能负载优化问题的解空间如图 1 所示,主要由模型选择、批量自适应和云边协同这 3 个维度构成,解空间中的点代表优化方案。

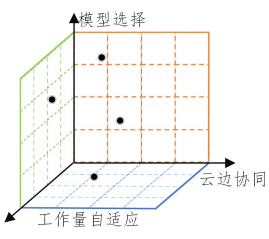


图 1 边缘智能性能优化问题解空间可视化

Fig. 1 Visualization of solution space for edge intelligent performance optimization problems

由于应用于不同场景下的边缘智能服务所需的 QoS 是不同的,应综合考虑多个性能指标,如推理时间、能耗、内存占用和准确率等,然后确定优化方案。因此,基于性能目标权重在解空间中找到合适的优化方案是本文后续的主要工作。

由于终端用户具有不同的性能期望,本研究设计了云边协同智能推理(Cloud -Edge Collaborative Inference, CECI)框架,该框架可针对用户的 QoE 动态调整优化目标,实现多约束条件下的多目标优化效果。CECI 是专为资源受限的边缘设备而设计的轻量级边缘智能框架,旨在为边缘赋能,将人工智能算法部署在边缘设备上。CECI 框架的实现细节如图 2 所示,该框架由云端和边缘端两部分组成。

云端和边缘端都具有模型选择模块、模型运行时模块和云边协同模块,这些模块是解决边缘智能性能优化问题的关键组件。模型选择模块可以根据边缘计算平台特征、用户性能期望、模型历史运行信息等来综合考虑,适配出最适合此次任务的算法模型;模型运行时模块则根据计算平台的资源约束为模型快速搜索出最适配的运行时批量;云边协同模块解决的是任务在云端和边缘端如何分配的问题,通过权衡计算开销、数据传输开销以及计算平台资源等因素,最终为任务提供一种合适的分配策略。边缘端独有的组件是数据采集模块,用于接收任务信息和用户的性能期望;而云端独有的组件是负载分析模块,该模块通过分析各个模型在不同计算平台上运行的历史信息,来增强对模型相对性能的量化认知。

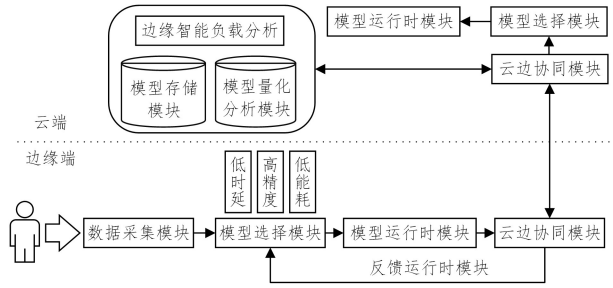


图 2 CECI 框架

Fig. 2 CECI framework

CECI 的整体处理流程如下:

(1)用户在边缘端提交任务请求、原始数据和性能期望。数据采集模块对数据进行预处理。

(2)模型选择模块将根据性能期望、运行平台特征、历史运行数据等挑选出最适配的模型运行结果。

(3)模型运行时模块搜索出最佳的运行批量配置。

(4)云边协同模块收集此次的运行信息并将其上传至云端,同时根据协同策略调整卸载到云端执行任务的比例。

本文提出的 CECI 框架旨在为边缘智能负载提供一种性能优化思路,它通过分析边缘智能负载特征为特定的边缘设备选择最合适的算法模型;通过运行时批量自适应调整,在满足计算平台资源约束的同时,达到优化负载性能的效果;通过边缘设备和云中心的协同合作来实现任务的重分配和计算资源的充分利用。

4 基于目标权重的模型自适应选择策略

4.1 目标权重设计

模型选择策略旨在解决边缘计算平台和算法模型之间的

适配问题。边缘设备具有不同的硬件特征，而算法模型也具有不同的资源需求，利用上述特点，可以基于用户的性能期望选择最合适的模型，执行推理任务。但单一目标的优化并不能覆盖所有用户场景，也不能满足用户复杂而多样的性能需求，因此模型选择策略应该聚焦于多约束条件下的多目标优化问题，如结合模型准确率、推理时间、能耗等性能的多目标优化。

如上所述，单目标优化问题的建模存在局限性，因此本研究结合边缘智能中的多个性能指标，提出了基于目标权重的多目标优化建模方式，目标方程如式(1)所示。式(1)中，优化目标为准确率和推理时间的加权和，其中推理时间和计算平台的相关度更高，准确率指标与模型相关性更强，μ_1和μ_2分别代表各优化目标的权重，N表示归一化函数，用于将多个维度的数据归一化到同一计算范围内，$A_{\min}$和$L_{\max}$分别代表所有模型中最低的准确率和最长的推理时间，这些常量将被进行归一化处理。约束条件如式(2)所示，内存上限和功耗上限都与计算平台系统资源有关。总而言之，该建模方式使得性能优化方案可根据用户的实际需求进行动态调整，优化目标和约束条件也可以动态追加。

$$\operatorname{argmax}\ \mu_1 N(A(m)-A_{\min})+\mu_2 N(L(m)-L_{\max}),\quad m\in models \tag{1}$$

$$\text{s.t.}\ \begin{cases}\mu_1+\mu_2=1\\ E(m)\leqslant energyLimit\\ M(m)\leqslant memLimit\end{cases} \tag{2}$$

4.2 模型自适应选择策略

本文提出了基于目标权重的模型选择策略，该策略主要包括预选、优选和选定3个阶段。其中，预选阶段主要负责根据约束策略对模型进行筛选，如根据计算平台的内存资源约束过滤掉内存占用高的模型；优选阶段则根据多种单优化目标的模型评估算法对模型进行量化评估；选定阶段则根据设定的目标权重对模型进行综合评估。该策略的流程图如图3所示，其中预选和优选阶段都可以根据实际需求对策略进行扩展，即当有新的约束条件和新的优化目标时，只需要编写符合框架接口的函数即可将匹配逻辑注入到模型选择过程中。

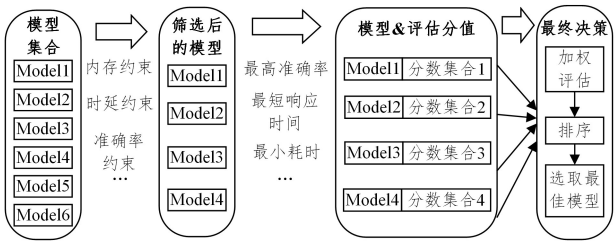


图3 模型自适应选择策略流程示意图

Fig. 3 Schematic diagram of model adaptive selection strategy process

预选阶段主要负责根据约束条件对模型进行筛选，最终留下满足约束条件的模型。具体的执行逻辑是针对每一个模型，都执行由约束构成的预选算法。以内存约束为例，模型的预选过程如算法1所示。该算法的输入为模型，输出为该模型在约束条件下的判断结果。具体的过程如下：首先，获取节点可用内存资源；其次，获取模型内存资源需求，既可通过历史运行信息获得，又可通过访存计算公式获得；最后，通过比较模型内存资源需求和节点可用内存资源的数值来判断模型是否满足约束条件。除了内存约束以外，其他的约束条件也被构建成类似的预选策略。

算法1 基于内存约束的预选算法

输入：待判断算法模型 model，计算平台 node，历史运行数据 history

输出：模型与内存约束的匹配结果 isMatch

```
availableMem←getNodeMem(node)
if hasHistory(model,history)is true then //是否存在历史运行数据
    usageMem←getUsageMem(model,history) //模型内存资源需求
else // 根据模型复杂度预估模型访存量
    usageMem←getEstimateMem(model)
endi f
if usageMem>=availableMem then
    isMatch←false
else
    isMatch←true
end if
```

优选阶段主要负责基于一定的评估指标和算法对模型进行量化评估，不同的评估算法给模型评估的分数也各有高低。优选阶段存在很多优选策略，包括默认策略和根据用户需求定制的评估策略，此处将举例详细说明其中的一个策略。以时延为优化目标的优选策略过程如算法2所示，该算法的输入model是待评估的模型，输出score是该模型的量化评估值，该值被归一化为0到100区间内的任意值。由于不同性能指标的数据范围和量化单位是不同的，往往不具备可比性，例如时延往往以ms为单位，而能耗则以J为单位。因此，对于所有的优选策略，都需要将最后的评估结果进行归一化处理，以保证不同策略可以结合起来进行综合评估。算法的执行逻辑主要分为两方面，一方面如果模型有历史运行数据，那么就可以从历史运行数据中获取当前模型的推理时间、所有运行模型的最长推理时间和最短推理时间，然后通过归一化公式对该模型进行评估；另一方面，如果模型没有历史运行数据，那么将从模型复杂度中获取该模型的计算量、所有模型中最大的计算量和最小的计算量，通过归一化评估公式进行评估。综上所述，优选阶段的策略都是针对某一个方面对模型进行评估，评估的结果均被限定在[0,100]的区间内，评估分数将影响最终选定的模型。

算法2 时延优先的模型量化评估算法

输入：待评估模型 model，历史运行数据 history，模型元信息 modelMeta

输出：评估结果 score

```
Initialize score
maxModelScore←100
if hasHistory(model,history) is true then //判断是否存在历史数据
    tMax←getMaxLatency(history)
    tMin←getMinLatency(history)
    t←getModelLatency(modek,history)
    score←(1-(t-tMin)/(tMax-tMin)) * maxModelScore
else //根据模型复杂度进行量化评估
```

```
9.  f Max←getMaxFlops(modelMeta)
10. f Min←getMinFlops(modelMeta)
11. f←getModelFlops(model,modelMeta)
12. score←(1−(f-fMin)/(fMax-fMin)) * maxModelScore
13. end if
```

选定阶段将结合每个优化目标的权重进行加权评估。具体的逻辑如算法 3 所示，预选算法以优选阶段的处理结果 scoreMap 和优化目标的权重映射 weight 为输入，以经过加权求和之后的评估结果最高的模型名为输出。算法 3 的主要逻辑分为 3 部分：首先，遍历 scoreMap 映射集合，取出模型在不同优化策略下的评估分值；然后，将当前模型各个策略下的评估分值进行加权求和，并将最终得到的总分值记录下来；最后，选择评估结果最高的模型作为当前请求的算法模型。此阶段最重要的步骤是结合优化目标权重，调整评估结果。权重列表的存在使得算法更具通用性，可适配更广泛的用户需求，即使用户的需求复杂而多变，也可以通过调整权重列表适配需求。当用户只需要优化一个目标时，可以通过将其他优化策略的权重置为 0 来适配需求并简化计算。

算法 3 基于目标权重的模型选择策略

```
输入：量化评估结果集合 scoreMap，目标权重键值存储 weight
输出：推荐的算法模型 suggestModel
1. InitializesuggestModel,finalScore
2. for model,result in scoreMap do
3.    totalScore←0
4.    for policy,socre in result do
5.       totalScore←score * weight[policy]+totalScore
6.    end for
7.    finalScore[model]←totalScore
8.    end for
9. suggestModel←getMaxScoreModel(finalScore)
```

本节提出了一种基于目标权重的模型选择策略，通过预选、优选和选定 3 个过程达到了多目标优化的平衡，尤其是在预选和优选阶段都设置了可扩展的方式，预留的扩展接口为调整优化目标和约束条件带来了更高的灵活性。在选定阶段预留了调整权重列表的方式，用于平衡多个性能指标的取舍，使模型更具通用性。

5 基于开销反馈的批量自适应调整算法

边缘智能负载推理阶段存在一些影响模型性能的参数。以图像分类的应用场景为例，批量的调整会影响模型的能耗、内存占用以及 FPS 等指标。具体而言，当批量增加时，FPS 会逐渐扩大，但是系统的能耗和内存资源占用也会显著增加，因此每次调整批量所带来的开销主要是能耗开销，收益是 FPS 的增加。同时，批量调整应一直在系统内存资源的约束下进行，因此批量自适应调整的核心问题是分配合适的批量来平衡 FPS 和系统能耗，并满足计算节点内存资源约束。

为了解决模型运行时存在的批量自适应问题，本节提出了基于开销反馈的批量自适应调整算法，具体过程如算法 4 所示。该算法的输入 modelHistory 为模型历史运行数据，是一个双层映射，可以取出对应批量下的性能指标数据；算法的输出是最终建议的批量。算法 4 的具体运行逻辑分为 3 部分：首先，初始化内存上限、批量、开销和收益；然后，在内存上限约束下进入循环，计算当前批量下的收益和开销的变化百分比；最后，在每次迭代中比较收益和开销变化比例，如果收益小于开销，则跳出循环。总之，通过搜索历史运行信息找到最适合模型运行时的批量是算法 4 的核心。

算法 4 基于开销反馈的批量自适应调整算法

```
输入：模型历史运行信息 modelHistory
输出：建议的模型运行时批量 suggestBatchSize
1. memLimit←getAvailableMem() //获取当前可用的内存信息
2. preCost←getMinEnergeCost() //获取初始的能耗开销
3. preBenefit←getMinFPS() //获取初试收益
4. batchSize←1 //初始化运行时批量
5. while true do
6.    mem←modelHistory[batchSize][mem] //获取内存消耗
7.    cost←modelHistory[batchSize][power] //获取能耗开销
8.    benefit←modelHistory[batchSize][FPS] //获取收益
9.    costPercent←(cost/precost−1) * 100
10.       benefitPercent←(benefit/preBenefit−1) * 100
11.   preCost,preBenefit←cost,benefit
12.       if benefitPercent<costPercnet or mem>memLimitthen
13.     break
14.   end if
15.   batchSize←update(batchSize) //更新批量
16. end while
17. suggestBatchSize←batchSize //保存最终合适的批量
```

6 基于 SLA 感知的云边协同策略

6.1 任务失效判定机制

云边协同中最重要的问题是判断任务应部署在边缘端执行，还是上传到云端执行。本研究的策略是将失败的任务部署至云端执行，这样做一方面可以避免上传所有任务带来的超长传输时间和大量带宽消耗，另一方面也可以充分利用云端的算力运行更复杂的模型来提升准确率。

然而，就图像分类任务而言，由于用户新输入的图片缺乏正确的标签，程序并不能自动判断此次推理是否正确。针对失败任务难以判断的问题，本研究使用 SM(Score Margin)指标协助判断任务正确与否，如式(3)所示：

$$SM = 1^{st}score - 2^{nd}score \tag{3}$$

其中，$1^{st}score$ 代表算法模型输出的张量中分类概率最大的值，而 $2^{nd}score$ 代表算法模型输出的张量中分类概率第二大的值。当 SM 足够大时，就意味着模型推理结果中，有更大的把握确定概率第一大的结果是正确的结果；而当 SM 较小时，模型推理结果中概率第一大和次大的结果差异不大，说明算法模型并不确定哪一个分类是正确的，分类预测出现错误的概率较高。本研究使用 SM 来估计此次推理结果正确与否，通过设定一个阈值来进行具体的逻辑判断。如式(4)所示，当 SM 的值小于阈值时，将判定结果置为 1(表示推理结果错误)；当 SM 的值大于或等于阈值时，将判定结果置为 0(表示推理结果正确)。

$$IsFailed = \begin{cases} 0, & SM \geqslant \theta \\ 1, & SM < \theta \end{cases} \tag{4}$$

对于边缘智能负载而言，其优化目标可以具体量化为

最小化时延、最大化准确率和最小化能耗的多目标优化。为了解决时延和能耗都与运行环境相关性的问题，本文使用收益函数来量化时延和能耗的性能优化效果。如式(5)所示，其中 N 代表样本总数量，$IsFailed_i$ 为第 i 个输入样本的分类结果。$Gain$ 取值越大，代表云端执行的任务数量越多，也意味着更高的时延和能耗；$Gain$ 取值越小，则意味着更少的时延和能耗。式(6)为云边协同模式下的准确率收益的数学模型，其中 Acc_{Cloud} 和 Acc_{Edge} 分别表示云端和边缘端成功推理的准确率。当云端推理的任务数量增加时，准确率收益会增加；相反，当任务数量减少时，准确率收益则会减少。

$$Gain=\sum_{i=1}^{N} IsFailed_i \tag{5}$$

$$AccBenefit=\frac{Gain}{N}*Acc_{\text{Cloud}}+\frac{N\text{-}Gain}{N}*Acc_{\text{Edge}} \tag{6}$$

6.2 云边协同策略

本研究设计的云边协同策略架构如图 4 所示，该策略通过预判算法模型的推理结果和感知 SLA(此处指时延和准确率)来确定需要卸载到云端运行的任务量。举例而言，给定一个输入到边缘设备，那么边缘设备会根据模型选择策略匹配最合适的算法模型来运行；然后，使用基于 SLA 感知的云边协同策略对模型的推理结果进行决策；最后，根据决策结果将适当比例的任务发送到云端推理执行。但使用云端的计算资源意味着要承受任务和处理结果往返的时间、能耗以及复杂网络环境带来的数据包丢失后果，因此应结合具体场景进行分析。

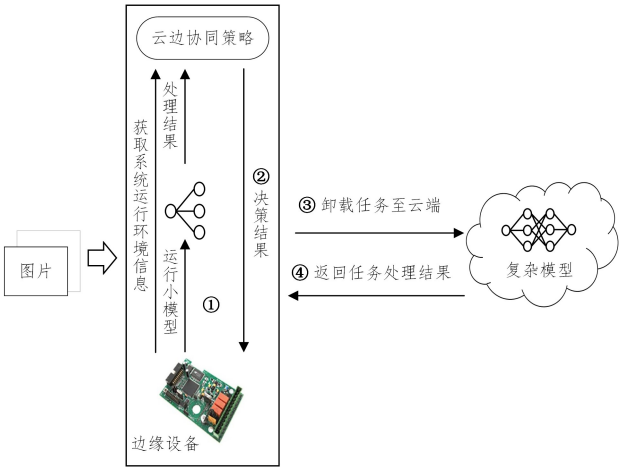


图 4 云边协同框架

Fig. 4 Framework of cloud edge collaboration

基于 SLA 感知的云边协同策略如算法 5 所示。首先，该算法运行选中的模型进行首次推理，将推理得到的张量结果记为 output。其次，遍历每一个输出结果，利用最大概率值和次大概率值之间的差来计算该推理结果的 smScore 值。然后，在遍历过程中进行判断，如果 smScore 大于阈值，那么说明该样本是失败概率更大的样本，将样本加入到失效集合中。最后，基于 SLA 进行云边协同决策，具体的决策逻辑是根据 smScore 要求的准确率和时延动态决策是否需要借助云端的计算能力来确定的。如果当前准确率满足了 accSLA，那么完全无需云端处理；如果准确率违反了 accSLA，那么就需要借用云端的计算能力。而云端计算往往需要考虑数据和结果在网络中传输的往返时延，根据设定的 timeSLA 和历史运行数据来决定最终发送至云端的数据量。

算法 5 基于 SLA 感知的云边协同策略

输入：边缘设备所选定的算法模型 model，数据集 dataSet，历史信息 history

输出：需要上传到云端的数据 uploadData

```
1. Initialize failedCases
2. output←model(dataSet) //当前算法模型的输出结果
3. for img,o in output do //遍历每一张图片的计算结果
4.    smScore←max(o)-secondMax(o)
5.    if smScore>θ then
6.       add img into failedCases
7.    endif
8. endfor
9. curAcc←failedCases/len(dataSet) //计算准确率
10. if curAcc>accSLA then
11.   uploadData←None
12.   exit
13. endif
14. uploadData←selectUploadData(failedCases,history,timeSLA)
```

上述工作对云边协同中存在的问题进行了分析与建模，同时提出了基于 SLA 感知的云边协同策略，旨在满足用户需求的前提下提高服务性能。在云计算范式下，人工智能算法往往部署在计算资源充足的云端，然而，网络距离较远时，云智能服务的体验将迅速降级。因此，利用云端和边缘端的优势，设计有效的云边协同策略对边缘智能而言有一定的应用价值。

7 边缘智能性能优化方案评估

7.1 实验环境

7.1.1 硬件平台搭建

本研究选取 4 台异构的计算平台搭建实验环境，使用 POWER-Z KM001 来测量边缘设备的能耗数据。这些计算平台包括适用于图像处理的 GPU 计算平台、处理通用任务的 CPU 计算平台、资源受限的树莓派以及提供边缘计算能力的 Jetson Nano 开发平台。为了模拟边缘计算实验环境，本研究将强力的 GPU 和 CPU 计算节点视为云中心的计算节点，而将资源受限的树莓派和 Jetson Nano 视作边缘计算节点。表 2 列出了相关计算节点的具体硬件配置信息，包括系统版本、处理器型号、核心数量、是否支持显卡、内存和硬盘的容量等。

表 2 节点硬件配置表

Table 2 Node hardware configuration

节点类型	系统版本	处理器	核数	显卡	内存/显存/GB	硬盘/GB
CPU 节点	CentOS 7.7	Intel Xeon E5-2620 v4	32	—	48	1100
GPU 节点	CentOS 7.9	Intel Xeon E5-2620 v2	24	NVIDIA K40	12	2000
Jetson Nano	Ubuntu 18.04	ARM Cortex-A57	4	NVIDIA Maxwell	4	32
树莓派	Raspbian 10	ARM Cortex-A72	4	—	4	16

7.1.2 算法模型与数据集

本研究采用计算机视觉领域中使用范围最广的 ImageNet[27]数据集作为实验所需的数据集。该数据集在 2010—2017 年这 8 年间一直作为 ImageNet 大规模视觉识别挑战(ImageNet Large-Scale Visual Recognition Challenge, ILSVRC)比赛的数据集。ILSVRC 比赛所用数据集共有 1000 类图像,总共有 1.2 亿张训练图片、5 万张验证图片和 15 万张测试图片。鉴于本研究的主要内容是边缘智能推理阶段的分析和性能优化工作,因此选择了 ILSVRC-2012 数据集中的验证集进行研究。

为了使实验所使用的模型具有代表性,在挑选算法模型的过程中,本研究使用结合多个指标综合考虑的差异化挑选方法,具体过程如下:

(1)从 Github 收集并复现经典的算法模型,共计 31 个算法模型。将这些模型应用到数据集中,测量得到准确率。同时,编写代码分析模型的复杂度,包括模型的计算量和参数量。

(2)将所有模型分别按照准确率、计算量、参数量 3 种排序标准进行排序,选取准确率最高的 5 个模型、计算量最小的 5 个模型、参数量最小的 5 个模型。

(3)取 3 个模型的并集作为新的模型结合,并完成去重操作。去重的过程主要是去除集合中共同的模型和算法模型相同但版本不同的模型。

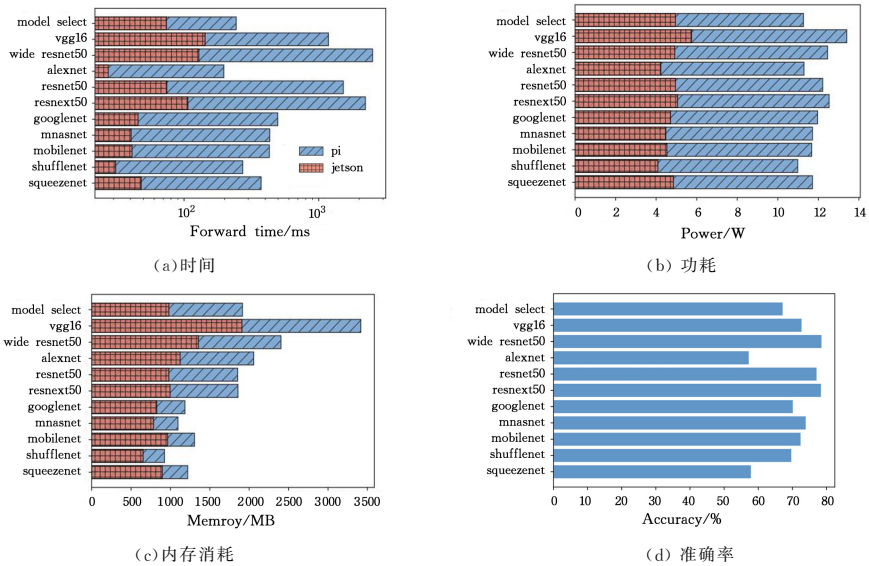
(4)加入历年比赛中的经典模型,主要是比赛中排名靠前的算法模型。

本实验中所用算法模型包括 AlexNet, GoogleNet, VGG16, ResNet50, ResNeXt50, SqueezeNet, Wide ResNet50, ShuffleNet, MASNet 和 MobileNet。

7.2 模型选择策略性能评估

本小节将针对提出的模型选择策略进行性能评估。评估过程将对模型选择策略与固定模型的方法在时间、功率、内存消耗和准确率方面的结果进行比较。

模型选择策略相比固定使用模型的方法在推理时间、功耗、准确率以及内存消耗方面都有一定的性能提升。如图 5 所示,图 5(a)—图 5(c)这 3 个子图表示固定模型和模型选择策略在 Jetson 和树莓派这两个计算平台上的性能表现,图 5(d)则表示固定模型和模型选择策略在两个平台上的平均准确率。从推理时间指标来分析,AlexNet 模型是推理速度最快的模型,两个平台的总推理时间达到了 197 ms,而 Wide ResNet50 模型是推理速度最慢的模型,推理时间达到了 2512 ms。使用模型选择策略相比单独使用 Wide ResNet50 模型能够降低 90%的推理时间,性能表现优于固定模型中 90%的模型。从功耗角度来分析,模型选择策略比单独使用 VGG16 模型降低了 16%的功耗,比单独使用 ShuffleNet 模型增加了2.5%的功耗,性能表现优于固定模型中 90%的模型。这是由于模型选择策略会挑选更契合计算平台的模型,减少使用复杂模型所带来的功耗。从内存使用来分析,模型选择策略所挑选的模型内存占用总量达到1914 MB,比单独使用 VGG16 模型内存消耗降低了 56%,优于固定模型中 30%的模型。从准确率方面来分析,模型选择策略所挑选的模型的准确率可以达到 67.15%,相比单独使用 AlexNet 模型,准确率提升了 17%,整体优于固定模型中 20%的模型。综上所述,模型选择策略在结合多个指标考虑时,能达到很好的效果。即在时延、内存等条件约束下,在增加准确率的同时,达到了降低功耗和推理时间的效果。这种设计思路适用于当前用户需求复杂多变的边缘智能场景。



(a)时间 (b) 功耗

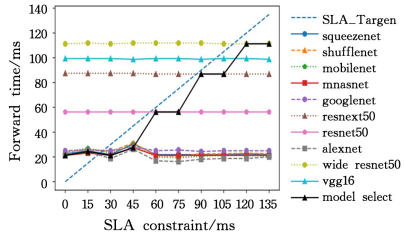
(c)内存消耗 (d) 准确率

图 5 固定模型与模型选择策略性能对比实验

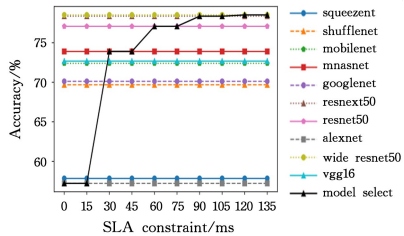
Fig. 5 Performance comparison experiment between fixed model and model selection strategy

模型选择策略与使用固定模型算法在时延约束下的性能对比结果如图 6 所示。图 6 中,横坐标表示设定不同的 SLA 约束;纵坐标表示具体的性能指标;不同颜色的线条代表不同的算法模型。由图 6(a)可知,模型选择策略能够感知 SLA 约束的调整,进而动态地选择最合适的算法模型。在整个过程中,该算法的 SLA 违约率只有 20%,超过固定模型中 40%的模型。同时,该算法与违约率最高的 VGG16 模型相比,降低了 75%的违约率,与违约率最低的模型 SqueezeNet 的违约率相同。观察图 6(b)可知,当 SLA 目标设定得较低时,模型选择策略会牺牲模型精度,选择速度更快的算法模型。在 SLA

约束下，模型选择策略的准确率达到了 74.37%，比准确率最高的算法模型 VGG16 提升了 2.5%的准确率，比准确率最低的算法模型 SqueezeNet 提升了 38.4%的准确率。并且在 SLA 约束严格时，模型选择策略最多会牺牲 12.5%的准确率，用于保证推理任务遵守 SLA 约束。综上所述，本研究所提出的模型选择策略在 SLA 约束动态调整的情况下，具有更好的灵活性，可以根据实际性能需求选择更合适的算法模型来处理任务。



(a)推理时间



(b) 准确率

图 6 固定模型与模型选择策略约束限制实验

Fig. 6 Constraint restriction experiment of fixed model and model selection strategy

7.3 批量自适应调整算法性能评估

本节实验设计主要从两方面来验证批量自适应调整算法的有效性：一方面，验证算法是否能根据其运行环境实时调整；另一方面，评估自适应批量与固定模型批量的性能提升效果。

将批量自适应调整算法应用于不同节点的多个模型时，该算法的实际效果如表 3 所列。

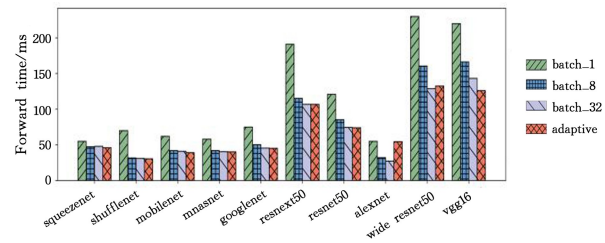
表 3 批量自适应调整算法结果展示

Table 3 Batch adaptive adjustment algorithm result display

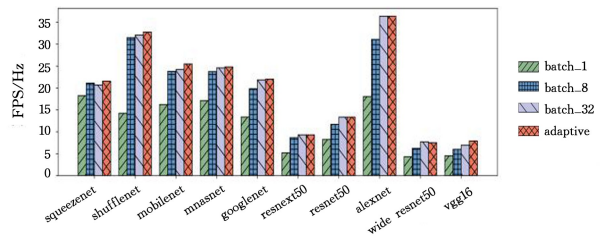
模型	Jetson	树莓派	CPU 节点	GPU 节点
AlexNet	32	128	512	32
GoogleNet	16	64	16	64
VGG16	16	32	512	128
ResNet50	32	64	8	512
ResNeXt50	32	32	4	128
SqueezeNet	16	32	16	32
Wide ResNet50	16	32	8	256
ShuffleNet	16	128	32	128
MNASNet	16	256	32	64
MobileNet	16	128	4	128

表 3 中，第一列表示所运行的模型集合，其余每一列表示各种硬件平台下最佳的运行批量。表 3 中的每一行表示该模型在各个计算平台下的最佳模型批量。对表 3 中的数据进行横向分析可知，模型 ResNet50 在 Jetson、树莓派、CPU 节点和 GPU 节点的最佳批量分别为 32，64，8，512，故同一模型在不同节点上可以自适应调整运行时批量。对表 3 中数据进行纵向分析可知，在 GPU 节点上，模型间存在不同的最佳运行时批量，其中模型 ResNet50 的运行时批量是最大的，为 512，而模型 AlexNet 的批量是最小的，为 32，前者是后者的 16 倍。综上所述，批量自适应算法可以有效地根据执行环境来动态调整运行时批量。

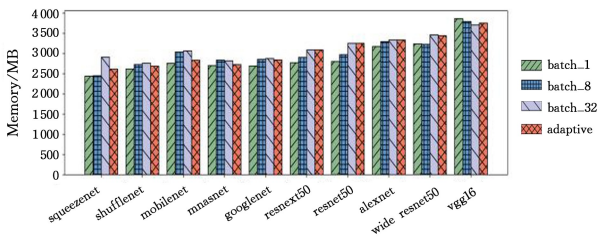
动态调整模型批量比固定模型批量更具灵活性，同时也在多个性能指标上表现良好，如图 7 所示。图 7(a)给出了推理时间的变化情况，其中批量自适应算法的平均推理时间达到了 69 ms，比将 batchSize 设定为 1 的情况缩短了 39%的推理时间，比将 batchSize 设定为 32 的情况缩短了 2%的推理时间。图 7(b)给出了 FPS 的性能优化结果，批量自适应算法的平均 FPS 达到了 20.11 Hz，优于图中所有固定 batchSize 的方法，比固定为 1 的方法提升了 68%的效果。图 7(c)给出了内存使用情况，批量自适应算法的平均内存消耗达到了 3 057 MB，该结果比固定为 32 的方法减少了 2%的内存使用量，比固定为 1 的方法增加了 5%的内存使用量。综上所述，本文提出的批量自适应算法最多提升了 68%的 FPS，并降低了 39%的推理时间，为边缘智能负载带来了性能提升。



(a)推理时间



(b)FPS



(c)内存使用

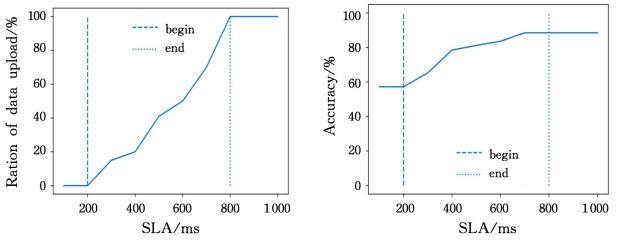
图 7 批量自适应算法性能优化结果

Fig. 7 Batch adaptive algorithm performance optimization results

7.4 云边协同策略性能评估

为了评估云边协同策略的有效性，本节将从 3 个方面进行实验。首先评估云边协同策略是否能够根据不同的 SLA 来动态调整数据传输量；然后评估单个边缘节点上云边协同策略所带来的性能变化；最后分析多个边缘节点的场景下云边协同策略带来的性能变化。

云边协同策略会感知 SLA 的变化而动态地调整向云端卸载的任务量，评估实验结果如图 8 所示。观察图 8 可知，SLA 初始阶段的约束较为严格，该策略不会考虑将推理任务卸载到云端执行，然而随着 SLA 标准逐渐放宽，该策略会感知到 SLA 约束的变化，结合网络时延，将失败任务发送到云端继续执行。当 SLA 的约束大于 200 ms 时，云边协同策略开始逐步上传边缘端推理失败的任务到云端，当 SLA 的约束大于 800 ms 时，该策略将上传所有失败任务到云端。由图 8(b)可知，当只有边缘端参与运算时，准确率只有 57.22%，而随着 SLA 约束逐步放宽，云端也可以参与部分失败任务的运算，准确率逐步提升。当所有失效任务都卸载到云端时，准确率高达 85.52%，相比只有边缘端参与运算提升了 49%。准确率提升正是因为借助了云端的算力资源运行高精度的算法模型。综上所述，该策略能够根据 SLA 的变化，动态地调整云端协同的任务量，在尽量满足 SLA 约束的情况下获得更高的准确率。

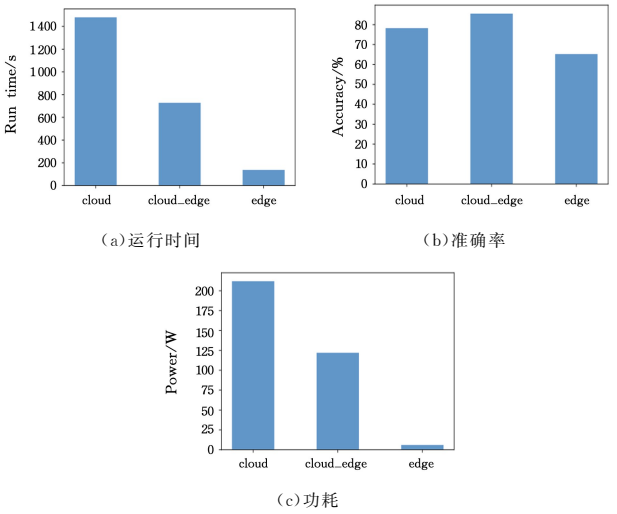


(a)数据上传百分比 (b) 准确率

图 8 云边协同策略 SLA 评估

Fig. 8 SLA evaluation of cloud edge collaboration strategy

云边协同策略与单独在边缘节点运行推理任务、将数据发送到云端执行推理任务的对比实验的结果如图 9 所示。



(a)运行时间 (b)准确率

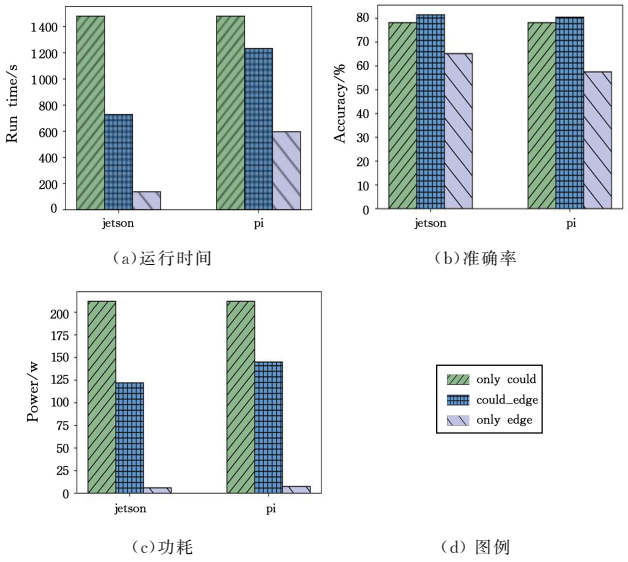
(c)功耗

图 9 单节点云边协同策略性能评估

Fig. 9 Performance evaluation of single-node cloud-edge collaboration strategy

观察图 9 中的数据可知，完成同一批任务，云边协同策略所需的总时间为 728.16s，比云端处理缩短了 50%，比边缘端处理增加了 4 倍的时间。这主要是由于云端处理和云边协同都需要向云端发送数据，相比边缘端处理会产生较高的总运行时间。在准确率方面，云边协同策略的准确率为 85.52%，比边缘端处理高 49%。在功耗方面，云边协同处理的峰值功耗小于将全部数据放置在云端处理所带来的功耗。综上所述，云边协同策略借助将失败任务发送到云端执行这一策略为边缘智能带来了性能提升，相比云端处理，它带来了更低的功耗和更快的实时响应；相比边缘端处理，它带来了更高的准确率。

为了探究云边协同策略在不同节点上的适用性，实验中将该策略分别应用于多个边缘节点，该实验结果如图 10 所示。由图 10 可知，在 Jetson 和树莓派计算平台上，云边协同策略的运行时间和瞬时功耗都比云端处理要低，准确率都比边缘端处理高。综上所述，即使将该云边协同策略应用于不同的边缘节点，该策略也能为边缘智能带来性能提升，具有普适性。



(a)运行时间 (b)准确率

(c)功耗 (d) 图例

图 10 多节点云边协同策略性能评估

Fig. 10 Performance evaluation of multi-node cloud edge collaborative strategy

7.5 框架综合性能评估

本小节将评估 CECI 框架的整体性能，评估主要侧重于两方面，一方面是该框架整合模型选择策略、批量自适应算法以及云边协同策略之后所带来的整体性能提升效果；另一方面则是该框架在目标权重调整时，是否具有足够的灵活性来响应权重调整。

CECI 框架是一个多目标性能优化框架，为了评估该框架的性能表现，在实验过程中将多个性能目标的权重进行固定。如式(1)所示，实验中分别设定$\mu_1=0.75$，$\mu_2=0.25$，表示优先考虑准确率，其次考虑运行时间。在固定权重后，该框架的性能表现如图 11 所示。观察图 11 中的数据可知，使用 CECI 框架后，边缘智能在各方面均略优于云智能。在 Jetson 节点上，边缘智能的准确率相比云智能提升了 4.52%，总运行时间缩短了 50.79%，能耗相比云智能降低了 42.46%；在树莓派节点上准确率提升了 2.37%，运行时间缩短了 16.6%，能耗降低了 31.67%。综上所述，CECI 框架能够有效结合边缘端处理的实时性与云端处理的高精度为边缘智能场景下的

应用带来更好的性能提升。

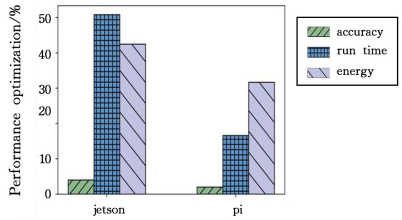
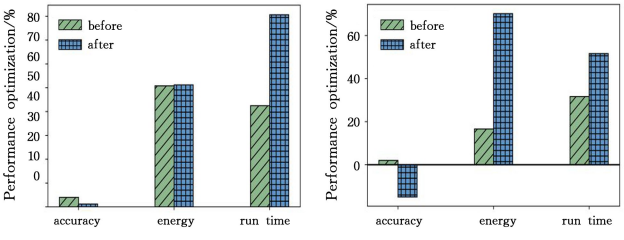


图 11 CECI 框架的性能评估

Fig. 11 CECI framework performance evaluation

CECI 框架被设计为能够针对目标权重的调整而自适应改变的框架，为了评估这种灵活性，实验过程中将目标权重更改为$\mu_1=0.25$，$\mu_2=0.75$。图 12 给出了调整目标权重后的各性能指标的变化，纵轴表示性能指标的优化效果。从图 12 中可以观察到，在调整任务需求后，CECI 框架的性能优化趋势发生了变化。具体而言，在 Jetson 平台上，准确率从 4.52% 的优化值降低到了 1.2%，能耗优化效果无显著变化，运行时间的优化效果从 42.46% 提升至 80.62%。而在树莓派计算平台上，准确率的优化效果从 2.37% 降低到了 −15%，能耗优化效果提升至 70.17%，运行时间优化效果提升至51.62%。产生这一结果的原因是提升了时间的目标权重，树莓派在模型选择阶段选取了快速但准确率低的模型；在云边协同阶段，放弃卸载数据到云端，以追求在本地快速地完成任务。综上所述，CECI 框架在面对目标权重调整的应用场景时，可以很好地调整框架中各方法的优化目标，从而达到一定的自适应性。



(a)Jetson 平台 (b)树莓派平台

图 12 CECI 框架在目标权重调整时的性能表现

Fig. 12 Performance of CECI framework when target weight is adjusted

结束语 为了解决边缘智能负载的性能优化问题，本文提出了云边协同智能推理框架(CECI)。该机制针对模型选择、批量自适应调整和云边协同这 3 个问题，分别设计了优化策略和算法。具体而言，针对边缘设备上的模型选择问题，提出了基于目标权重的模型选择策略；针对模型运行时存在的批量自适应调整问题，提出了基于开销反馈的自适应调整方案；针对云边协同计算的问题，提出了基于 SLA 感知的云边协同策略。为了验证所提策略的有效性，本文验证了各种算法的优化效果。其中，模型选择策略能够根据边缘节点特征为设备选择最合适的算法模型，同时还可以根据资源约束和需求约束调整最终的算法。批量自适应算法可以根据运行环境实时调整批量，并且相比固定批量的方法可以获得更高的性能提升。云边协同策略可以根据 SLA 约束的调整，动态地改变卸载到云端的任务量，并且该策略具有普适性，在不同的边缘节点上都有较好的性能提升。

由实验结果表明，本文设计的方案在边缘智能场景下与云智能相比，最多可以降低 50.79% 的运行时间，降低 42.46% 的能耗，提升 4.52% 的准确率，并且可以根据目标权重的变化自适应调整优化策略。

参 考 文 献

[1] LU Y，CHEN Y，LI T，et al. Embedded FPGA convolutional neuralnetwork construction method for edge computing[J]. Journal of Computer Research and Development，2018，55(3)：551-562.

[2] DWORK C，MCSHERRY F，NISSIM K，et al. Calibrating noise to sensitivity in private data analysis[C]// Theory of cryptography conference. 2006：265-284.

[3] FREDRIKSON M，JHA S，RISTENPART T. Model inversion attacks that exploit confidence information and basic countermeasures[C]// Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015：1322-1333.

[4] DU M，WANG K，XIA Z，et al. Differential privacy preserving of training model in wireless big data with edge computing[J]. IEEE transactions on big data，2018，6(2)：283-295.

[5] LIU T T，YANG C Y，SUO S Q，et al. Edge Intelligence in Wireless Communication[J]. Signal Processing，2020，36(11)：1789-1803.

[6] REN J，GAO L，YU J L，et al. Energy-efficient deep learning task scheduling strategy for edge devices[J]. Chinese Journal of Computers，2020，43(3)：440-452.

[7] BHATTACHARYA S，LANE N D. Sparsification and separation of deep learning layers for constrained resource inference on wearables[C]// Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM. 2016：176-189.

[8] VALERIO L，PASSARELLA A，CONTI M. A communication efficient distributed learning framework for smart environments[J]. Pervasive and Mobile Computing，2017，41：46-68.

[9] HSIEH K，HARLAP A，VIJAYKUMAR N，et al. Gaia：Geo-distributed machine learning approaching speeds[C]// 14th Symposium on Networked Systems Design and Implementation (NSD). 2017：629-647.

[10] OGDEN S S，GUO T. MODI：Mobile deep inference made efficient by edge computing[C]// Workshop on Hot Topics in Edge Computing(HotEdge 18). 2018：30-36.

[11] HU C，BAO W，WANG D，et al. Dynamic adaptive DNN surgery for inference acceleration on the edge[C]// IEEE INFOCOM 2019-IEEE Conference on Computer Communications. 2019：1423-1431.

[12] ZHAO Z，BARIJOUGH K M，GERSTLAUER A. DeepThings：Distributed adaptive deep learning inference on resource-constrained IoT edge clusters[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems，2018，37(11)：2348-2359.

[13] SHENG T，FENG C，ZHUO S，et al. A quantization-friendly

separable convolution for mobilenets[C]// 2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications(EMC2). 2018:14-18.
[14] SANTOS A G, DE SOUZA C O, ZANCHETTIN C, et al. Reducing squeezenet storage size with depthwise separable convolutions[C]// 2018 International Joint Conference on Neural Networks(IJCNN). 2018:1-6.
[15] CERUTTI G, PRASAD R, BRUTTI A, et al. Neural network distillation on IoT platforms for sound event detection[C]// Interspeech. 2019:3609-3613.
[16] YAO S, ZHAO Y, ZHANG A, et al. DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework[C]// Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems. 2017:1-14.
[17] HAN S, KANG J, MAO H, et al. Ese: Efficient speech recognition engine with sparse lstm on fpga[C]// Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2017:75-84.
[18] LIU S, LIN Y, ZHOU Z, et al. On-demand deep model compression for mobile devices: A usage-driven model selection framework[C]// Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. 2018:389-400.
[19] HARDY C, LE MERRER E, SERICOLA B. Distributed deep learning on edge-devices: Feasibility via adaptive compression [C]// 2017 IEEE 16th International Symposium on Network Computing and Applications(NCA). 2017:1-8.
[20] DEAN J, CORRADO G, MONGA R, et al. Large scale distributed deep networks[C]// Advances in Neural Information Processing Systems. 2012:1223-1231.
[21] YU Z, HU J, MIN G, et al. Federated learning based proactive content caching in edge computing[C]// 2018 IEEE Global Communications Conference(GLOBECOM). 2018:1-6.
[22] WANG S, TUOR T, SALONIDIS T, et al. When edge meets learning: Adaptive control for resource-constrained distributed machine learning[C]// IEEE INFOCOM 2018-IEEE Conference on Computer Communications. 2018:63-71.
[23] FANG B, ZENG X, ZHANG M. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision [C]// Proceedings of the 24th Annual International Conference on Mobile Computing and Networking. 2018:115-127.
[24] TAYLOR B, MARCO V S, WOLFF W, et al. Adaptive deep learning model selection on embedded systems[J]. ACM SIGPLAN Notices, 2018, 53(6):31-43.
[25] KANG Y, HAUSWALD J, GAO C, et al. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge[J]. ACM SIGARCH Computer Architecture News, 2017, 45(1):615-629.
[26] MAO J, CHEN X, NIXON K W, et al. MODNN: Local distributed mobile computing system for deep neural network[C]// Design, Automation & Test in Europe Conference & Exhibition (DATE). 2017:1396-1401.
[27] ImageNet website update [OL]. 2021-06-28. https://www.image-net.org/.

HU Zhao-xia, born in 1997, postgraduate. Her main research interests include data center scheduling, and edge computing.

JIANG Cong-feng, born in 1980, Ph.D, professor, is a member of China Computer Federation. His main research interests include edge computing, system optimization, performance evaluation and distributed system benchmarking.

(责任编辑:喻藜)