中图分类号：TP391　　论文编号：
学科分类号：520.20　　密　　级：公　开

安徽理工大學

专业学位硕士学位论文

（全日制）

基于深度强化学习的DNN协同推理加速方法研究

作者姓名：许浩
专业名称：计算机技术
研究方向：边缘智能
导师姓名：朱晓娟　副教授
导师单位：安徽理工大学
答辩委员会主席：孙见山
论文答辩日期：2024 年 6 月 1 日

安徽理工大学研究生院
2024 年 6 月 11 日

A Dissertation in Computer technology

Research on DNN Collaborative Inference Acceleration Method based on Deep Reinforcement Learning

Candidate: XuHao

Supervisor: Zhu Xiaojuan

School of Computer Science and Engineering

Anhui University of Science and Technology

No. 168, Taifeng Street, Huainan, 232001, Anhui, P. R. China

摘　要

随着人工智能研究的日益深化，深度神经网络（Deep Neural Networks, DNN）的模型层次不断加深，虽有效提高了众多智能应用的推理精准度，但对算力的需求也显著提升。将推理任务交由云端执行会产生较高的数据传输时延，而将任务全部卸载到终端或边缘执行则会面临较大的计算压力。所以，本文主要研究深度神经网络的协同推理加速方法。现有的智能推理加速算法在当网络状态和任务需求快速变化时难以获得全面的实时数据进行准确预测，导致模型的推理性能下降。如何在环境感知、智能算法优化方面进行改进，使协同推理算法能够适应网络状态和任务需求的动态变化，从而实现推理时延、成本等多目标的优化是当前面临的挑战。因此本文在基于软件定义网络（Software Defined Network, SDN）的云边协同推理架构下，使用模型划分与早期退出技术对模型的推理效率进行优化研究，具体研究内容如下：

（1）为解决因网络计算压力大、成本高而导致推理效率低的问题，提出了一种基于模型划分的云边协同推理算法（Cloud-Edge Collaborative Inference Algorithm based on Model Partition, SDN-CEP），首先在边缘环境中构建任务复杂度预测器决策任务的执行环境，然后利用架构中的控制器实现网络状态全局信息的共享，最后将全局视图与 DQN 算法相结合，实现推理模型的有效分配。本文分别从推理时延、网络带宽、计算成本等方面评估算法的性能。实验结果表明，SDN-CEP 提高了模型的推理效率，在动态的环境中具有较好的鲁棒性。

（2）为进一步加快深度神经网络模型的推理速度，提出了一种基于自适应早期退出的推理加速算法（Inference Acceleration Algorithm based on Adaptive Early Exit, AEE），通过缩减模型推理层次的方式来提升推理速度。首先使用 BranchyNet 框架训练模型的早期退出分支，然后利用 NoisyDQN 算法根据全局视图中的实时网络状态信息对模型分支进行自适应预测。最后，通过仿真实验证明 AEE 提高了深度神经网络的推理速度，具有良好的决策性能。

图[25] 表[3] 参[87]

关键词：软件定义网络；云边协同；协同推理；模型划分；早期退出

分类号：TP391；

Abstract

With the deepening of artificial intelligence research, the model level of DNN (Deep Neural Networks) continues to deepen. Although the inference accuracy of many intelligent applications is effectively improved, the demand for computing power is also significantly increased. Transferring inference tasks to the cloud will result in high data transmission delay, while unloading all tasks to the terminal or edge will face greater computing pressure. Therefore, this paper mainly studies the collaborative inference acceleration method of deep neural networks. It is difficult for the existing intelligent inference acceleration algorithm to obtain comprehensive real-time data for accurate prediction when the network state and task requirements change rapidly, which leads to the deterioration of the inference performance of the model. How to improve the environment perception and intelligent algorithm optimization, so that the collaborative inference algorithm can adapt to the dynamic changes of network state and task requirements, so as to achieve multi-objective optimization such as inference delay and cost is the current challenge. Therefore, under the cloud edge collaborative inference architecture based on SDN (Software Defined Network), this paper uses model partitioning and early exit technology to optimize the inference efficiency of the model. The specific research content is as follows:

(1) In order to solve the problem of low inference efficiency caused by high pressure and high cost of network computing, SDN-CEP (Cloud-Edge Collaborative Inference Algorithm based on Model Partition) was proposed. Firstly, the decision execution environment of the task complexity predictor is constructed in the edge environment, and then the global information of the network state is shared by the controller in the architecture. Finally, the global view is combined with the DQN algorithm to realize the effective allocation of the inference model. In this paper, the performance of the algorithm is evaluated from the aspects of inference delay, network bandwidth and calculation cost. The experimental results show that SDN-CEP improves the inference efficiency of the model and has good robustness in dynamic environment.

(2) In order to further accelerate the inference speed of deep neural network models,AEE (Inference Acceleration Algorithm based on Adaptive Early Exit) was

proposed. The inference speed is improved by reducing the inference level of the model. Firstly, BranchyNet framework is used to train the early exit branches of the model, and then NoisyDQN algorithm is used to make adaptive prediction of the model branches according to the real-time network state information in the global view. Finally, simulation results show that AEE improves inference speed of deep neural network and has good decision-making performance.

Figure [25] Table [3] Reference [87]

Key Words： Software defined networking, cloud edge collaboration, collaborative inference, model division, early exit

Chinese Books Catalog:TP391

目 录

插图或附表清单

1 绪论

1.1 研究背景与意义

1.1.1 研究背景

随着智能物联网的发展，物联网设备不断增加，导致边缘数据呈指数级增长[1-2]。目前，网络中的边缘设备通常采用将数据上传到云端的方式进行统一处理。然而，在数据传输过程中，大量智能终端设备接入和海量数据占用网络带宽，导致传输时延增大、安全风险增加。且随着移动设备的更新换代，部署分散的终端设备对计算平台提出了低时延、高带宽等需求。因此，研究者在云计算的基础上提出了边缘计算（Edge Computing, EC）的概念[3-4]，通过将数据下放到网络边缘来提升系统整体的可用性和可拓展性。当前，以云计算为中心的数据处理方式正逐渐向以物联网为核心的边缘计算转变。

边缘计算是将云计算服务从网络核心推向更接近智能终端与数据源的网络边缘[5]。通过在网络边缘部署服务器，形成靠近物联网设备的边缘计算平台，从而可以在网络边缘实时处理终端设备所产生的数据，加速数据流。边缘计算可以减少主干网络数据流量、缓解网络带宽的压力，同时还具有保护数据安全的作用。相较于传统的云计算模式，边缘计算使得服务与数据源之间的距离更接近，降低了数据传输时延和通信能耗，提升了用户服务质量。然而，由于资源有限，可能无法满足所有计算任务的需求。

近年来，人工智能（Artificial Intelligence, AI）再次迎来了发展高潮，无人驾驶、智慧农业、智慧城市等领域都是 AI 发展的延伸。深度学习（Deep learning, DL）作为 AI 最耀眼的领域，在计算机视觉、语音识别、自然语言处理等多个领域被广泛应用。随着算法、算力以及大数据等研究进展的推动，DL 取得了实质性突破，并与强化学习结合，解决了高难度的决策问题，受到人们青睐。然而，AI 算法的实现需要大量计算资源，当前大部分 AI 计算任务部署在云中心或其他大规模计算密集型资源平台上，但考虑到计算平台与智能终端设备的距离以及网络边缘中的海量数据，限制了 AI 的便利性，所以催生了边缘智能（Edge Intelligence, EI）的概念。边缘智能并非简单地将边缘计算与人工智能结合，其涉及的主题十分广泛[6]。具体来说，EI 利用网络边缘环境中的可用计算资源执行 AI 推理任务，重点研究如何在有限资源的边缘环境中部署和执行 AI 推理模型，以及如何实时、高效、可靠地分析和提取海量数据特征并做出相应决策。此外，研究者还广泛关

注推理模型的处理，例如模型划分、模型压缩、早期退出等关键技术[7-8]。目前，边缘智能在云边协同架构下取得了更好、更高效的发展，但在 DNN 模型的加速推理计算方面仍存在着巨大的挑战和机遇。

1.1.2 研究意义

边缘计算作为一种新兴的计算模式，经过多年的技术发展和标准规范的制定，已经可以有效解决网络环境中的诸多计算问题[9]。其核心思想是将计算和数据处理推向离用户更近的边缘设备，使智能设备能够在本地进行数据处理和决策，以提高响应速度、减少网络延迟和降低数据传输成本。然而，面对越来越复杂的推理计算模型，其消耗的网络资源巨大。随着深度神经网络研究的深入和模型优化的持续进行，一些深度神经网络已经超过了 300 层甚至更多，这些模型仅依靠单一终端设备无法部署。若将这些推理任务交由边缘进行计算可能会导致模型推理精度下降、时延增加等问题[10]。因此，本文旨在研究基于 SDN 的云边协同推理架构，以有效缓解网络计算资源和存储压力，通过改进的智能计算方法结合模型划分与早期退出技术来更好地提高模型的推理效率，这就是本文研究的主要意义。

1.2 国内外研究现状

1.2.1 云边协同计算

云边协同是基于云计算和边缘计算的一种协同工作方式，旨在实现数据的快速处理和实时响应[11]。通过云边协同，可以灵活地分配和调度计算、存储和网络资源，提升系统性能和效率。其核心思想是将数据处理任务分配到最合适的位置，既可以在云端进行大规模的数据处理和分析，也可以在边缘设备上进行实时的数据处理和响应。这样可以减少数据传输延迟，提高系统响应速度，并节省带宽和能源消耗[12]。

基于深度神经网络的云边协同不同于传统模式的协同，其除了在数据、服务等方面协同外还有算法的协同。算法协同指的是多个算法或模型之间的合作和协同工作，以达到更好的结果或解决特定问题。在算法协同中，不同的算法可以相互交互、互相影响，共同完成任务，例如基于云边协同的分布式推理算法，通过相应的模型划分策略将边缘环境中的部分推理任务卸载到云端执行，从而解决边缘设备计算资源紧张的问题[9]。在机器学习中，算法协同可以通过集成学习来提高模型的准确性和鲁棒性。集成学习通过将多个基础模型的预测结果进行组合，

得到更准确的预测结果，也有效地提高了系统稳定性和资源的整体使用效率[13]，此外，研究者还提出基于云边协同的推理调度策略，通过对推理模型进行有效部署，充分利用网络计算资源，以实现最佳的推理时延效果。

尽管云边协同架构弥补了边缘计算与云计算各自的不足，为加速 DNN 模型推理提供了新的思路，但也带来了一些新的问题，例如云、边之间网络带宽的稳定性、任务协同计算卸载的决策、数据传输的安全性等方面的问题[14]。

1.2.2 DNN 推理加速

当前对于 DNN 推理加速的研究主要集中在模型优化、算法优化以及推理框架优化等几个方面[15]，以满足不同应用场景对推理速度和效率的需求，进而推动深度学习技术在各种领域的应用与发展。随着 DNN 模型层数的日益加深，模型推理过程中产生的时延以及计算成本也随之增加。因此，当前的研究重点在于如何满足模型推理需要的计算资源与实时性要求，以达到快速推理的目的[16]。

针对边缘服务器计算、存储资源不足的问题，现有两种解决方案：一是通过多个边缘节点协作学习，提高算力资源，实现实时边缘智能；二是通过云和边协同服务，在云端进行训练过程，然后将训练好的模型部署到边缘侧，既满足训练所需的计算存储资源，又满足推理需要的低时延和实时性要求。然而，仅采用云边协同推理的方式来处理深度学习任务仍存在一些问题。例如，集中式云训练深度学习模型时，需要将原始样本集中到云端，但是边缘产生的实时数据流对云服务器的网络带宽构成了挑战[17]。这种方式不能保证模型参数更新的实时性，也无法及时应对边缘环境的变化。此外，边缘数据具有独特的特征，使得集中云训练的模型难以处理个性化的边缘问题，同时将所有终端数据传输到云中心也会影响数据安全性[18]。针对上述问题，学术界提出了关键技术，包括机器学习模型部署到边缘节点、模型计算迁移、边缘设备模型优化以及协同调度等[19-21]。例如通过使用轻量级压缩技术，使 DNN 模型具有较低的复杂度，从而适用于边缘设备。

深度神经网络模型应用的关键性指标在于通用性、低功耗性及任务的处理效率，虽然依托于深度学习得到了快速发展和应用，但仍面临着一些挑战。现有协同推理算法无法在动态环境中快速调整模型划分策略，模型划分大多考虑云边协同和边端协同两种协同推理方式，而在实际应用环境中，边端环境通常不只有一个服务器。因此，需考虑边边协同和云边端三方协同，以充分利用计算资源。其次，现有的云边协同推理架构通常仅从单任务角度考虑推理任务，然而，在实际应用环境中，系统所要面向服务的用户不一定是唯一的，计算资源也是有限的，

因此需要需考虑资源分配和任务调度问题[22-23]。

1.3 研究内容

本文在深度学习与边缘计算领域进行了广泛调研，并借鉴了现有思想、算法、模型以及框架。同时，参考了其他学者对协同推理计算问题的定义和解决方案。在此基础上，深入研究了现有的协同推理问题，使用模型划分与早期退出技术进行分析研究，建立本文的协同推理模型。研究内容主要包括以下两个方面：

1)针对边缘环境计算资源紧张和单一计算节点无法完成模型部署而导致的推理高时延问题，本文在 SDN 架构中提出了基于模型划分的协同推理算法。首先，通过利用 SDN 技术实时掌握网络的全局信息并在云边、边边中实现传递，使任务的协同推理更有效率。然后在边缘环境中设计一个任务预测器用于判断任务的复杂程度，决策任务的执行环境，实现网络计算资源的动态平衡。模型采用纵向划分的方式，以推理时延和计算成本作为优化目标，利用 DQN 算法进行自适应划分。算法以模型各层输出数据大小、网络传输速率、边缘计算资源等参数信息为输入，并根据这些输入信息执行任务划分与卸载动作来最大化其累积奖励，解决边缘环境下的模型分配问题。最终能够根据不同的推理模型、网络计算资源分布情况以及网络状态得到最优的模型分配策略。

2）为缩减推理模型的层次、进一步加快 DNN 模型的推理速度，本文将推理模型与早期退出技术相结合，提出了一种基于自适应早期退出的推理加速算法。通过 BranchNet 框架训练推理模型的早期退出分支，并利用 NoisyDQN 算法结合 SDN 全局视图实现早期退出分支的预测。算法以网络环境参数、推理需求等信息为输入，以推理时延构建奖励函数，通过执行早期退出分支的不同决策来最大化其累积奖励，解决早期退出分支模型的决策问题。最终可以根据实时的网络环境信息与推理需求预测最优的早期退出分支，优化算法的决策时延性能，减少模型推理的层次，从而提高 DNN 模型的推理效率。具体的研究框架如按下图所示：

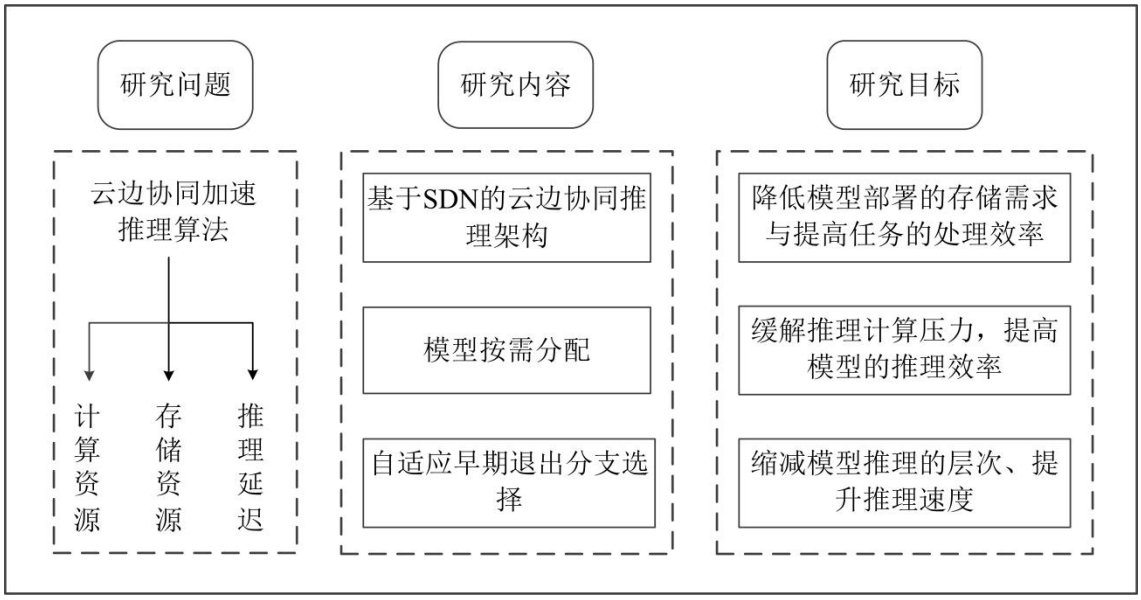

图 1 研究框架

Fig.1 Research framework

1.4 论文结构

本文的结构概述如下：

第一章：绪论。本章主要包括本文的研究背景与意义、国内外的研究现状以及现阶段存在的问题和难点，然后从这些研究方向现存在的发展瓶颈为切入点，简要阐述了本文的主要研究内容并表明研究意义，最后给出本论文的整体研究框架。

第二章：相关理论与技术。该章分别阐述深度神经网络、软件定义网络以及深度强化学习的基本概念、特点以及相应的应用背景，SDN 网络架构优势和模型划分、早期退出等 DNN 推理加速的方法。

第三章：基于模型划分的云边协同推理算法。本章阐述了解决的问题与方法，即在动态的网络状态与任务需求环境下，如何划分推理模型来缓解边缘环境的计算压力，实现推理低时延的目标。首先，提出了一种基于 SDN 的云边协同推理架构，该架构利用 SDN 全局视图全面掌握网络计算资源，相比传统的协同计算架构，具有更强的自适应性。然后详细描述了架构的各个组件和工作流程，并通过控制器监控全局信息实现对网络的全局管理。最后介绍了模型推理加速的方法，即利用改进后的 DQN 算法来处理边缘环境中的模型分配问题，该算法利用 SDN 获取环境信息并利用反馈信息创建适当的奖励，以评估时延和计算划分策略的效果，

并通过对比实验验证了算法的有效性。

第四章：基于自适应早期退出的推理加速算法。本章详细介绍了早期退出的推理框架、模型退出预测器等组件的功能，随后描述了模型基于 BranchyNet 框架训练分支的过程，并阐述了基于自适应早期退出的分支决策算法。即通过全局视图获取实时网络状态信息训练算法模型，利用噪声函数通过空间扰动的方式来推动探索，函数策略的诱导随机性可用于帮助高效探索，从而缩短决策时延并提高模型分支选择的准确性，进一步提升模型推理的时延性能。

第五章：总结与展望。本章首先对本文提出的基于深度强化学习的 DNN 协同推理加速方法进行总结，针对协同推理架构与算法的核心思想进行简要介绍，同时对协同推理的研究方向做一个展望。

2 相关理论与技术

2.1 软件定义网络概述

2.1.1 软件定义网络的概念与特点

软件定义网络是一种网络架构，其核心思想是将网络控制平面与数据转发平面分离，通过集中式的控制器对网络进行统一管理和控制。相较于其他网络架构而言，SDN 具有集中式控制与开放性编程的特点，能够通过软件编程的方式定义和控制网络，使管理者可以通过软件程序来控制网络中的数据流[24]。SDN 通过集中控制能力实时获取网络状态信息，根据策略对网络流量进行调整，并支持快速创新和灵活适应各种应用场景，如云边协同计算。其开放接口能力使得通过简单的软件程序操作实现网络新功能和快速部署成为可能[25]。总的来说，SDN 通过创新的架构和技术，实现了对网络的集中式控制、自动化配置和灵活性管理，从而为网络提供了更高的效率、可管理性和创新性[26]。

SDN 与传统分布式网络结构不同，其将数据平面与控制平面分离，包括应用平面、控制平面和数据平面。应用平面提供各种网络程序接口，控制平面有逻辑集中的控制器负责网络集中管控，数据平面包含网络设备用于信息收集和转发[27]。SDN 的关键在于集中式的控制器通过南向接口和北向接口连接数据平面，控制器发送控制指令配置网络设备的数据转发规则，实现对网络的集中管控[28-29]。SDN 架构可包含单个或多个控制器共同协作运营网络[30]。总的来说，SDN 架构具有以下优势：

1）灵活性和可编程性：SDN 允许网络管理员根据需要动态配置网络，实现灵活的网络管理和服务提供。通过集中式控制器，SDN 可以实现高级别的网络自动化，减少手动配置和管理的需求，提高网络效率，使得 SDN 更容易实现网络创新，快速部署新的网络服务和应用。

2）集中控制：控制器具有网络的全局视图，可以获取网络的全局状态信息，通过配置控制器可以实现对流表的集中管理。

3）降低成本：通过集中管理和自动化，SDN 可以降低网络管理的成本，提高资源利用率，从而降低整体成本。

2.1.2 软件定义网络的应用背景

软件定义网络作为一种新兴的网络架构，其应用背景涵盖了广泛的领域，从

企业网络到数据中心再到运营商网络，都能够获得显著的好处。SDN 的核心理念是将网络控制平面从数据转发平面中分离出来，以实现网络的集中管理和灵活控制。这种分离的架构使得网络更具可编程性和自动化程度，从而带来了诸多应用场景的革新。

首先，SDN 在运营商网络中发挥着重要作用[30]。运营商网络需要支持大规模的用户连接和不断增长的流量需求，传统的网络架构往往面临着管理和扩展的挑战。SDN 的集中管理和自动化控制可以帮助运营商更好地管理和优化网络资源，实现灵活的服务部署和流量调度，提高网络的可靠性和性能，同时降低运营成本。

其次，SDN 在企业网络中的应用备受关注[31]。传统的企业网络往往采用分布式的网络设备和控制系统，导致管理和配置复杂繁琐。而通过应用 SDN，企业可以通过集中式的控制器来管理整个网络，实现统一的策略管理和流量控制。这使得网络更具灵活性，能够更快地适应不断变化的业务需求，并且能够提供更好的安全性和可靠性。

此外，SDN 还在数据中心网络中的应用也十分广泛[32]。数据中心网络需要处理大量的流量和复杂的应用程序交互，传统的网络架构往往无法满足其灵活性和可扩展性的需求。而 SDN 的可编程性和灵活控制使得数据中心网络能够更好地适应不同应用的需求，实现灵活的流量管理和资源分配，从而提高整体的性能和效率。

总的来说，软件定义网络在运营商网络、企业网络和数据中心网络等领域的应用都能够带来显著的好处，提高网络的灵活性、可编程性和自动化程度，从而更好地满足不断变化的业务需求，推动网络的创新和发展。

2.2 深度神经网络概述

2.2.1 深度神经网络的概念与特点

在 AI 迅速发展的时代，机器学习算法（Machine Learning, ML）是其重要分支之一。传统的 ML 是对真实应用场景中获取的网络数据进行训练、分类和预测，如线性回归、贝叶斯网络以及元神经网络等学习预测方法。然而，智能应用越来越多地采用 DNN 算法，这种现代机器学习方法利用深度神经网络学习数据的深层表示和特征，以获取更有效的信息和更准确的预测结果[33]。

深度神经网络技术利用多层人工神经网络学习数据的深层表示和特征，在计算机视觉与人脸识别等领域有着出色的表现。这种模型由多层神经网络组成，每

层包含多个神经元，能够产生线性或非线性输出，如图 2 所示是一个简单的若干层神经网络模型与神经元结构，神经网络模型的每一层包含若干个神经元。与典型的神经网络相比，深度神经网络模型通过更深层次的中间隐藏层对数据进行大量训练，从而学习到输入数据的特征和表示，实现高精度的推理[34]。但是在 DNN 模型推理计算过程中会产生大量神经元，导致 DNN 模型的计算开销巨大[35]。

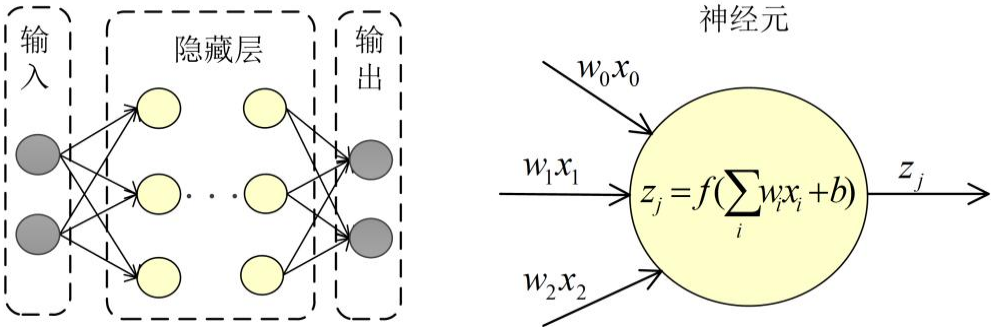

图 2 深度神经网络模型

Fig.2 Deep neural network model

2.2.2 深度神经网络的应用背景

深度神经网络是一种在人工智能领域引起广泛关注的技术，其应用背景涵盖了多个领域，从图像识别到自然语言处理都能够获得显著的应用效果[36]。

在图像识别领域的应用中，深度神经网络备受关注。随着数字图像的广泛应用，人们需要自动化地识别图像中的对象、场景等信息。传统的图像识别方法往往需要手工设计特征提取器，而深度神经网络能够通过大量的图像数据自动学习到有效的特征表示，从而实现更准确、更高效的图像识别任务，如人脸识别、物体检测等。此外，深度神经网络在自然语言处理领域也发挥着重要作用。自然语言处理涉及到文本分析、语义理解、机器翻译等任务，传统的基于规则或统计方法的模型往往面临着语义理解不足、泛化能力弱等问题。而深度神经网络通过学习大规模的文本数据，能够捕捉到语言中的复杂结构和语义信息，从而实现更准确、更自然的自然语言处理任务，如情感分析、文本生成等。

综上所述，深度神经网络在各个领域的应用非常广阔，推动了人工智能技术的发展和应用。随着深度神经网络模型的不断优化和算法的进步，相信其在未来会在更多领域展现出强大的应用潜力。

2.3 深度强化学习概述

2.3.1 深度强化学习的概念与特点

深度强化学习是指在强化学习中使用深度学习算法来学习策略或价值函数的一类方法。具体而言，深度强化学习是基于强化学习的思想，即通过试错和奖励来优化策略或价值函数，从而使智能体能够在环境中获得最大的累积奖励。同时，使用深度神经网络来表示策略或价值函数来处理高维、非线性和连续的状态和动作空间，具有很强的泛化能力和自适应性[7-8]。深度强化学习不需要事先了解环境的动力学模型，可以直接从环境中采样并通过试错来学习策略或价值函数，其只需要在探索和利用之间平衡，既要尝试新的动作来发现更好的策略，又要利用已知的策略来最大化累积奖励。

2.3.2 深度强化学习的应用背景

深度强化学习是结合了深度学习和强化学习的一种方法。它可以应用于各种领域，例如游戏、机器人控制、自然语言处理等等[9]。具体来说，深度强化学习能够通过多次试错来学习到一个复杂任务的最优策略，这个过程中需要不断地与环境进行交互并根据反馈来调整策略。

深度强化学习在人工智能领域的应用也越来越广泛，例如机器人控制领域；深度强化学习可以应用于机器人控制中，帮助机器人完成各种复杂任务，即通过训练深度神经网络，机器人可以学会在不同的环境下走路、跑步、跳跃等等。此外，深度强化学习也可以应用于自然语言处理领域中，例如对话系统和智能客服，即通过训练深度神经网络，可以让机器人根据用户的问题和反馈来进行自主学习和调整。

2.4 模型推理技术

DNN 推理技术的核心思想在于模型的训练和推理，其推理阶段发生在模型训练完成之后。例如，在执行目标检测任务时，需要收集大量图片数据作为模型的训练样本，此过程通常包括前向传播和反向传播两个步骤。在前向传播阶段，输入数据通过多层神经网络，经过一系列的加权和激活函数操作，经过逐层计算后得到最终输出。而在反向传播阶段，通过计算损失函数得到输出结果与真实标签之间的差异，并反向传播误差，更新模型参数，再经过多批次训练来优化 DNN 模型的推理性能[37]。本文主要针对 DNN 模型推理计算的效率进行优化研究，现

阶段关于 DNN 推理加速有模型划分、早期退出、模型压缩以及模型选择等技术，本文通过在基于 SDN 的云边协同推理架构中利用模型划分、早期退出技术来提升 DNN 模型的推理速度。

2.4.1 模型划分

模型划分是将整个 DNN 推理模型分解成多个部分，然后在不同计算设备上执行这些部分的推理计算，是一种基于两个及以上计算节点进行协同推理计算的常用方法[38]。这种划分有助于在边缘设备和中心云之间实现计算负载均衡，从而优化整体推理性能。由此可知，模型划分技术是根据网络资源情况，选择最佳的模型分区节点，使其在面对有限的计算资源环境时，可以实现推理所需的计算代价与参数传输时延的均衡[39]。

当前，模型划分在边缘智能和云边协同计算领域备受关注。学者们致力于开发算法和框架，以实现对深度学习模型的有效划分和部署，从而在边缘设备和中心云之间实现高效的推理计算。通过研究各种模型划分方法，分析 DNN 模型内部层结构和外部推理架构，发现基于云边协同的模型划分推理计算具有较大的性能提升空间，并且能够在保证推理精度的前提下更好地适应边缘计算[40]。

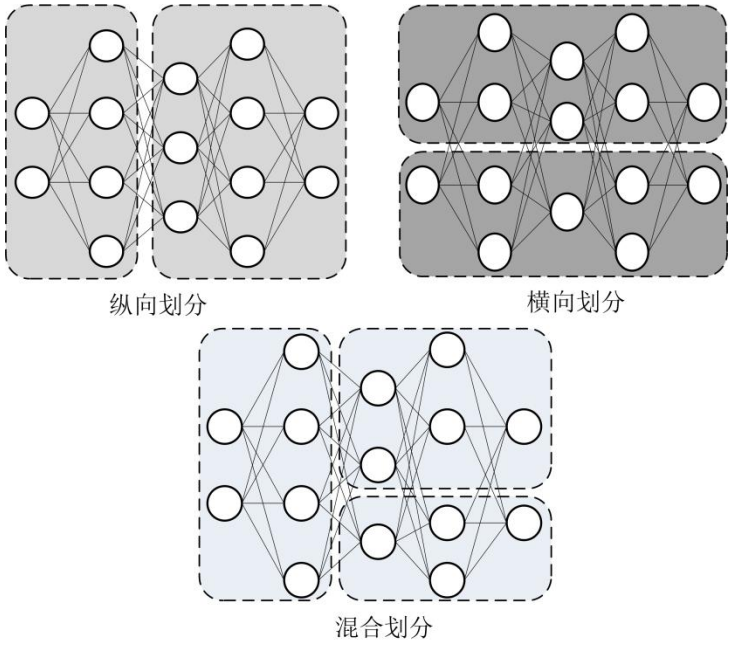

图 3 模型划分类型

Fig.3 Type of model partition

深度神经网络模型具有良好的内部结构，可以通过多种切分方式将整个推理

模型分成不同粒度且具有相互依赖关系的子模型，如图 3 所示。这些子模型可以根据其相应的依赖关系分别部署在云或边缘环境中，如采用纵切的 DeepThings、横切的 Neurosurgeon 以及混合方式的 DeepX 等方法。这些方法通过优化计算和通信资源的代价函数对模型内部的划分点进行枚举，以求找到满足用户或系统需求的最佳划分方案[23]。

综上所述，深度神经网络模型划分为多个部分并在不同设备上执行时，需要考虑模型结构、计算资源、通信开销等多方面因素，故而如何设计有效的划分策略是一个具有挑战性的问题。此外，模型划分会引入不同设备之间的数据传输和通信开销，如何在保证推理性能的同时最小化时延开销也是一个需要解决的问题。

2.4.2 早期退出

模型的早期退出是指在进行模型推理计算时，在一定的条件下提前停止计算过程，从而加速推理计算并减少资源消耗。这种技术通常用于深度神经网络等模型的推理计算过程中，当输入数据已经得到足够的置信度预测结果时，即可提前停止计算，而不必等待整个过程完成。这种方法通过增加多个推理退出分支来提升 DNN 模型的推理速度[41]。

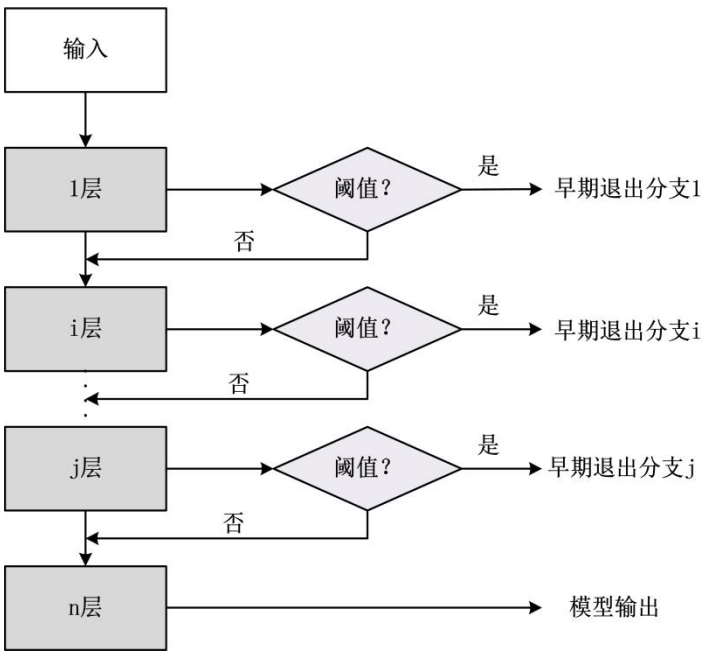

图 4 早期退出模型

Fig.4 Early exit model

早期退出是通过分支选择器计算分支可信度来确定推理退出分支，通常通过

计算 Softmax 输出层的熵大小获得，结构流程如图 4 所示，也可增加可信度决策模型来确定退出分支。在连续多推理任务请求场景中，可以考虑满足时延要求的退出分支，并将推理时延建模为收益函数，按请求次序进行推理，从而获得最优推理结果。例如研究学者提出的 DDNN 模型，该模型以模型早期退出分支为切割点，并将其分别部署在边缘端与云端，通过采用云边协同的方式来减少网络计算资源的消耗、加速模型的推理[42]。

目前，早期退出的研究表明，这一技术在边缘智能的模型推理计算中具有广泛的应用前景。研究人员致力于开发早期退出的算法和框架，以在保证预测准确度的前提下提高模型的推理速度，从而满足实时性要求。但是早期退出技术需要在保证预测结果准确性的前提下实现计算的提前终止，因此如何确保提前退出时仍能保持结果的准确性和鲁棒性是一个重要挑战[43]。此外，由于输入数据和模型状态可能不断变化，如何根据实时网络环境状态来决策早期退出策略，以适应不同场景下的推理需求，也是一个关键难点[44]。

2.5 本章小结

本章主要介绍了研究方向所涉及到的相关技术：软件定义网络与深度神经网络模型。首先，阐述了软件定义网络的概念和架构特点，强调了软件定义网络在云边协同领域应用的优势以及软件定义网络的应用背景。其次，叙述了深度神经网络与深度强化学习的概念以及应用背景。最后介绍了加速 DNN 模型推理的方法，阐述了模型划分与早期退出技术的基本概念以及现存的研究难点。随着协同推理技术的不断发展与创新，我们可以更好的应对边缘计算资源紧张、内存不足和推理高时延的问题，从而实现协同推理在边缘智能领域的广泛应用与发展。

3 基于模型划分的云边协同推理算法

3.1 引言

深度学习，尤其是深度神经网络作为当代人工智能的骨干技术，在计算机视觉、语音识别、自然语言处理等领域广泛应用。因其是计算密集型网络且在依靠单一的云端或边缘节点执行推理任务时会产生计算资源紧张、推理时延难以预测、推理性能明显下降等问题。为解决这些问题，出现了许多云边协同处理方案，如基于云边协同的分布式推理算法[45-47]，通过将部分推理任务卸载到边缘节点以缓解云端的负载压力，从而达到推理低时延、高可靠性的目的。研究者还提出基于云边协同的推理调度策略，采用启发式贪婪算法对模型进行有效部署，以实现最佳的推理时延效果[16]。

然而，在实际应用场景中，面对计算资源有限的网络环境，如何在网络状态和任务需求动态变化的情况下协同多个边缘节点合理分配推理任务，实现边缘集群与云端协作以优化推理时延，是一个值得深入研究的问题[48]。再者现有的研究未考虑到异构的边缘设备对推理任务的影响[9]。

因此，本章结合 SDN 技术提出了基于模型划分的云边协同推理算法，通过纵向切分推理模型的方式，协同多个计算节点解决边缘网络在执行推理任务时计算资源紧张的问题，以此来优化模型推理的时延性能。首先，利用云、边协同方法处理推理计算问题，针对边缘环境中单一节点算力不足的问题，采用协同多个计算节点的方法参与 DNN 推理，并结合深度强化学习中的 DQN 算法实现推理模型的自适应分配。其次，设计任务预测器，根据任务自身的复杂程度来决策任务的执行环境，以提高模型推理的效率。最后，算法结合 SDN 技术从全局感知推理任务与网络资源信息，通过协同架构的南向接口屏蔽边端设备之间的差异，使用全局视图获取网络资源信息，并通过边端控制器的北向接口按需配置并调用资源，为推理任务提供差异化的资源和服务保障，从而达到提高资源利用效率和优化推理时延的目标。实验结果表明，在动态的网络环境中，SDN-CEP 算法能显著降低 DNN 的推理时延。

3.2 问题描述

目前，已有大量的研究来加速 DNN 模型推理。主要从模型压缩、模型划分、早期退出以及模型选择等方面降低网络环境中的推理时延和成本[49-50]。例如将云

计算环境中的推理计算问题建模为最短路径和整数线性规划问题，并结合模型压缩方法以减少通信成本[51]。针对模型划分方法，以 DNN 模型每层的神经元为单位，采用 DRL 自适应卸载策略在各个物联网设备上完成运算，实现在资源受限的物联网设备上低时延的推理目标[52]。模型选择则是通过训练一个轻量级的二进制神经网络分支设计复合协同的 DNN 模型来减少系统推理开支[53]。以上方法虽然有效降低模型的推理时延与计算资源需求，但会对模型的推理准确度造成损失，而且现有协同推理模型的最佳划分点是在离线阶段下通过网络带宽、计算资源及模型层粒度大小等因素相对均衡得出的。因此，如何在网络状态和任务需求的动态变化中寻找最佳划分点是我们研究的目标。

随着深度强化学习被广泛应用于感知决策领域，通过与网络环境的不断交互来实现优化目标的最优策略[54-55]；软件定义网络则可以通过控制器获取实时网络状态信息，从而更加灵活地控制网络、提高网络性能[56]。所以本章结合 SDN 技术提出了基于模型划分的云边协同推理算法，并通过 DQN 算法对 DNN 模型进行自适应的分配。通过实验验证对于不同的推理任务，该算法可以有效提高模型的推理效率。

3.3 系统推理架构

3.3.1 基于 SDN 的云边协同推理架构

SDN 是一种新型网络架构，相较于传统网络架构，SDN 具有更高的灵活性和可编程性。SDN 可以通过编程控制网络且拥有获取实时网络状态信息的能力[57-58]，从而优化云边协同网络配置、改善网络性能，使得运用 SDN 技术加速云边协同推理更加可行。架构通过将 SDN 与云边协同推理计算相结合，利用 SDN 技术来获取整个网络的状态信息，使不同边缘服务器之间的任务处理更高效[59]。如图 5 所示，将基于 SDN 的控制器融入边缘层中，并通过控制器的南北向与东西向接口使其连接成一张可以快速信息交换的网络，实现云边环境的协调控制，从全局感知网络的资源信息并进行合理分配。因此，结合 SDN 技术创建的协同网络架构能够更好的发挥云与边的优势，进而满足协同推理任务中在资源调度和信息集中控制的需求。

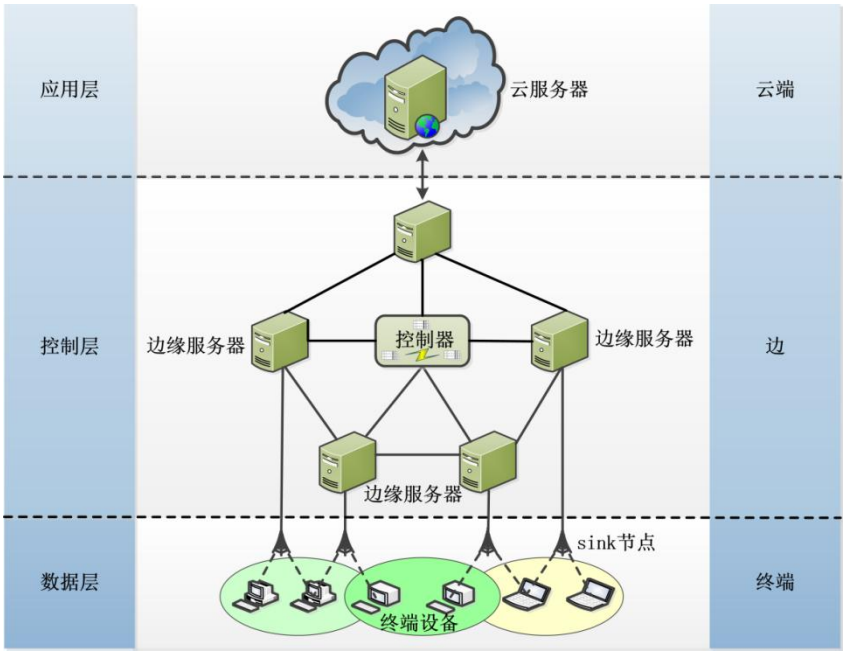

图 5 基于 SDN 的云边协同架构

Fig.5 Cloud edge collaboration architecture based on SDN

SDN 控制器位于云边协同架构的边缘层，可以通过共享全局视图来了解整个云边网络计算资源的分布情况[60-61]。全局视图消息包含控制器、链路和边缘服务器等信息，并使用统一的 XML 文件格式进行传递，使得消息格式具有灵活性和易扩展性[62]。

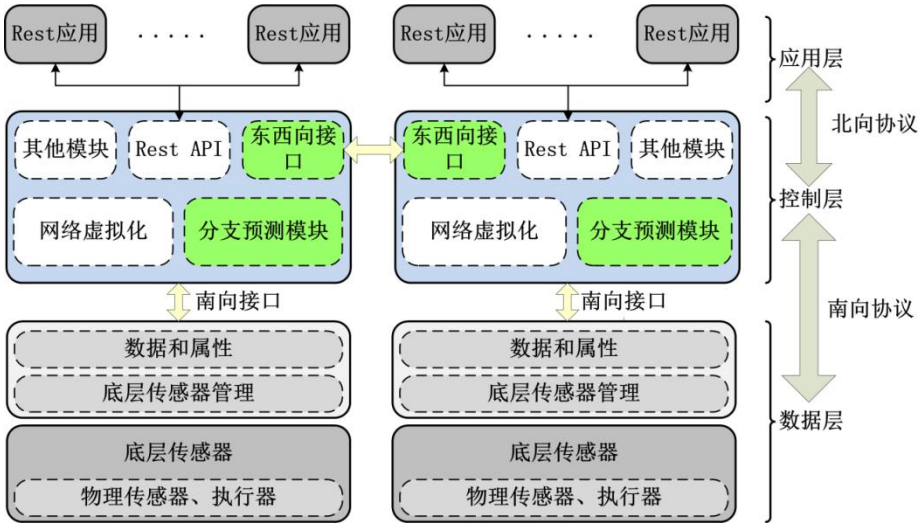

图 6 SDN 控制器

Fig.6 SDN controller

全局网络视图的构建由各控制器之间不断交互视图信息来完成，如图 6 所示，所有网络视图信息都是由北向的 Rest API 接口提供给网络应用程序[63]。本文通过添加分支预测等模块来实现 SDN 架构下网络环境信息的获取以及模型的加速推理计算。SDN 控制器可以相互获知彼此所在的地址，控制器之间通过定期驱动的方式来交换和共享网络的全局状态信息，在基于 SDN 的云边协同架构中，SDN 控制器可以获取整个系统网络的资源信息。

3.3.2 系统模型框架

基于 SDN 的云边协同推理系统框架如图 7 所示，该系统框架主要有三部分组成：

1）任务预测阶段：在边缘环境的主服务器中建立一个预测器来判断任务数据的复杂程度。

2）推理选择与划分阶段：经过任务预测器的判断机制决策任务执行环境。对于部署在边缘环境中的模型，根据全局视图获取的实时网络状态、节点计算资源、节点数量以及模型层粒度大小等因素结合 DRL 中的 DQN 算法求取模型分配策略。

3）任务卸载阶段：根据任务复杂度的预测结果与边缘环境中的模型分配策略来执行推理任务，相关的参数信息数据通过 SDN 全局视图进行传递和控制。

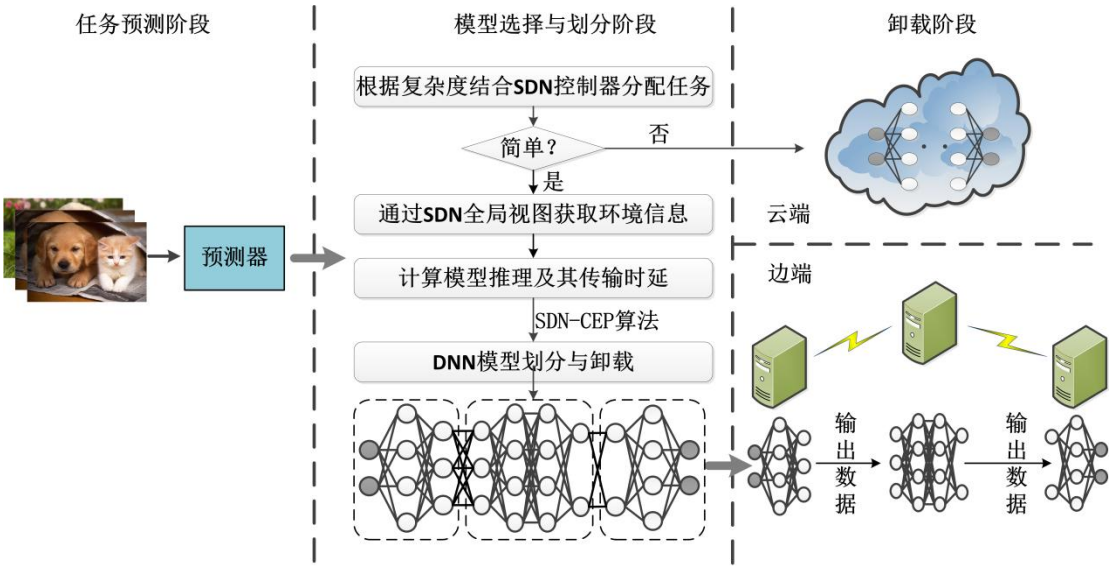

图 7 系统模型框架

Fig.7 System model framework

3.3.3 任务复杂度预测器

现有的协同推理研究过多关注推理模型和网络环境，而忽视对模型输入数据的处理[64]。本章构建任务预测器，通过对输入的数据提取特征进行复杂度判断，使其能够选择合适的推理环境来降低推理时延。DNN 模型的推理任务是在动态的环境条件下收集的，并不总是以可预测的方式呈现，这增加了任务自身的复杂性，且这种复杂性与模型的推理能力息息相关[65-66]。考虑到这一点，如果允许边缘节点进行所有推理，则模型的推理准确性与推理时延可能无法令人满意。因此为了实现更高的精度和速度，采用云-边协同架构处理模型的推理任务，该架构的主要思想是：在云端部署深度神经网络模型，利用其丰富的计算资源实现高性能的推理；边缘环境中则将与云端相同的 DNN 模型进行划分，通过边缘节点间协同的方式进行推理，解决单一计算节点资源紧张的问题。复杂度预测器利用函数 q 来判断任务的复杂程度，以决策任务的执行环境，预测流程如下图 8 所示。

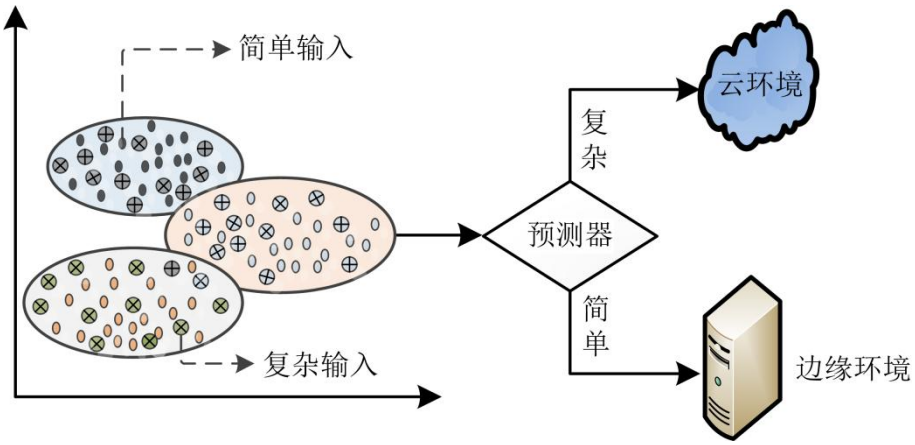

图 8 任务复杂度预测流程

Fig.8 Task complexity prediction process

任务的复杂程度受多方面因素的影响，但主要取决于其自身的信息元素，且在使用深度神经网络执行推理任务时，都需要将任务表示为高维度的向量数值矩阵，以便于后续的推理计算，所以我们对推理任务进行复杂度判断时，通过特征提取将其用 $M\times N$ 的数值矩阵进行表示。一般而言，任务的数值化表示矩阵中，数值变化得越多、越频繁，其复杂度则越高。因此，可以利用相关的数值变化来反映任务的复杂程度。故而判断任务复杂度的函数 q 可以定义如下：假设任务的矩阵表示大小为 $M\times N$，则矩阵相邻元素的个数为：$M\times(N\text{-}1)+N\times(M\text{-}1)=2MN\text{-}M\text{-}N$。复杂度预测函数 q 表示矩阵中发生变化的相邻元素个数与矩阵所有相邻元素的个

数之比，所以预测函数表达式如下所示：

$$q = \frac{K}{2MN - M - N} \tag{1}$$

其中 K 表示矩阵中发生变化的相邻元素的个数。

具体而言，采用基于阈值的方法来选择任务的执行环境，即如果 q 函数的预测值大于所设定的某个阈值δ，则将任务卸载到云端中处理，反之则在边缘环境中进行处理，通过将边端模型分配到边缘计算节点上进行协同计算，并结合 SDN 技术掌控全局的网络信息，从而保证推理精度和速度。

3.4 DNN 模型的划分与卸载

3.4.1 模型划分决策

深度神经网络在执行推理任务过程中的主要计算量在于全连接层和卷积层，各层的计算时延与其浮点运算次数（floating point operations, FLOPs）有关，因此通过 FLOPs 可以预估推理模型各层的计算时延，为 DNN 模型划分和卸载提供决策依据[67]。全连接层的 FLOPs 计算公式为：

$$FLOPs = (2I - 1) \times O \tag{2}$$

其中全连接层的输入与输出维数用 I、O 表示。卷积层的 FLOPs 计算公式为：

$$FLOPs = (2C_{in}K^2 - 1)HWC_{out} \tag{3}$$

其中 H 和 W 分别表示输入图片的高和宽，C_{in} 和 C_{out} 表示卷积计算的输入和输出，K 表示卷积核的大小。本文通过执行 AlexNet 深度神经网络来展示推理模型的计算时延与通信特征。如图 9 所示，从 AlexNet 模型的输出数据以及相应的计算时延可以看出模型每层输出的数据大小与其计算时间的相关性难以分析，因此我们考虑层计算量、粒度大小和节点资源等因素，通过结合 DQN 算法和 SDN 全局视图，以纵向划分的方式对部署在边缘的推理模型进行划分和卸载，并与部署在云端的模型并行推理，从而实现云边协同的双赢局面。

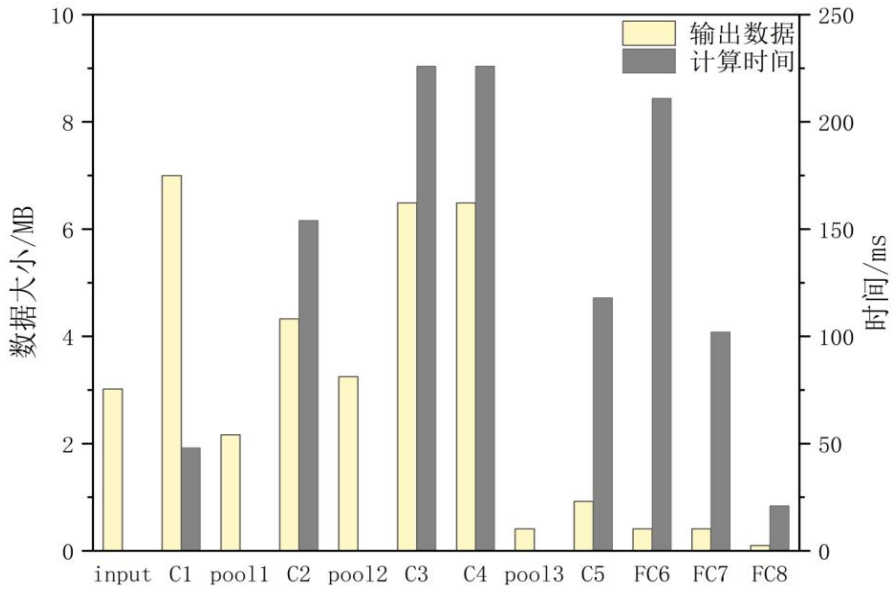

图 9 模型各层输出数据大小及计算时间

Fig.9 The output data size and calculation time of each layer of the model

3.4.2 基于 DQN 的模型分配

使用深度强化学习寻找问题的最优解，需将问题建模为马尔科夫决策过程（Markov Decision Process, MDP）[67-69]，即问题模型用一个四元组 (S,A,P,R) 来表示。优化目标则是通过最优策略使智能体长期累积奖励且 DRL 采用深度神经网络来近似价值函数或策略。

1）状态：$S=\{B,ES,D\}$，其中 B 表示当前网络的传输速率；$ES=(ES_1,ES_2,...,ES_n)$ 表示边缘环境中的 n 个边缘节点；$D=(D_1,D_2,...,D_m)$ 表示拥有 m 层的推理模型，其各层输出数据大小。

2）动作：当前网络状态的 DNN 模型有 *m-1* 个划分点，边缘环境中有 n 个边缘节点。而动作空间由所有可能的动作组成，对于基于 DQN 算法的模型分配而言，动作就是在边缘服务器间选择 DNN 推理模型的划分层次以及执行该部分模型的计算节点。所以，动作空间 $A=\{a^{1,1},a^{1,2},\ldots,a^{i,j},\ldots,a^{n,m}\}$，$a^{i,j}$ 则表示第 j 层网络模型在边缘计算节点 i 执行推理任务，初始动作由算法随机产生，随后与网络环境不断交互而逐渐稳定。

3）奖励：奖励值越大表示选择的动作越好。但在执行推理任务中，因以优化时延为目标，所以根据时延特征将奖励值定义为：

$$R = -((1-\omega)T_{all} + \omega C_{all}) \tag{4}$$

其中，ω为奖励函数中的权重因子，用来调节延迟与成本的权衡。具体来说，用户可以根据不同的应用需求，通过调整时延和成本的上界进行设置，假设时延和成本的上界分别为 T^u 和 C^u，则权重因子可以表示为 T^u/C^u[70]。

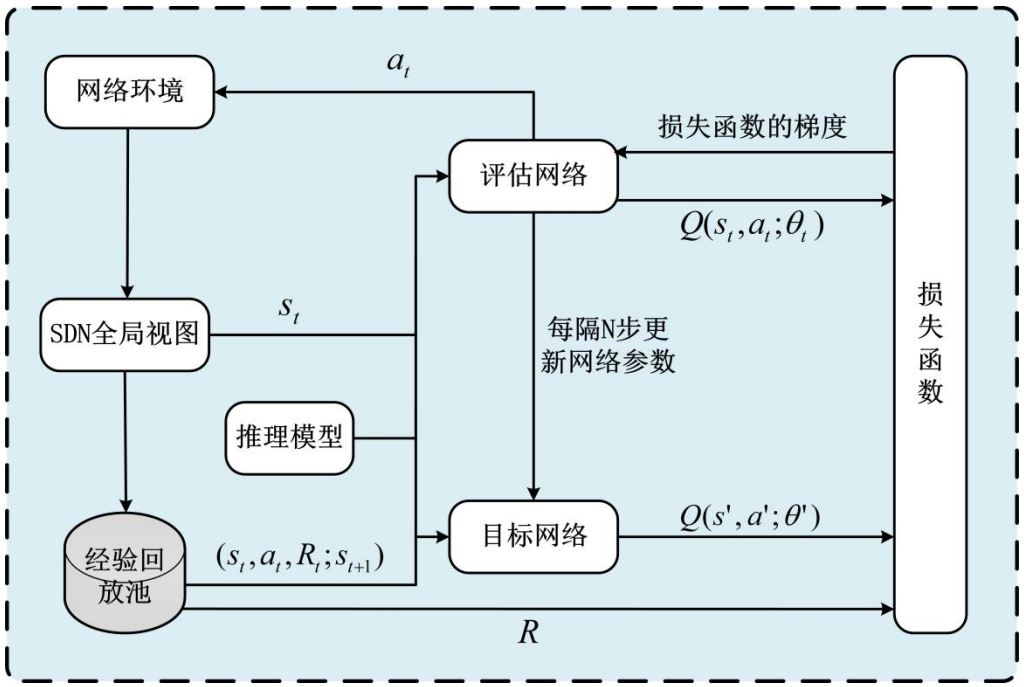

图 10 改进的 DQN 算法模型

Fig.10 Improved DQN algorithm model

DQN 算法是神经网络与强化学习算法的融合，通过利用神经网络将高维数据转化为状态做为强化学习的输入，是一种基于价值而非策略的算法[71]。本章通过 DQN 算法结合 SDN 全局视图来处理边缘环境中推理模型的划分与卸载问题，改进后的 DQN 结构模型如图 10 所示。本文通过动作价值函数 Q 来确定在状态 s 时采用动作 a 后任务推理的总时延与计算代价，所以，目标价值函数具体如公式（5）所示，其中 s' 和 a' 分别表示网络状态 s 时，采用划分动作 a 后的下一个状态与将要采取的动作，θ' 表示目标网络的参数，γ则表示折扣因子，R 为任务推理时延与成本的奖励值。此外，本文还将经验回放池中的奖励信息作为 DQN 算法中网络模型的输入，以此来提高目标网络求取模型划分最优策略的准确性。

$$Q_t = R + \gamma \max_{a'} Q(a', s', \theta') \tag{5}$$

使用强化学习处理任务时，由于数据之间是相关联且非静态的，直接使用会导致神经网络难以收敛，所以 DQN 算法把智能体与网络环境交互的样本数据存储在经验池中，对神经网络进行训练时直接通过经验池随机批量抽取。DQN 算法训练本质是为了使得当前 Q 值无限接近于目标 Q_t 值，但由于目标函数中含有 Q 函数，导致目标函数变化，给训练增加了一定的难度，因此，为更好的训练神经网络，消除样本数据之间的依赖性，本章在 DQN 算法结构中结合 SDN 的全局视图来读取边端网络环境中的信息数据，通过模型输出与目标值之间的均方误差来表示损失函数，通过反向传播使用梯度下降法来更新神经网络的参数θ，从而不断降低神经网络的损失函数并提高神经网络精度。算法定义的损失函数如下：

$$Loss(\theta) = E[(Q_t - Q(s,a,\theta))^2] \tag{6}$$

3.4.3 任务推理时延与成本

假设 DNN 模型有 m 层，边缘节点数量为 n。使用 n 作为限制对 DNN 模型进行不规则划分，所有切片情况为 N={0,1,.....,n-1}。分区点 $i \in N$ 表示将 DNN 模型划分成 i+1 部分并部署于边缘节点间进行协同推理，第 i 层的输出数据大小为 D_i。其中，N=n 和 N=0 是两种特殊情况。N=n 表示 DNN 推理需要协同所有边缘节点进行处理；N=0 表示在边缘节点中选择一个最优节点来执行推理任务。系统整体框架在执行 DNN 推理任务时，同批次任务 T 的推理时延 T_k 由边缘节点执行时间 T_{edge}、云端的执行时延 T_{cloud} 以及数据上传到云边的时延 T_{trans} 三部分组成。由于模型推理输出的结果很小，其传输到终端设备上的时间被忽略。

$$T_{all}^{k} = T_{edge} + T_{cloud} + T_{trans} \tag{7}$$

基于回归模型预测 DNN 每层在边缘节点上的执行时间，即每张图片在边缘推理的时间可以表示为：

$$T_{edge} = \sum_{j=1}^{m} ED_j + \sum_{p=1}^{m-1} \frac{D_p}{B} \tag{8}$$

其中 ED_j 表示边缘节点执行第 j 层所需的时延，D_p 为模型划分点第 p 层所输出的参数大小，B 为当前的网络传输速率。云端的推理时延则为：

$$T_{cloud}=\sum_{j=1}^{m}CD_j \tag{9}$$

假设任务 T 中包含 k 张图片，其中有 i 张在边缘节点中执行，k-i 张在云端中执行，则传输时延 T_{trans} 为：

$$T_{trans}=D_i+D_{k-i}\Big/B \tag{10}$$

其中 D_k 表示在边缘环境中执行数据的大小，$D_{k\text{-}i}$ 为云端执行数据的大小。综上可知，整个任务推理的总时延为：

$$T_{all}^{k}=\sum_{k=1}^{i}T_{edge}+\sum_{k=1}^{k-i}T_{cloud}+D_i+D_{k-i}\Big/B \qquad s.t.\begin{cases}C1:\sum_{i=1}^{k}\text{memory}_i\le M\\ C2:\sum_{i=1}^{k}\text{CPU}_i\le C\\ C3:I_j\in N\text{-}and\text{-}ED_n\in EDC\end{cases} \tag{11}$$

$C1$，$C2$，$C3$ 为目标函数的约束条件，M 和 C 分别表示边缘节点的内存与其 CPU 的计算资源，k 则表示当前执行的任务数量，$C3$ 则表示算法所选的策略在网络所有可选的策略中。

模型推理任务的成本由执行成本 C_{exe} 和传输成本 C_{trans} 组成。所以同批次任务 T 的执行成本可以表示为：

$$C_{exe}=C_{edge}+C_{cloud}=T_{edge}q_i^e+T_{cloud}q^c \tag{12}$$

$$q_i^e\in\{q_1^e,q_2^e,...,q_n^e\} \tag{13}$$

q^e,q^c 表示任务所在的边缘服务器和云服务器的单位时间执行成本，q_n^e 则表示边缘服务器 n 的单位时间执行成本。任务传输所产生的成本可以表示为：

$$C_{trans}=D_k\times b_c \tag{14}$$

其中 D_k 为传输数据的大小，b_c 为单位数据的传输成本。所以系统的总任务成本可以表示为：

$$C_{all}=C_{exe}+C_{trans} \tag{15}$$

3.4.4 算法设计

基于模型划分的云边协同推理算法解决了边缘环境下因计算资源紧张而导致的模型推理高时延问题，实现了云边协同环境中模型高效率的推理。该算法首先利用全局视图获取环境中的状态信息，根据复杂度预测器分配任务的推理环境，然后利用改进的 DQN 算法根据实时的网络环境状态得到最优的模型分配策略。该算法的伪代码如下所示：

算法 1：基于模型划分的云边协同推理算法

输入：

网络传输速率 B、各边缘节点与云端资源、DNN 模型各层输出数据大小，推理任务 T、预测器阈值 δ

输出：

DNN 推理模型的划分和卸载策略

1：初始化实验所需的参数；

2：for episode in T do：

3： for each task do：

4： 将任务转换为数值矩阵 $M \times N$；

5： 根据公式 $q = \frac{K}{2MN - M - N}$ 计算任务复杂度；

6： if complexity_score>= δ

7： 任务传输到云端进行处理并根据公式 $T_{cloud} = \sum_{j=1}^{m} CD_j$ 计算推理时延；

8： else

9： 通过 SDN 全局视图得到环境 s 并传递给智能体 agent；

10： 利用贪婪策略随机执行动作 a^{ij}；

11： 结合部署的推理模型利用公式 $T_{edge} = \sum_{j=1}^{m} ED_j + \sum_{p=1}^{m-1} \frac{D_p}{B}$ 计算时延；

12： 根据公式 $R = -((1-\omega)T_{all} + \omega C_{all})$ 计算奖励值；

13： 计算 $Q_t = \begin{cases} R_t, s' \text{为最终状态} \\ R_t + \gamma \times Q(a', s', \theta'), \text{否则} \end{cases}$

14： 执行动作后，利用全局视图收集状态 s'；

15： 将经验 $(s, a, R; s')$ 存储到经验回放池中；

16： 定期更新目标网络参数 $\theta' = \theta$；

17： 更新 DQN 算法的最优策略；

18： end for

19： 返回步骤 2；

20：end for

3.5 实验结果与分析

3.5.1 实验环境

本文在 64 位操作系统的 Windows 10 环境下进行实验，使用 PyCharm 和 Anaconda 开发平台进行算法编程。PyCharm 版本为 2021.2，Anaconda 版本为 Anaconda3-2021.03。通过 EdgeCloudSim[72]来搭建仿真实验环境，EdgeCloudSim 是以 CloudSim 仿真环境为基础，通过在 CloudSim 的基础上添加边缘计算模块，来实现对边缘计算的环境仿真，并且其仿真出的网络环境还可以考虑计算与网络等资源。本章实验通过 NS-3 来模拟 SDN 架构。SDN 的全局视图消息则使用统一的格式将其封装到一个 XML 文件中并通过仿真环境的网络模型进行读取，以便于网络全局视图消息的形式灵活且易于扩展。

3.5.2 数据集来源

本文提出的 SDN-CEP 算法是基于目标检测任务所设计的，选择常用的 Cifar-10 数据集进行实验验证。Cifar-10 是一个广泛应用于计算机视觉领域的数据集，用于物体识别任务。该数据集包含 60000 张 RGB 彩色图片，每张图片的尺寸为 32*32 像素，分为 10 个不同的类别，每个类别有 6000 张图片。其中，40000 张图片用于训练，组成 4 个训练集，每个批次包含 10000 张图片。剩余的 20000 张图片中，一半用于验证，另一半用于测试，形成一个测试集[73]。实验所使用的数据集特征如下表所示：

表 1 数据集信息表

Table 1 Data set information table

数据集	数据大小	类别
训练	40000	10
验证	10000	10
测试	10000	10

3.5.3 实验分析

为展示 SDN-CEP 算法在时延和资源利用方面的优化效果，实验模拟的网络环境和边缘节点性能采用随机生成的方式，DNN 模型与 DQN 算法则使用 Python 软件下的 PyTorch 框架实现。在算法的初始阶段中，EdgeCloudSim 会读取封装网络全局信息状态的视图文件，通过利用当前网络状态信息、任务推理模型以及从经验回放池中采样的样本作为神经网络的输入层，采取的模型划分动作为输出层并将最佳动作保存为参数，EdgeCloudSim 执行输出的最佳任务卸载方案。本章使用 ReLU 函数作为深度神经网络的激活函数，并将深度神经网络应用于边缘计算的模拟环境中，以验证模型推理任务处理的效率。

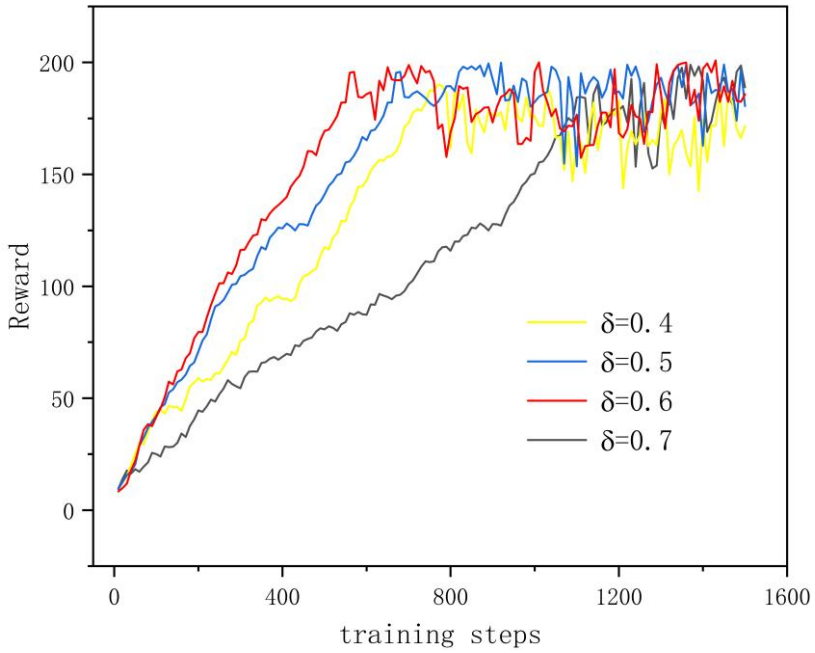

图 11 任务预测器不同阈值的奖励变化

Fig.11 Reward variations for different thresholds of the task predictor

在 SDN-CEP 算法中，相关实验的参数设置如下：学习率设置为 0.01，折扣因子为 0.09，经验池大小为 500，训练批次大小为 64。任务复杂度预测器的阈值大小决定任务卸载到云与边中的数量，其与模型的推理时延是呈正相关的，由奖励函数可知，时延与奖励值也是正相关的，所以阈值与奖励值也具有正相关性。图 11 为不同阈值下 SDN-CEP 算法对应的奖励值，可以发现随着阈值的增大，奖励值并未表现出更优的收敛性。在经过 1200 轮左右的迭代后，相关阈值的奖励基本都维持在一个较高的水平。当阈值为 0.6 时，只需要 600 轮左右训练就可以快速使奖励收敛到更高的值，其虽有阈值为 0.5 时的奖励趋势走向接近，但相同阶段阈值 0.6 的奖励值要高于 0.5，所以对比于其他的阈值，选择 0.6 为任务复杂度的阈值可以使算法具有更好的收敛性。

通过在不同的网络环境下执行 ResNet[74]，AlexNet[75]，GooleNet[76]等三种 DNN 模型，并与现有的推理算法进行对比，来衡量本章提出的基于模型划分的云边协同推理算法（SP）对缓解 DNN 模型推理的计算压力与时延优化效果。

1）仅云计算（OC）：DNN 推理任务全部分配到云端进行处理。

2）仅边计算（OE）：推理任务全部卸载到边缘节点中执行。

3）Neurosurgeon（NN）：依据预测模型动态的选择推理模型的最优划分点，结合网络带宽进行云边协同推理计算[22]。

4）DPTO 算法（DP）：使用 DRL 对推理任务进行按需划分，将任务卸载到边端计算资源最多的设备并与云端进行协同处理[67]。

1）静态环境下的推理时延

图 12 展示了各算法在不同计算需求下使用 AlexNet 网络模型的推理时延效果。在推理任务计算需求较少的情况下，边端计算能力足以支撑，无需上传到云端进行处理，此时 OE 方法的执行时延小于 OC 方法的传输时延，所以 OE 的推理时延性能优于 OC，且推理时延与 NN 方法相近。但随着任务数量的增多，计算需求增加、边缘服务器负载变大，导致 OE 方法的响应时间大于 OC 方法的传输时延，此时 OC 方法表现更优。SP 方法是利用基于 SDN 的云边协同架构优势，结合 SDN 全局调度的特点进行动态任务调度。算法还考虑了任务复杂度和边缘服务器在奖励状态下的计算能力，对比于 OC 方法，SP 有效解决了边缘环境面对高计算需求时计算资源紧张的问题。相较于 OC 方法，SP 还利用协同计算的方式减少了任务传输到云端时延以及数据在传输过程中可能面临的安全问题。所以面对不同的计算需求时，SP 方法的推理时延对比于其他两种算法相对较优。

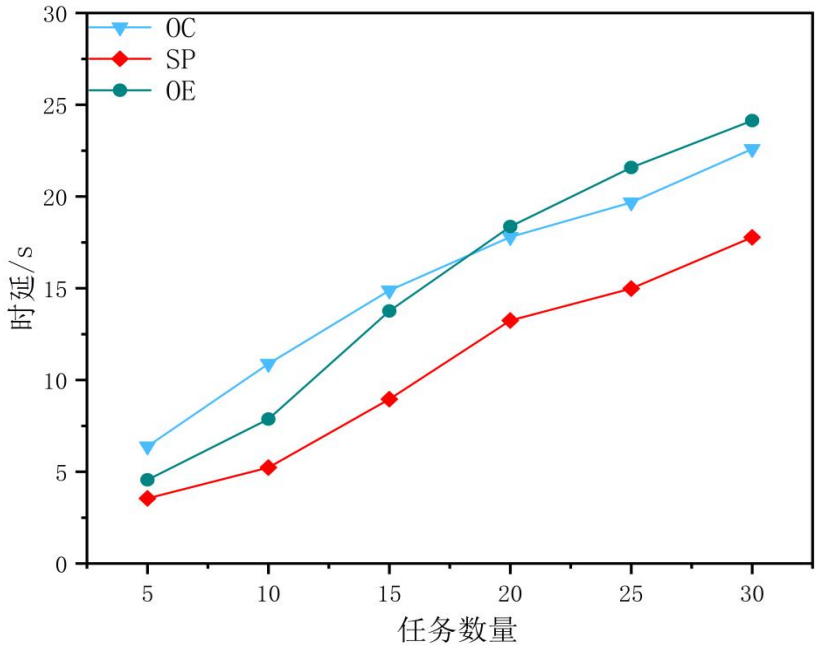

图 12 任务数量对推理时延的影响

Fig.12 Effect of task number on inference delay

图 13 展示了在不同网络带宽环境下，NN、DP 和 SP 这三种协同推理方法使用 AlexNet 网络模型处理相同批次任务所需时延效果图。当网络速率达到一定时，SP 方法的效果明显优于其他两种方法。在 NN 推理方法中，当网络环境的传输速率过低时，推理任务在边缘环境中执行，系统的 DNN 推理时延相对较低，此时面对需要高算力的任务时推理性能较差。所以当网络传输速率较高时，NN 方法倾向于将所有任务传输到云端处理，这与 DP 协同推理的思想相近，但 DP 方法考虑了网络计算设备的资源差异性与推理模型的参数大小，并在云边协同架构下使用 DQN 算法对任务进行按需划分与卸载，所以其推理时延优于 NN。

然而由于网络传输速率的差异，SP 方法的推理时延也存在差异。对于 DP 方法而言，网络带宽的变换会产生不同的划分策略，不同的卸载策略又影响着任务的执行时延，所以当网络传输速率较高时，DP 方法接近于 OC 方法。而 SP 是通过复杂度预测器将图片分别传输到云、边环境中执行，传输速率越快，其传输时延越小。此外，DP 方法在使用 DQN 算法处理推理模型按需划分时，没有考虑算法的过度估计问题。SP 方法却利用 SDN 控制器不断与网络环境进行交互，获取的全局视图提高了算法采样的效率，再经过 DQN 算法的执行优化，提高了算法输出最优模型划分策略的准确性，所以 SP 方法的推理时延要优于 DP 方法。

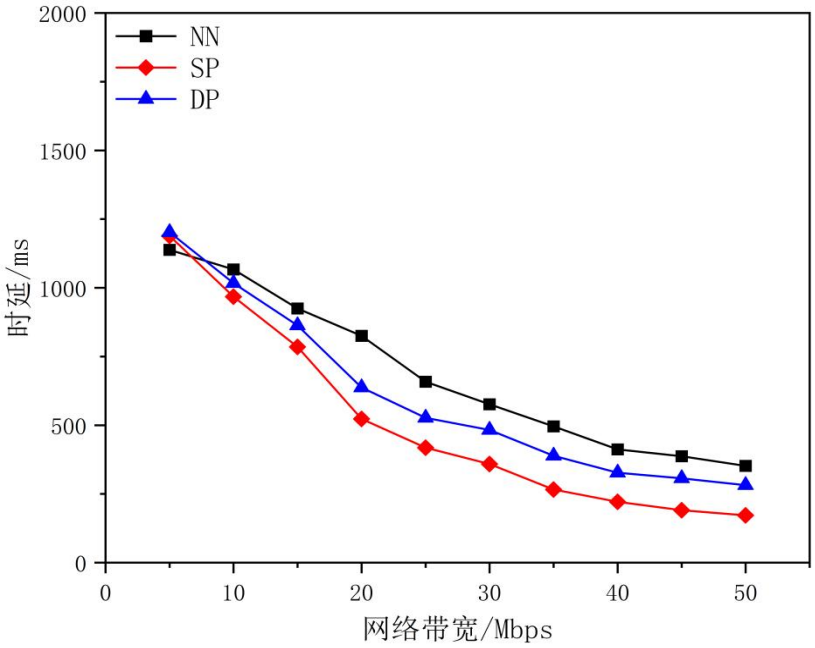

图 13 网络带宽对推理时延的影响

Fig.13 The influence of network bandwidth on inference delay

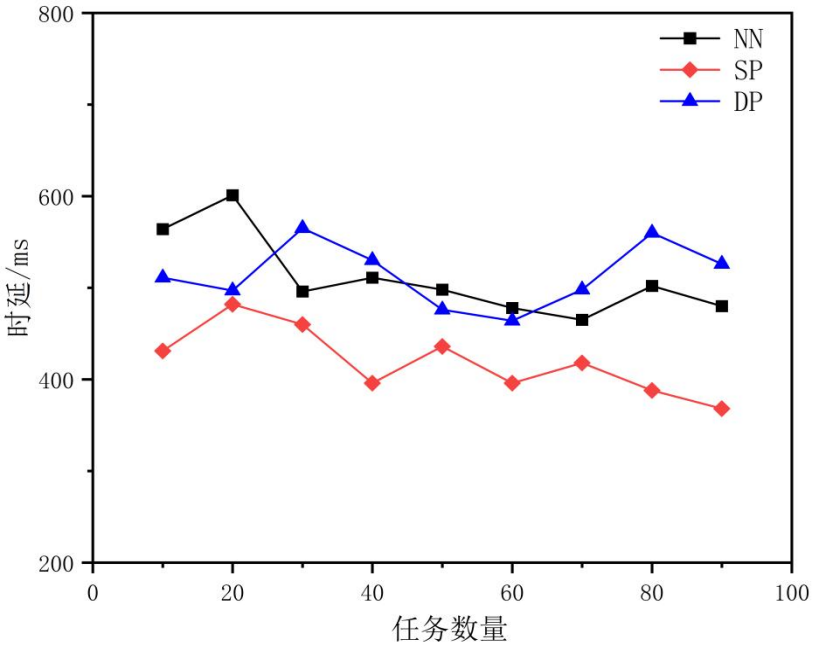

图 14 任务数量对推理时延的影响

Fig.14 Effect of task number on inference delay

2）动态环境下的推理时延

为进一步验证三种协同推理算法的性能，本章采用随机选取任务的方式并在动态的网络区间下与 NN、DP 算法相比较，各算法推理的时延效果如图 14 所示：可以看出，当网络状态与任务需求同时发生变化时，会导致了时延性能的波动，这对加载模型、传输中间结果等数据都有影响。所以当边缘节点的数量或模型划分点增加时，通信条件对算法延迟有很大的影响，在不同的网络状态下会表现出明显的波动。SP 算法因其通过 SDN 全局视图对网络状态进行实时的观察与更新，算法的状态空间还考虑了网络传输速率，导致任务数量的变化对算法的时延性能影响较小，从而表现出更好的推理性能。所以对比于其他两种协同算法，动态的网络环境对 SP 算法的影响相对较小，任务推理的时延优势也显而易见。

3）各 DNN 模型的推理时延

各 DNN 模型在不同推理环境下的时延效果如图 15 所示，协同推理环境相对于其他两种环境表现更优。云计算将 DNN 卸载到云端执行，其推理性能受限于网络传输速率，在某些限定条件下不能有效发挥模型最优推理性能。边缘计算的推理性能则受限于边缘服务器的计算能力。但是，协同推理计算方法弥补了单一环境执行 DNN 推理计算的劣势。所以采用协同环境执行推理计算，DNN 模型的时延性能表现要优于仅云与仅边环境。

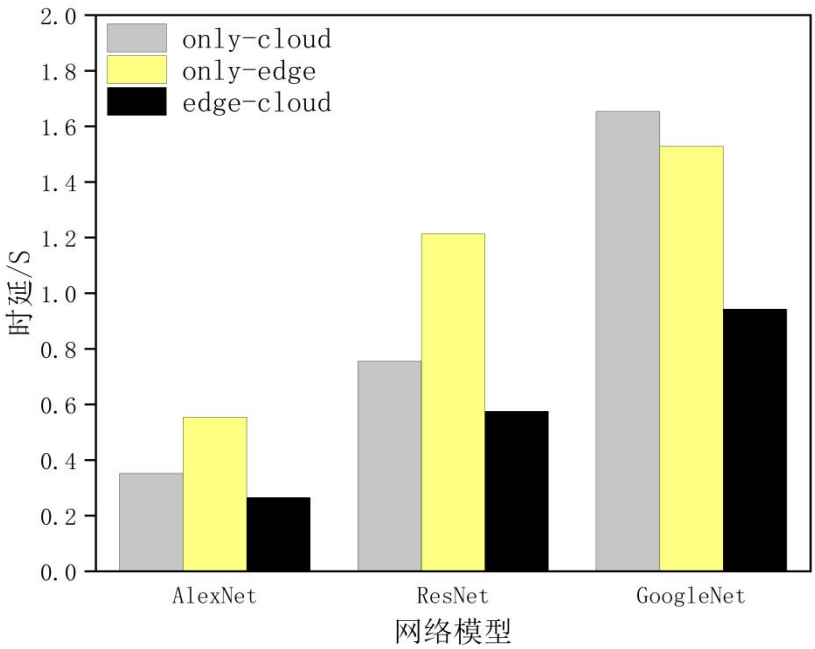

图 15 推理环境对模型推理时延的影响

Fig.15 The influence of inference environment on inference delay of model

在 SP 协同推理计算的网络环境中，DQN 算法将 SDN 技术与深度强化学习相结合处理模型的划分与卸载问题，通过 SDN 控制器的南北向接口连接云和边，实现云边环境的协调控制，从全局感知资源信息进行合理分配，并通过全局视图来反映网络资源的动态变化。此外，在网络资源限定的条件下，模型的推理时延与其本身的复杂程度呈正相关，从图中也可看出，随着模型层次的增加，推理所需的时延也越来越大，所以，面对计算密集型神经网络任务利用协同环境处理是必要的。

4）各算法的任务成本

任务成本包含了推理任务的计算成本与传输成本，此实验中本章对比了四种不同的推理算法，各算法在使用 AlexNet 模型执行不同任务数量时的推理性能如图 16 所示。相比于边缘计算环境和云计算环境，云边协同环境下的任务成本更低。OC 方法中云传输成本较高，但计算成本较低，而 OE 方法则相反。SP 方法采用基于 SDN 的云边协作机制，充分结合边缘和云的优点，并采用深度强化学习划分边缘环境中的推理模型，其算法的状态空间考虑了边缘服务器的计算能力与任务复杂度，从而可以根据奖励和损失不断的更新环境中的卸载策略，在奖励函数中，还考虑了任务成本以减少任务在分配过程中产生不必要的代价，所以 SP 方法的任务成本要优于其他几种推理方法。

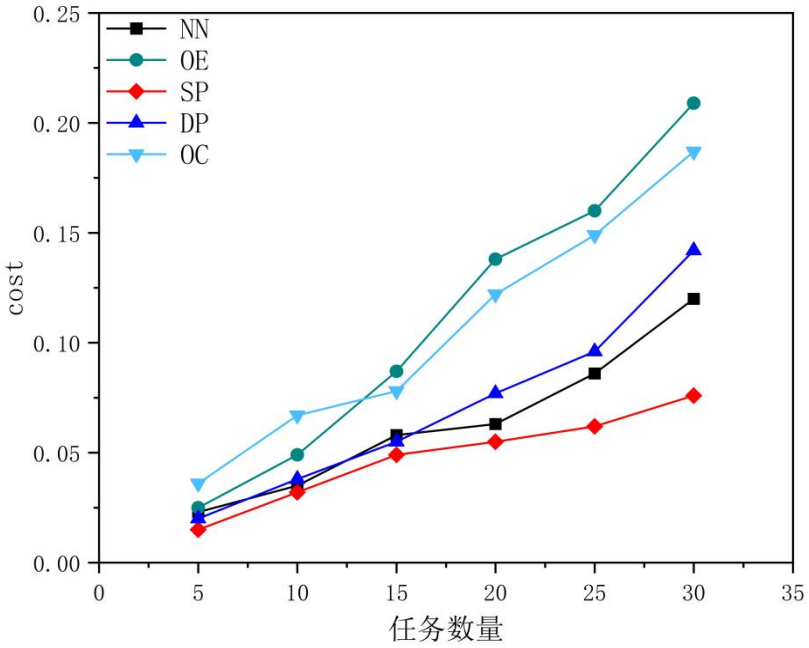

图 16 任务数量对成本的影响

Fig.16 The impact of the number of tasks on cost

3.6 本章小结

为了解决网络环境中因计算设备资源紧张造成的推理高时延问题，本章提出了基于模型划分的云边协同推理算法。该算法在基于 SDN 的云边协同推理架构中执行，利用 SDN 获取全局网络状态信息并通过边缘环境中的任务复杂度预测器决策任务的执行环境，实现云与边协同推理。通过改进的 DQN 算法对边缘环境中的推理模型进行划分，并结合实时网络状态信息决策最优的模型划分与卸载策略，从而解决边缘环境中模型推理计算资源紧张的问题，同时，优化 DNN 模型的推理效率。

4 基于自适应早期退出的推理加速算法

4.1 问题描述

随着 AIoT 的迅速发展，对模型推理计算低时延的需求也日益增高。当智能终端设备产生大量 DNN 推理任务时，会给计算资源有限的终端设备、边缘节点带来巨大的计算负载，从而导致推理高时延的问题[77-78]。在第三章中我们提出了基于模型划分的云边协同推理算法，在基于 SDN 的云边协同推理架构上实现了 DNN 推理模型按需划分的在线决策与分配，即通过纵向切分模型的方式，协同多个计算节点来解决 DNN 推理在网络环境中计算资源紧张的问题。然而，面对深层次的推理模型，如何进一步提高模型的推理计算速度，减少算法时延开销对实时 DNN 推理任务的影响则是本章主要的研究目标[69]。

DNN 推理模型可以分为链式与 DAG 两种结构[23]。对于链式结构的 DNN 推理模型，传统的早期退出技术采用遍历的思想评估各个模型分支的推理性能，通过判断是否满足时延与推理准确性要求来决策模型的早期退出分支。对于 DAG 结构的 DNN 推理模型，在其推理过程中，早期退出选择思想则是按照从第 n 个退出分支到第 1 个退出分支的顺序逐个进行评估，每次评估也都需要相应的算法分析。所以传统的早期退出技术会导致一定的评估时延开销，从而影响到深度神经网络任务整体推理性能[6]。因此，如何准确、高效的选择早期退出分支是当前所面临的挑战之一。此外，在实际部署中，模型早期退出技术需要能够适应不确定的计算环境和数据条件，如计算资源的变化、网络延迟等，如何在网络环境不确定的情况下选择最佳的早期退出分支也亟待解决[44]。

为了解决上述问题，本章从加速 DNN 模型推理速度的方案中评估了决策模型分支的规律，发现推理模型早期退出分支的选择会受到当前运行时网络环境参数变化的影响，因此本章从 DNN 模型推理深度的角度出发，通过分析实时网络环境参数与模型早期退出分支间的关系来决策模型的早期退出分支。摒弃使用遍历的传统方式，采用深度强化学习来自适应决策退出分支，根据网络参数与早期退出分支的历史经验训练神经网络模型，使其学习实时网络环境参数与推理模型早期退出分支之间的关系。这样，在后续的推理过程中，可以基于当前的网络状态信息实现模型最佳退出分支的预测。所以，本章首先在基于 SDN 的云边协同推理架构中加入了模型退出预测器，使用 NoisyDQN 算法对早期退出分支进行自适应决策。此外，算法还通过 SDN 的全局视图来获取网络运行时的环境参数信息，优化自适应决策算法的性能。与传统算法、机器学习算法相比，基于自适应早期

退出的推理加速算法提高了模型推理的速度、优化了模型早期退出在线决策的时延性能。

4.2 早期退出的协同框架

4.2.1 框架设计

为了压缩模型的推理层次、降低分支模型决策算法的时延，本章基于 SDN 的云边协同推理架构设计了早期退出的协同机制，用于支持模型分支的自适应早期退出选择方案。为了实现在实时网络环境信息下快速、准确预测最佳退出分支的目标，将模型退出预测器引入到协同推理框架的控制层，如图 17 所示，模型退出预测器是协同框架的核心组件，其主要功能是利用深度强化学习的 NoisyDQN 算法根据实时的网络环境信息与推理要求实现早期退出分支的自适应预测。

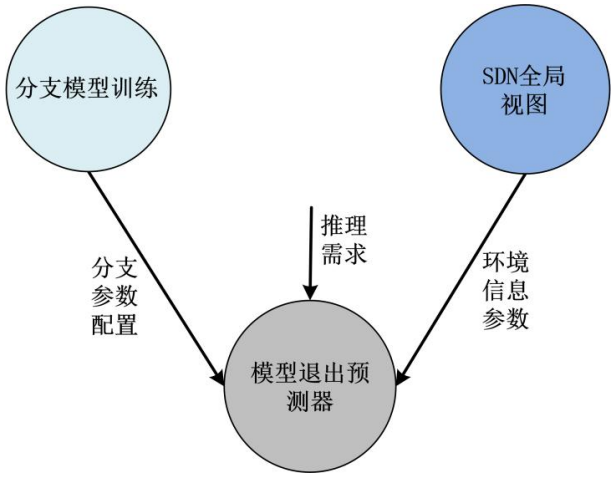

图 17 基于早期退出的协同机制

Fig.17 Collaboration mechanism based on early exit

4.2.2 模型分支的训练

随着深度神经网络的深入研究，模型的层次越来越深，相关参数的计算量也越来越大，使得网络计算资源面临巨大的压力。然而对于任务数据本身来说，简单的任务只需要少量的计算量即可达到目的，复杂的任务则需要更大的计算量，所以使用 BranchyNet[79]框架来完成分支模型的训练，通过利用不同的计算通路来执行推理计算，从而使得模型推理计算更加高效，在减少模型推理计算量的同时，也实现了 DNN 模型推理加速的目的。如图 18 所示，我们训练了一个带有多个早期退出分支的 AlexNet 模型，较小的分支网络结构对应较小的模型尺寸，因而其

推理所需的时延更短。BranchyNet 的训练框架包含一个输入点和一个以上的退出点，每个分支网络是整个深度神经网络的一个子集，在相邻且不重叠的层中。在引入早期退出分支结构之前，主分支结构即为原始的推理模型[80]。举例来说，如下图所示，我们可以对每个早期退出分支进行编号，以备后续参考。

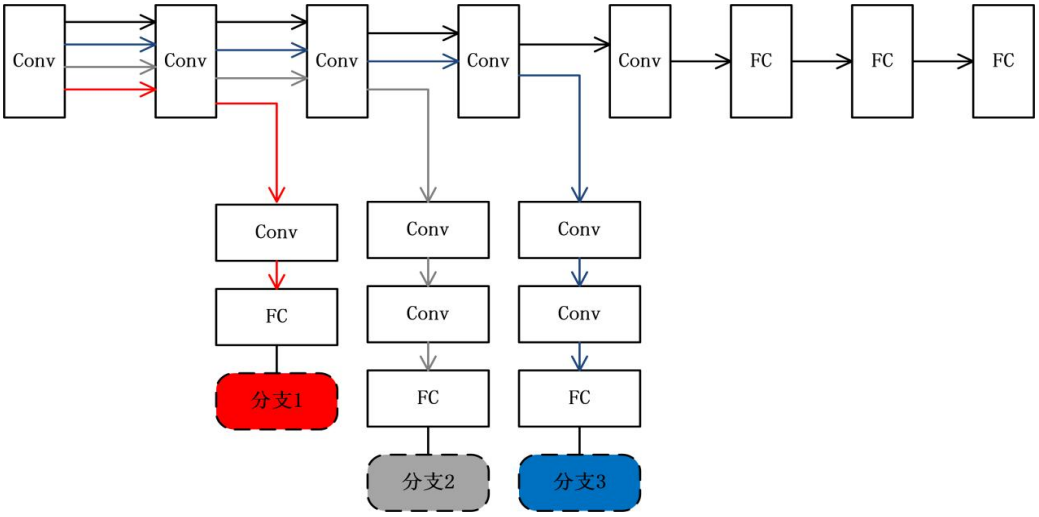

图 18 AlexNet 早期退出分支模型

Fig.18 AlexNet early exit from the branch model

使用 BranchyNet 框架训练 DNN 的早期退出分支网络可以划分为两个主要阶段，其一为分支模型的训练阶段。用于具有图像识别、分类预测等网络模型，一般会选择 softmax 交叉熵损失函数作为分支训练阶段的误差度量函数，即损失函数的表达式为：

$$L(\hat{y}, y; \theta) = -\frac{1}{C}\sum_{c \in C} y_c \log \hat{y}_c \tag{16}$$

$$\hat{y} = soft\max(z) = \frac{\exp(z)}{\sum_{c \in C}\exp(z_c)}, z = f_{exit}^{i}(x; \theta) \tag{17}$$

其中，x 为输入样本，y 为输入样本的真实标签，C 为所有可能的样本标签集合，$\hat{y}$为输入样本的预测输出，f_{exit}^{i} 为第 i 个早期退出分支，θ则代表某分支模型的参数集合。训练分支模型的目标是减少损失函数的值，使分支模型具有更好的表现，所以为了减少模型分支的训练时间，保证各分支模型的表现最优，本文对各个早期退出分支的损失函数进行加权和计算，每层的损失函数权重与每层的推理时耗相一致，以确保模型在高、低层退出分支之间的表现相均衡，从而使得推理分支

模型能够承接更大的并发量。综上所述，本章建立了一个联合优化目标函数，以便于得到推理性能较优的分支模型。优化目标函数如下所示：

$$L(\hat{y}, y; \theta) = \sum_{i=1}^{S} W_i L(\hat{y}_{exit_i}, y; \theta) \tag{18}$$

S 表示推理模型早期退出分支的集合。

分支模型的训练与普通的 DNN 模型训练方式是相一致的，都包含两个步骤：前向传播（Forward propagation, FP）与反向传播（Back propagation, BP）。在 FP 过程中，将训练数据集作为各个分支网络模型的输入，记录每个退出分支模型的输出以及对应的误差损失。而在 BP 过程中，误差利用神经网络从后向前传播，同时，使用梯度下降算法来更新权值[81]。

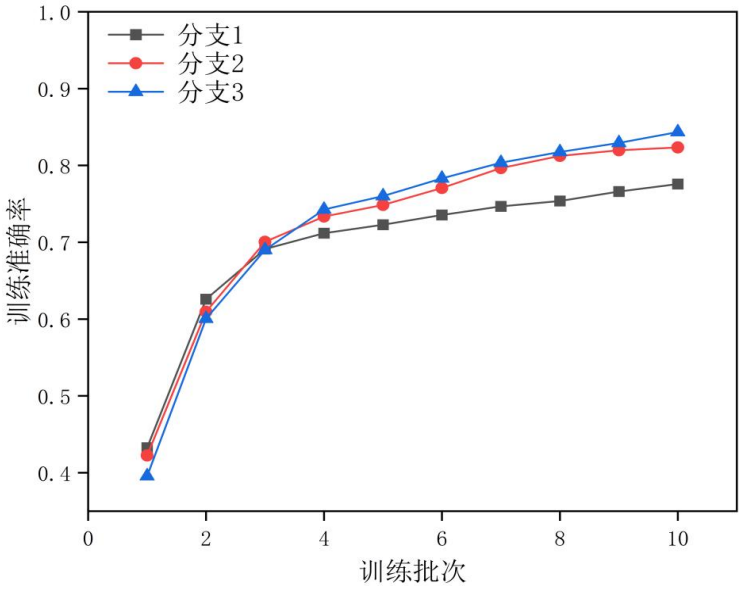

图 19 AlexNet 各分支模型的精度

Fig.19 Accuracy of AlexNet branch models

图 19 是对 AlexNet 模型各个分支模型推理正确率的相关描述，各分支模型的具体层次结构如图 18 所示，分支模型从左往右分别对应图 19 中模型分支 1 到 3，可以看出，分支模型的层次越深、其推理精度越高、延迟越大。

针对模型的推理阶段，在分支模型训练结束后，我们可以根据需求选择推理模型的早期退出分支，以达到推理加速的目的。当前，基于早期退出技术的 DNN 加速推理是从推理时延与准确性的角度出发，以求在尽量不损失推理精度的前提下减少模型的推理时延。

使用 BranchyNet 框架训练 DNN 分支模型需要对比每个分支模型退出点分类器中的阈值，以此来满足推理精度要求。具体而言，在每个退出分支中，将分类结果的信息熵作为对预测的置信度度量，并设定退出阈值。如果当前测试样本的熵低于退出阈值，即分类器对预测结果有信心，则该样本在这个退出分支退出网络。如果当前测试样本的上不低于退出阈值，则认为分类器对此样本的预测结果不可信，样本继续到网络的下一个退出分支进行处理。如果样本已到达最后一个退出分支，即网络的最后一层，则直接退出[82]。信息熵的可以定义为：

$$entropy(y) = \sum_{c \in C} y_c \log y_c \tag{19}$$

综上所述，基于 BranchyNet 框架训练的 DNN 分支网络模型的具体操作流程如下：在离线训练之前，使用者设定 DNN 推理模型的早期退出分支并设计相应的阈值。基于各个退出分支分类器中所设置的阈值多次训练推理模型，当每个早期退出分支的测试样本熵符合需求时，则结束此阶段的训练并记录相应的推理时延及样本的推理准确率。通常假设给定的推理模型经过训练后具有 m 个分支，每个分支的总执行时间为 $t_i(i=1,2,...,m)$。因此带有分支模型的总执行时延可以定义为：

$$\tau_{all} = \sum_{i=0}^{m} p_i * t_i, \sum_{0}^{m} p_i = 1 \tag{20}$$

其中，p_i 值为推理执行阶段由从 i 分支退出的任务数与任务总数的比值。因 DNN 模型离线配置所产生的训练成本开销对其在线推理过程产生的影响较小，所以本章忽略不计。

4.3 早期退出选择方案

4.3.1 自适应方法

自适应方法是一种根据环境和需求自动调整策略或参数的技术。它可以根据实时的数据和反馈信息，动态地改变自身的行为以适应不同的情况。自适应学习是常见的一种自适应方法，它可以根据数据的特征、噪声水平等因素，自动选择合适的学习率、正则化参数或网络结构，以提高学习算法的泛化能力和鲁棒性。为了有效应对 DNN 推理模型在执行决策时时间消耗较大的问题，本章采用 NoisyDQN 算法的理念，提出了一种基于自适应早期退出的推理加速算法（AEE）。

AEE 算法思想描述如下：当智能终端设备不断产生推理任务且汇聚到边缘环

境中处理时，对于任意的推理任务都需要监测实时网络运行的环境参数来决策最佳的早期退出分支，假设最初的网络环境参数数据为 Data=$\{exit, B, ES, A\}$，其中 *exit* 为早期退出分支、*B* 是网络传输速率、*ES* 为边缘节点的计算资源，*A* 是推理准确率。我们利用 NoisyDQN 算法对所采集并存储在边缘数据库中的网络信息数据进行周期性训练，通过学习网络运行时 *B*、*ES*、*A* 与 *exit* 等环境参数之间的关系模型，以便于根据实时网络环境对最佳早期退出分支的快速评估。具体的流程结构如图 20 所示，各模块的功能如下：

1）边缘环境数据库：接收从 SDN 全局视图中获取的实时网络环境参数和模型早期退出分支信息且定期更新。

2）训练模型：根据数据库存储的数据参数对神经网络进行周期性训练，并将训练好的模型参数实时更新到预测模型。

3）预测模型：从数据库中读取从全局视图中获取的实时网络环境参数，并结合 NoisyDQN 算法对推理模型的最佳早期退出分支进行预测。

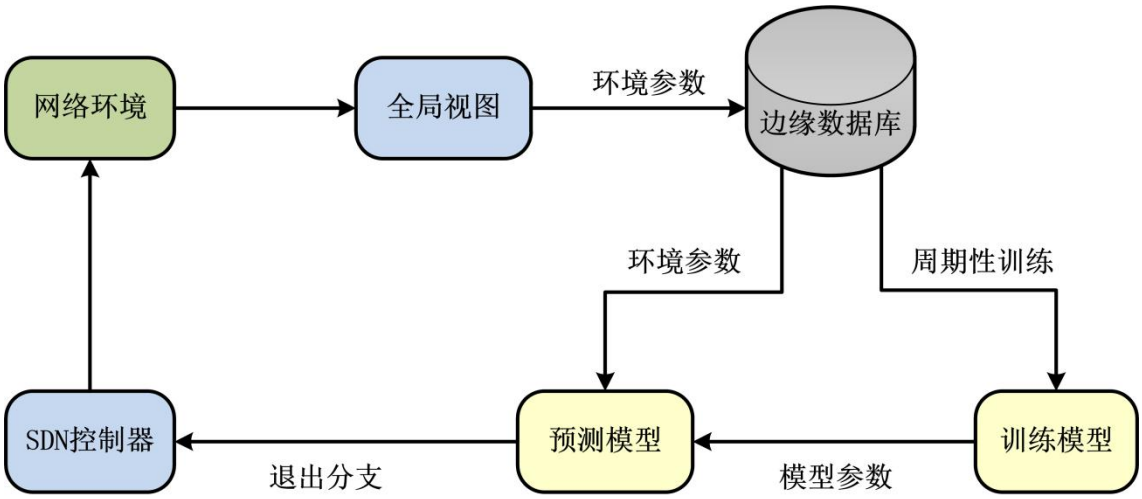

图 20 模型分支的预测流程

Fig.20 Prediction flow of model branches

4.3.2 算法设计

根据前文所描述的自适应早期退出选择方案，在模型退出预测器中设计了自适应早期退出选择算法。为了提高算法的鲁棒性、优化决策时延，本章使用深度强化学习中的 NoisyDQN 算法来实现分支的自适应决策。

DQN 算法是用于解决离散动作空间下的马尔可夫决策过程，但其在处理早期退出分支的决策问题时，因符合目标条件的早期退出分支并不唯一，容易受到过

度估计和不稳定性的问题影响，从而可能导致选择的早期退出分支并非最优。NoisyDQN 算法是 DQN 的一种变体，引入了参数化的高斯噪声来增加动作的随机性，从而改善探索与利用的平衡，减少过度估计和增加训练稳定性。在处理早期退出分支的决策问题中，NoisyDQN 算法通过将噪声引入神经网络的权重中，增加了探索的随机性，有助于更好的学习环境、探索环境并减少对过去经验的依赖。

在深度强化学习智能体中引入参数噪声，利用梯度下降法学习噪声参数和剩余网络价值，可以帮助智能体进行有效的探索策略。与 DQN 算法相比，利用 NoisyNet 代替传统的探索启发式方法，可以使用额外较小的计算成本实现相对于传统的启发式更优的效果，既保证了探索动作的多样化，又提高的探索效率[83]。

NoisyDQN 算法与 DQN 算法除了神经网络部分，其余的完全相同。该方法的优势在于向神经网络中注入高斯噪声，因此全连接层的噪声标准差成为可学习的参数，神经网络能够通过学习动态调整噪声大小。在仿真环境的单任务场景中，NoisyDQN 算法与 DQN 算法训练效果[84]，如图 21 所示：

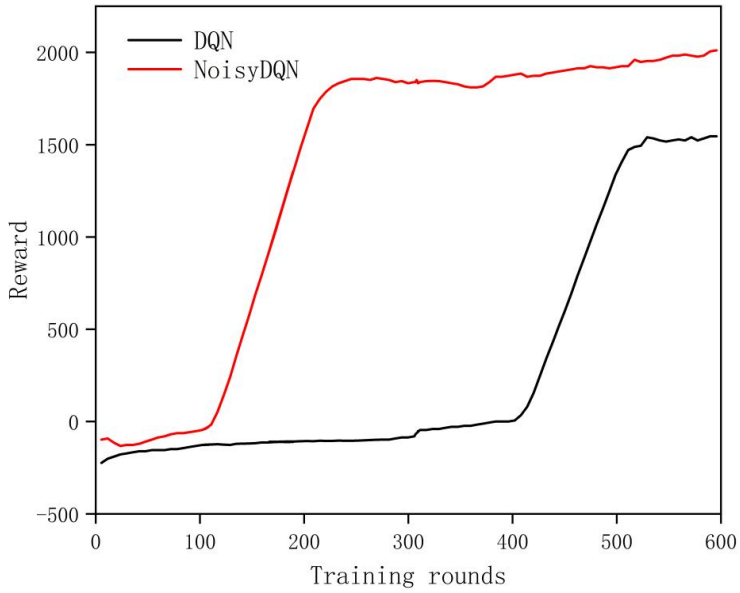

图 21 NoisyDQN 与 DQN 算法的对比效果图

Fig.21 NoisyDQN and DQN algorithm comparison effect

从图中可以看出，在 0 至 110 次左右迭代过程中，两种算法的奖励值接近且相应的奖励值较差。但在 110 轮迭代后，NoisyDQN 算法的奖励值迅速增加，在达到 250 轮迭代后，NoisyDQN 算法的奖励值收敛在 1900 左右。而 DQN 算法却在 400 轮之后奖励值才开始爬升，且最终的奖励值也小于 NoisyDQN 算法。所以综上可知，NoisyDQN 算法在训练速度和效果上都优于 DQN 算法。

使用 NoisyDQN 算法求解模型分支的最优早期退出分支，求解过程与 DQN 算法一致，需将问题建模为马尔科夫决策过程，即问题模型用一个四元组 (S, A, P, R) 来表示。

1）状态：状态表示算法对网络环境的感知信息，可以抽象为 S={B,ES}，其中 B 表示实时网络的传输速率；ES=(ES_1,ES_2,...,ES_n)表示 n 个边缘节点的计算资源。

2）动作：经过分支训练的 DNN 推理模型最多有 m-1 个早期退出分支，而动作空间由所有可能的动作组成，即选择其中一个模型分支来完成推理任务。所以，动作空间 $A=\{a_{exit}^1, a_{exit}^2, \ldots, a_{exit}^i, \ldots, a_{exit}^{m-1}, a_{exit}^m\}$，其中$a_{exit}^i$表示选择第 i 个早期退出分支来执行此次的任务，a_{exit}^m则表示选择主模型来处理任务。初始的模型分支是由算法随机选择的，之后在与网络环境的不断交互过程中逐渐趋于稳定。

3）奖励：奖励用于评估行动的有效性以便于进一步改进策略，不同的行动获得的回报也不相同，一般而言，奖励值越大表示选择的动作越好。但在执行推理任务中，因以减少推理时延为优化目标，所以根据时延特征将奖励值定义为：

$$R = \frac{1}{\tau_{all}} \tag{21}$$

其中，τ为模型的推理时延。从公式（21）可以看出，任务推理时延越高，奖励值越小。

噪声深度Q网络是对 DQN 的改进，噪声网络是指在Q函数中的每个网络参数上都加上一个高斯噪声，将原来的Q函数变成Q^{ε}，所以Q^{ε}表示噪声Q函数，其与 DQN 根本区别在于在参数的空间上加入噪声。所以在处理预测早期退出分支问题时，本章通过Q^{ε}来确定在状态 s 时选择动作 a 后所得到的奖励期望，以便于更好的替换函数中的高斯噪声，定义如下：

$$Q^{\varepsilon} = E(R + \gamma \times Q^{\varepsilon}(s', a')) \tag{22}$$

NoisyNet 是一个神经网络，其权值和偏差会受到噪音的影响。所以通常将 NoisyNet 用数学表示为：$y = f_{\theta}(x)$，x 为输入，y 为输出，θ表示噪音参数，传统的线性神经网络层用 y=wx+b 来表示，而带噪声的线性层定义如下所示：

$$\begin{aligned} w &= \mu^w + \sigma^w \odot \varepsilon^w \\ b &= \mu^b + \sigma^b \odot \varepsilon^b \end{aligned} \tag{23}$$

对于 NoisyNet 的数学定义中，θ的具体定义为：

$$\theta = \mu + \sum \odot \varepsilon \tag{24}$$

其中(μ，∑)为一组可学习的参数向量集合，用ζ表示，⊙为逐元素乘法，ε为噪声的矢量。噪声产生的方式主要有两种：独立的高斯噪声与分解的高斯噪声，本文采用分解高斯噪声的方式，在有效减少噪声数量的同时，也可减少推理模型的计算压力，产生噪声的计算公式为：

$$\varepsilon_{i,j}^{w} = f(\varepsilon_i) f(\varepsilon_j) \tag{25}$$

$$\varepsilon_j^{b} = f(\varepsilon_j) \tag{26}$$

其中 ε_i， ε_j 表示输入与输出的单位高斯噪声变量，f 函数的表达式为：

$$f(x) = \mathrm{sgn}(x)\sqrt{|x|} \tag{27}$$

算法的损失函数可以定义为：

$$\mathrm{L}(\zeta) = E[E_{(s_t,a_t,R_t,s_{t+1})\sim D}[R + \gamma \max_{a^*\in A} Q^{\varepsilon}(s_{t+1}, a^*, \varepsilon'; \zeta^{-}) - Q^{\varepsilon}(s_t, a_t, \varepsilon; \zeta)]^2] \tag{28}$$

其中ζ表示为一组可学习的参数向量集合，D 为经验池，γ表示折扣因子，R 为任务推理时延的奖励值， $a*$ 为目标动作，ε则表示噪声参数。

符合推理需求的退出分支未必是最佳的模型分支，因此需要将网络环境中的服务器计算能力、计算资源以及网络带宽等信息输入深度神经网络中进行学习，以得到最优的早期退出分支。因此，本章结合 NoisyDQN 算法和基于 SDN 的云边协同推理架构优势，通过 SDN 观察边缘环境中的计算资源和带宽情况，然后根据这些输入信息执行早期退出分支的选择动作，使其累积奖励值最大化。此外，噪声网络的加入提高了智能体的探索能力，减少了算法选择的随机性，优化了早期退出分支的决策性能。AEE 算法的伪代码如下所示：

算法 2：基于自适应早期退出的推理加速算法

输入：*exit*={1,2...*m*}，*A*，*B*，各边缘节点与云端资源

输出：退出分支 $exit_i$

1：初始化经验回放单元；

2：随机初始化参数θ、θ'与噪声参数ε；

3：for episode in *T* do：

4： 观测得到初始状态 s_1；

5： for each task do：

6： 使用贪婪策略随机选择动作a_{exit}^{t}；

7： 执行动作a_{exit}^{t}，观察奖励 R_t 并通过全局视图收集下一个状态 s_{t+1}；

8： 将(s_t,a_t,R_t,s_{t+1})存储到经验回放单元中；

9： 从经验池中随机采用一批样本 h；

10： for j=1 to h do：

11： 令 $y_i=\begin{cases}R_j, s_{j+1}\text{是最终回合}\\ R_j+\gamma\max Q^{\varepsilon}(s_{j+1},a',\zeta';\theta'),\text{其它}\end{cases}$；

12： 使用梯度下降策略通过$(y_i-Q^{\varepsilon}(s_j,a_j,\zeta'';\theta))^2$对$Q^{\varepsilon}$网络参数进行更新；

13： end for

14： 每隔固定步数对目标网络参数进行更新$\theta'=\theta$；

15： end for

16：end for

4.4 实验分析

为验证本章提出的 AEE 算法在预测模型退出分支的决策时延开销与推理准确率的性能，我们使用 CloudSimSDN[85]仿真工具构建实验环境。针对 DNN 推理模型的早期退出分支，我们使用 BranchyNet 框架结合 Python 软件中的 PyTorch 进行离线阶段的训练。前文我们选择 AlexNet 神经网络作为训练模型，设计了分支网络，并详细说明各分支网络的推理精度。为了充分验证 AEE 算法的有效性，我们还选择了 ResNet50 作为基础模型，并训练了 5 个早期退出分支，如图 22 所示，每个退出分支之间间隔 9 层，以此来突出模型的推理深度差异。模型各退出分支的推理精度以及单位平均推理时延如下表所示：

表 2 ResNet50 分支模型的推理精度与时延

Table 2 Inference precision and delay of ResNet50 branch model

退出分支	1	2	3	4	5	6
推理精度	69.87%	74.63%	80.63%	85.91%	91.27%	95.46%
推理时延	35.1ms	39.6ms	42.8ms	62.3ms	76.4ms	80.7ms

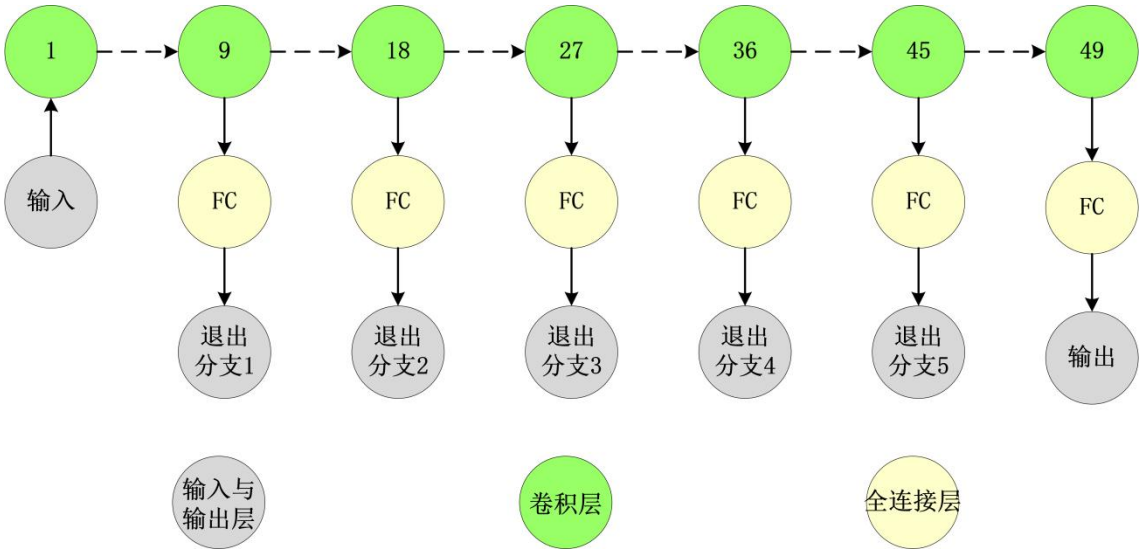

图 22 ResNet50 早期退出拓扑结构

Fig.22 ResNet50 exits the topology early

用于早期退出分支自适应选择的 NoisyDQN 算法与第三章相一致皆在 PyTorch 框架实现。预测模型的训练数据依旧使用 cifar-10 数据集。AEE 算法的相关参数如下表所示：

表 3 AEE 算法的相关仿真参数

Table 3 Simulation parameters of AEE algorithm

参数	值
学习率	0.01
折扣因子	0.9
经验回放池大小	500
训练批次	64
网络带宽	[10-20]Mbps

本章通过在动态网络环境下执行带有早期退出分支的 ResNet、AlexNet 与 GooleNet 深度神经网络模型，并与现有的早期退出分支决策算法进行比较，包括机器学习自适应预测算法[45]、基于遗传算法的在线划分分支网络[86]以及基于给定基线模型设计早期退出网络的方法[87]等，同时，对本章提出的 AEE 算法在提升 DNN 模型推理性能方面进行多角度的分析。

1）机器学习预测算法：采用随机森林（RF）与支持向量机（SVM）算法实现模型早期退出分支的自适应选择。

2）基于给定基线模型设计早期退出网络的方法（BEE）：通过探索早期退出分支和基线模型之间的密集连接，提取给定目标类的多层特征并利用面向类的方法确定早期退出的最佳位置，以在减少计算时间的同时提高分类精度。

3）基于遗传算法的在线划分分支网（GA）：建立一种基于加权平均值的新计算方法，通过区分 GA 进化过程中的分区和部署来估计给定 BranchyNet 和两层染色体 GA 的总执行时间，不仅能缩短执行时间，降低设备平均成本，而且需要更少的时间来获得最优的部署方案。

1）算法的预测精度分析

AEE 算法与 RF、SVM 算法对模型早期退出分支的预测性能评估如图 23 所示。随着推理模型任务数量的增加，三种自适应选择算法对模型早期退出分支的预测准确率逐渐提高。经过约 400 次任务推理后，三种算法的预测准确率基本相同，均表现出高精度的预测性能。

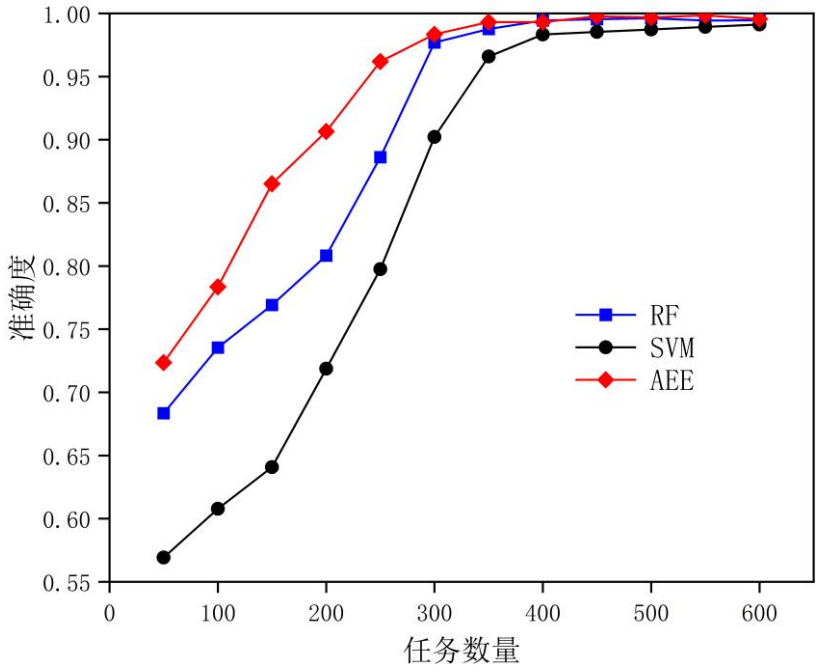

图 23 任务数量对算法预测准确度的影响

Fig.23 Effect of task number on prediction accuracy of algorithm

由图 23 可知当推理任务数量相对较少时，相较于 RF 与 SVM 算法 AEE 算法的预测性能更具优势，其主要原因在于 AEE 算法是借助 SDN 全局视图，通过实时获取的网络环境数据信息进行预测，其参数信息数据比其他两种自适应算法采集的运行时参数更为准确。此外，以推理时延为优化目标，AEE 算法还充分考虑

了网络负载，并以推理时延为 NoisyDQN 算法的奖励函数，可以在算法的迭代过程中不断更新退出分支的决策策略。

因此，AEE 算法不仅能够根据推理任务的奖励和损失动态调整推理分支的决策，减少不必要的推理时延。而且还利用 NoisyNet 神经网络计算 $Q^{\varepsilon}(s,a)$，确保了探索动作的多样性，也提高了探索效率，从而导致 AEE 算法的决策效率更优。因此，在推理的前期阶段，AEE 算法相对于 RF 与 SVM 算法能够更快地实现高精度预测，也为模型最佳早期退出分支的准确选择提供了有效支持。

2）算法的决策性能分析

随着 DNN 推理模型深度增加，自适应预测算法的决策时间也随之增加。如图 24 所示是三种自适应算法针对不同的推理模型选择分支退出点的平均单位决策时延性能。GA 算法通过模拟自然进化过程来选择最优退出分支，但由于是近似算法，无法确保获得绝对最优解。其决策时延性能在复杂问题下可能受到较大影响，导致性能较差。GA 算法以边缘设备的存储与计算资源来决策模型的早期退出分支。尽管有效降低了网络设备的计算代价，但随着模型分支与计算设备的增加，其与最优值之间的差距会随着问题的复杂程度而增大，从而影响了算法的稳定性和决策时延性能。

对比而言，BEE 算法基于基线模型设计早期退出网络，能在保持原始基线模型准确性的同时最大限度地降低决策时间。BEE 算法使用类优先级的思想来降低平均推理时延，相对于 GA 算法，BEE 算法表现出更好的分类性能和较短的计算时延。其使用参数损失函数并通过整个网络进行联合优化训练，提高整个神经网络的能量消耗和预测精度，因此在早期退出分支的平均决策时延性能上优于 GA 算法。

针对 AEE 算法，其使用 NoisyDQN 算法实现推理模型的早期退出分支的自适应预测，相较于其他两种算法，AEE 算法利用 NoisyNet 策略的诱导随机性来帮助高效探索，通过对空间进行扰动，可以推动探索，从而缩短决策时延以及提高模型分支选择的准确性。而且 NoisyDQN 通过 SDN 全局视图获取整个网络环境的负载状况，根据全局视图获得的实时网络环境信息训练预测模型，实现最优早期退出分支的自适应决策。这样可以更好地适应不同的网络环境，提高决策的准确性。相比之下，BEE 算法需要通过分析每个分支的推理性能来调整置信水平阈值，进而决策早期退出分支。虽然这种方法可以保证模型的推理精度，但分类率

和准确度水平会受到相应的影响。综上所述，AEE 算法在决策时延性能方面优于 BEE 算法。

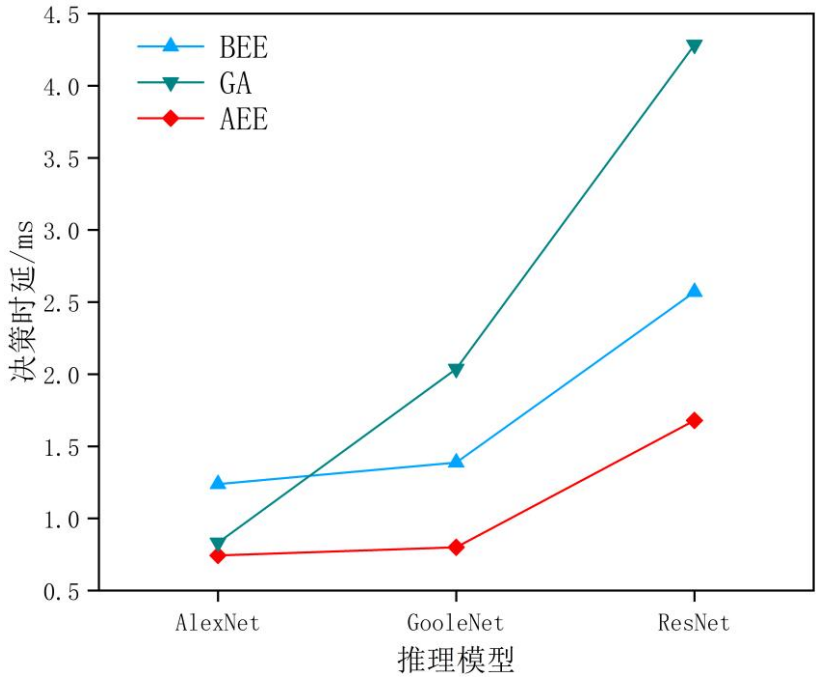

图 24 各推理模型的决策时延

Fig.24 Decision delay of each inference model

3）算法的时延增益分析

模型划分是将一个整体模型分割成多个子模型或组件的过程，通过模型划分可以将复杂的推理模型分解为更小、更易于管理和理解的部分，从而提高模型的部署以及系统计算能力。早期退出则旨在通过减少要计算的层的数量来减少执行推理的时延和计算成本。图 25 是以 AlexNet 深度神经网络为推理模型并结合了模型划分与早期退出技术的各算法在不同任务数量下的推理效果图，其纵坐标时延增益表示的是分支预测算法所带来的整体时延性能提升相较于原方案的倍数。

Edgent 方法是一种优化模型推理的方法，它通过在离线配置阶段训练分支模型，并在在线调优阶段尝试获得模型的最优分区。这样可以在给定的延迟要求下最大限度地提高推理精度，并将两个子模型部署到终端设备和边缘服务器中。DDPI 是一种改进后的分区推理算法，针对 DNN 结构进行了优化。它支持提前退出机制，即在推理过程中可以根据一定的条件提前结束推理，从而节省计算资源。此外，DDPI 还采用了改进的 DAG 式 DNN 分区推理方案，可以更高效地进行模型推理。SP 则是一种结合了 SDN 技术的分区推理算法。它以实时网络环境信息为算法的输入，并以推理时延为优化目标。SP 使用 DQN 算法来求解最优划分策

略，从而优化推理模型与计算资源的分配。这种方法在模型推理时延性能和推理准确率等方面都有明显的提升。DDPI 通过按需分区加速模型推理的思想来构建近似最优解，以求取最佳的分区方案。相同条件下，由于模型结构的限制，Edgent 可能无法获取最佳的推理分区方案，导致其时延开销相对较大。因此，在 Edgent 方法上使用 AEE 预测算法所产生的优化效果要低于 DDPI+AEE 方案。

SP 是以支持云和边缘之间的协作、DNN 分区推理算法为基础，旨在将一个大型计算任务拆分为多个子任务，并将它们分发给一组边缘设备，以充分利用这些边缘设备的可用资源，并通过任务分类器与云端推理模型形成并行协同推理。SP+AEE 方案还同时考虑了模型分区与部署情况下的决策模型的早期退出分支。通过结合 SDN 的全局视图，使用改进后的 DQN 算法按照细粒度对推理模型进行自适应分配，并使用 NoisyDQN 算法来完成早期退出分支的自适应决策。

NoisyDQN 是对 DQN 算法的优化，通过策略贪婪地优化值函数代替ε-greedy。将值函数的全连接层参数化为噪声网络，其中参数从每个重放步骤后的噪声网络参数分布中抽取，噪声的参数是通过梯度下降以及剩余的网络权重来学习的。参数上的梯度通过计算效率高的仿射函数相互关联，从而使得算法的计算开销与决策时延较小。此外，由于 DDPI 需要构建 DAG 和进行最小切割算法的评估，此举导致算法的运行时间开销较大。因此，SP+AEE 方案的决策时延性能表现要优于 DDPI+AEE 方案，且 NoisyNet 易于实现几乎不会增加计算开销。

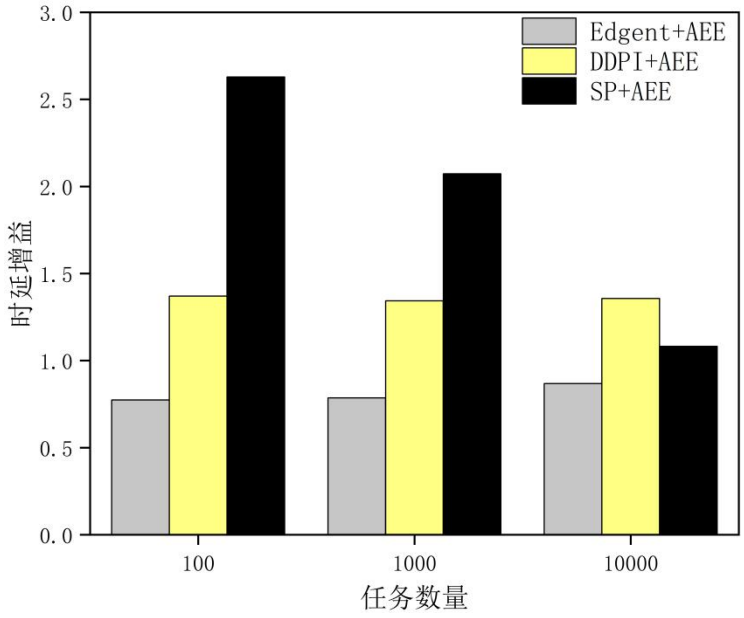

图 25 AlexNet 模型的时延增益

Fig.25 Delay gain of AlexNet model

从计算需求的角度来看，随着任务计算量的增加，网络计算资源的负载平衡变得更具挑战性。比较不同任务计算量下三种方案的表现，可以发现，SP 通过结合 AEE 可以有效解决模型划分与早期退出的决策优化问题，在计算需求相对较小的情况下，SP+AEE 方案的整体性能提升要高于其他两种方案。然而，随着任务计算量的逐渐增加和网络环境的复杂化，DDPI 针对模型内部结构的划分方法更有利于提升推理时延的性能，其所带来的时延增益效果略优于 SP 方法。

4.5 本章小结

为提升模型推理速度，本章以深度神经网络模型推理深度为出发点，使用早期退出技术来缩减推理模型的层次，减少其推理时延。首先，详细介绍了支持推理模型自适应早期退出选择方案的协同框架设计。然后，在基于 SDN 的云边协同推理架构中融入模型的早期退出技术，设计了模型退出预测器，并通过深度强化学习中的 NoisyDQN 算法实现推理模型早期退出分支的自适应预测。最后，与相关的自适应算法进行对比分析，详细描述了 AEE 算法在预测精度以及决策时延性能方面的优化效果。

5 总结与展望

5.1 总结

随着物联网的迅速发展，在边缘环境中部署和执行 DNN 推理任务的需求正日益增长，模型自身的推理层次也不断加深。然而，单一边缘节点的计算与内存资源有限，无法执行 DNN 模型的推理计算。因此，本文主要的研究目标是 DNN 模型的高效推理方案。具体研究工作总结如下：

首先指出以单一计算节点为核心的集中式处理模式无法高效处理 DNN 模型推理，面向云边协同计算模式成为有效的解决方案，且不同的边缘设备计算能力也存在较大差异，如何在网络环境与计算需求快速变化的情况下，实现 DNN 模型的推理性能的提升是我们的研究方向。

其次本文在基于 SDN 的云边协同架构中进行 DNN 模型的推理计算，利用 SDN 全局视图获取网络全局状态信息实现多个边缘计算节点之间的协作，并在边缘层中加入模型复杂度预测器，以支持更好的模型划分策略。基于云边协同推理架构提出了基于模型划分的云边协同推理算法，通过 DQN 算法与获取的全局状态信息对边缘环境中的推理计算模型进行按需分配，从而解决边端计算资源压力大的问题，实现推理时延、成本等多目标的优化。文章从不同模型的推理时延、任务数量以及计算成本评价 SDN-CEP 算法的性能并证明算法的有效性。

最后文章提出了基于自适应早期退出的推理加速算法，利用 BranchyNet 框架训练模型分支，在获取全局网络状态信息的基础上，使用 NoisyDQN 算法实现对推理模型早期退出分支自适应的预测，减少了算法的决策时延与模型推理的层次，进一步提升了 DNN 模型的推理速度。

综上所述，本文提出的研究策略有效解决了网络设备算力不足以及智能推理方法在网络环境和计算需求快速变化时性能下降的问题，为模型推理计算的高效实施提供了新的思路。此外，该研究也为边缘智能和软件定义网络的研究提供了新的方向。

5.2 展望

近年来，随着物联网的高速发展，智能移动设备的普及，边缘智能已成为当前重要的研究方向。本文主要研究当网络资源受限时，在基于 SDN 的云边协同架构中使用模型划分和早期退出技术优化 DNN 模型的推理效率。尽管在推理性能

方面取得不错的进展，但仍有问题值得改进与探索，未来可以从以下两个方面对该方向继续探讨、研究：

1）由于设备与网络环境等硬性条件的限制，本文中的实验是在仿真环境中进行的，因此所贴合的实际应用相对简单。所以在今后的研究工作中需要进一步考虑在更加复杂的应用场景中实验，为解决实际应用问题提供更加成熟的方案。

2）在更复杂的网络环境中，模型按需分配策略可能导致某些边缘服务器处于空闲或过载状态，因此需要更好的卸载策略来实现负载均衡。同时，若边缘服务器出现异常情况，需要高效的容错解决方案来确保模型推理划分和卸载顺利进行。此外，早期退出技术的引入增加了 DNN 模型的复杂性，降低了部署计划的适用性和灵活性。

参考文献

[1] Z.Chang,S.Liu,X.Xiong,et al. A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things, in IEEE Internet of Things Journal, 2021,8(18):13849-13875.

[2] FAN W, GAO L, SU Y, et al. Joint DNN Partition and Resource Allocation for Task Offloading in Edge-Cloud-Assisted IoT Environments[J]. IEEE Internet of Things Journal, 2023, 10(12): 10146-10159.

[3] 万朵,胡谋法,肖山竹等.面向边缘智能计算的异构并行计算平台综述[J].计算机工程与应用,2023,59(01):15-25.

[4] 郭斌,刘思聪,刘琰等.智能物联网：概念、体系架构与关键技术[J].计算机学报,2023,46(11):2259-2278.

[5] 李辉,李秀华,熊庆宇等.边缘计算助力工业互联网:架构、应用与挑战[J].计算机科学,2021,48(01):1-10.

[6] 张晓东,张朝昆,赵继军.边缘智能研究进展[J].计算机研究与发展,2023,60(12):2749-2764+2769.

[7] CHEN S, CHEN J, MIAO Y, et al. Deep Reinforcement Learning-Based Cloud-Edge Collaborative Mobile Computation Offloading in Industrial Networks[J]. IEEE Transactions on Signal and Information Processing over Networks, 2022, 8: 364-375.

[8] WU W, YANG P, ZHANG W, et al. Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning[J]. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, 2021, 17(7): 11.

[9] WANG X, HAN Y, LEUNG V C M, et al. Convergence of Edge Computing and Deep Learning: A Comprehensive Survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(2): 869-904.

[10] ZHANG L, XU J. Learning the Optimal Partition for Collaborative DNN Training With Privacy Requirements[J]. IEEE Internet of Things Journal, 2022, 9(13): 11168-11178.

[11] 陈思光,陈佳民,赵传信.基于深度强化学习的云边协同计算迁移研究[J].电子学报,2021,49(01):157-166.

[12] 陈玉平,刘波,林伟伟等.云边协同综述[J].计算机科学,2021,48(03):259-268.

[13] 李波,侯鹏,牛力等.基于软件定义网络的云边协同架构研究综述[J].计算机工程与科学,2021,43(02):242-257.

[14] 张文柱,余静华.移动边缘计算中基于云边端协同的任务卸载策略[J].计算机研究与发展,2023,60(02):371-385.

[15] ZHOU H, ZHANG W, WANG C, et al. BBNet: A Novel Convolutional Neural Network Structure in Edge-Cloud Collaborative Inference[J]. Sensors, 2021, 21(13): 4494.

[16] XUE M, WU H, PENG G, et al. DDPQN: An Efficient DNN Offloading Strategy in Local-Edge-Cloud Collaborative Environments[J]. IEEE Transactions on Services Computing, 2022, 15(2): 640-655.

[17] HU S, DONG C, WEN W. Enable Pipeline Processing of DNN Co-inference Tasks In the Mobile-Edge Cloud[C]//2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS). Chengdu, China: IEEE, 2021: 186-192.

[18] XUE M, WU H, LI R, et al. EosDNN: An Efficient Offloading Scheme for DNN Inference Acceleration in Local-Edge-Cloud Collaborative Environments[J]. IEEE Transactions on Green Communications and Networking, 2022, 6(1): 248-264.

[19] JIN Y, HUANG B, YAN Y, et al. Edge-Based Collaborative Training System for Artificial Intelligence-of-Things[J]. IEEE Transactions on Industrial Informatics, 2022, 18(10): 7162-7173.

[20] XU S, LIU X, GUO S, et al. MECC: A Mobile Edge Collaborative Caching Framework Empowered by Deep Reinforcement Learning[J]. IEEE Network, 2021, 35(4): 176-183.

[21] 路松峰,屠向阳,周军龙等.云边端协同的增量联邦学习算法[J].华中科技大学学报(自然科学版),2023,51(10):12-18.

[22] ZHANG X, CHEN J, WANG B, et al. An Edge-Cloud Collaborative Scheduling Algorithm Suitable for Electric Power Image Recognition Inference[C]//2021 IEEE 5th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC). Xi'an, China: IEEE, 2021: 1527-1531.

[23] 王睿,齐建鹏,陈亮等.面向边缘智能的边缘协同推理综述[J].计算机研究与发展,2022-11-19:1-17.

[24] Nisar K., Jimson E. R., Hijazi M. H. A., et al. A Survey on the Architecture, Application, and Security of Software Defined Networking: Challenges and Open Issues[J]. Internet of Things, 2020, 12: 100289-100301

[25] OKWUIBE J, HAAVISTO J, KOVACEVIC I, et al. SDN-Enabled Resource Orchestration for Industrial IoT in Collaborative Edge-Cloud Networks[J]. IEEE Access, 2021, 9: 115839-115854.

[26] REN X, AUJLA G S, JINDAL A, et al. Adaptive Recovery Mechanism for SDN

Controllers in Edge-Cloud Supported FinTech Applications[J]. IEEE Internet of Things Journal, 2023, 10(3): 2112-2120.

[27] PATRA S S, GOVINDARAJ R, CHOWDHURY S, et al. Energy Efficient End Device Aware Solution Through SDN in Edge-Cloud Platform[J]. IEEE Access, 2022, 10: 115192-115204.

[28] DU J, JIANG C, BENSLIMANE A, et al. SDN-Based Resource Allocation in Edge and Cloud Computing Systems: An Evolutionary Stackelberg Differential Game Approach[J]. IEEE/ACM Transactions on Networking, 2022, 30(4): 1613-1628.

[29] 韦睿,祝长鸿,王怡等.基于软件定义网络和移动边缘计算的车联网高效任务卸载方案[J].计算机应用研究,2023,40(06):1817-1824.

[30] Ashraf A.,Iqbal Z., Khan M.A., et al.Scalable offloading using machine learning methods for distributed multi-controller architecture of SDN networks[J]. The Journal of Supercomputing, 2022,78(7): 10191-10210.

[31] Hussain M, Shah N, Amin R, et al. Software-Defined Networking: Categories, Analysis, and Future Directions[J].Sensors. 2022; 22(15):5551.

[32] KOTACHI S, SATO T, SHINKUMA R, et al. Fault-Tolerant Controller Placement Model by Distributing Switch Load among Multiple Controllers in Software-Defined Network[J]. IEICE Transactions on Communications, 2022, 105: 533-544.

[33] JAHIER PAGLIARI D, CHIARO R, MACII E, et al. CRIME: Input-Dependent Collaborative Inference for Recurrent Neural Networks[J]. IEEE Transactions on Computers, 2020: 1-1.

[34] ZHANG B, LI Y, ZHANG S, et al. An Adaptive Task Migration Scheduling Approach for Edge-Cloud Collaborative Inference[J]. Wireless Communications and Mobile Computing, 2022, 2022: 1-12.

[35] LU H, DU M, HE X, et al. An Adaptive Neural Architecture Search Design for Collaborative Edge-Cloud Computing[J]. IEEE Network, 2021, 35(5): 83-89.

[36] Hussain, H., Tamizharasan, et al. Design possibilities and challenges of DNN models: a review on the perspective of end devices. Artif Intell Rev,2022, 55, 5109–5167.

[37] LI G, LIU Z, LI F, et al. Block Convolution: Toward Memory-Efficient Inference of Large-Scale CNNs on FPGA[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(5): 1436-1447.

[38] SU M, WANG G, CHOO K K R. Prediction-Based Resource Deployment and Task Scheduling in Edge-Cloud Collaborative Computing[J]. Wireless Communications and Mobile Computing, 2022, 2022: 1-17.

[39] YAN J, BI S, ZHANG Y J, et al. Optimal Task Offloading and Resource Allocation in Mobile-Edge Computing With Inter-User Task Dependency[J]. IEEE Transactions on Wireless Communications, 2020, 19(1): 235-250.

[40] Shan Yin, Yurong Jiao, Chenyu You, et al, Reliable adaptive edge-cloud collaborative DNN inference acceleration scheme combining computing and communication resources in optical networks[J].Opt. Commun. Netw. 2023,15, 750-764.

[41] CHEN S, SHEN H, WANG R, et al. Towards improving fast adversarial training in multi-exit network[J]. Neural Networks, 2022, 150: 1-11.

[42] D. Llorente-Vidrio, M. Ballesteros, I. Salgado, et al. Deep Learning Adapted to Differential Neural Networks Used as Pattern Classification of Electrophysiological Signals[J].in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022,44(9):4807-4818

[43] PASSALIS N. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits[J]. Pattern Recognition, 2020: 9.

[44] LASKARIDIS S, KOURIS A, LANE N D. Adaptive Inference through Early-Exit Networks: Design, Challenges and Directions[C]//Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning. 2021: 1-6.

[45] 钟长溢. 基于深度神经网络的边-云协同计算模型与算法研究[D].湖南大学,2021.

[46] RONG G, XU Y, TONG X, et al. An edge-cloud collaborative computing platform for building AIoT applications efficiently[J]. Journal of Cloud Computing, 2021, 10(1): 36.

[47] 徐诗浩. 边缘智能中云边联合任务推理和模型协同训练研究[D].北京交通大学,2021.

[48] WU W, YANG P, ZHANG W, et al. Accuracy-Guaranteed Collaborative DNN Inference in Industrial IoT via Deep Reinforcement Learning[J]. IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, 2021, 17(7): 11.

[49] 周俏妤. 基于神经网络模型精简和划分的边端协同推理机制的研究[D].西安电子科技大学,2021.

[50] Zhang W, Zhou H, Mo J, et al. Accelerated Inference of Face Detection under Edge-Cloud Collaboration[J]. Applied Sciences. 2022; 12(17):8424.

[51] ESHRATIFAR A E, ABRISHAMI M S, PEDRAM M. JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services[J]. IEEE Transactions on Mobile Computing, 2021, 20(2): 565-576.

[52] HUANG Y, QIAO X, DUSTDAR S, et al. Toward Decentralized and Collaborative Deep Learning Inference for Intelligent IoT Devices[J]. IEEE Network, 2022, 36(1): 59-68.

[53] HUANG Y, QIAO X, REN P, et al. A Lightweight Collaborative Deep Neural Network for the Mobile Web in Edge Cloud[J]. IEEE Transactions on Mobile Computing, 2021: 1-1.

[54] 李少波,刘意杨.基于改进深度强化学习的动态移动机器人协同计算卸载[J].计算机应用研究,2022,39(07):2087-2090+2103.

[55] 叶佩文,贾向东,杨小蓉等.面向车联网的多智能体强化学习边云协同卸载[J].计算机工程,2021,47(04):13-20.

[56] 张元龙,廖晓群,张佳庚.基于网络拓扑划分的大规模 SDN 控制器部署[J].计算机工程与设计,2022,43(09):2485-2492.

[57] NISAR K,JIMSON E R,HIJAZI M H A,et al.A survey on the architecture, application, and security of software defined networking: Challenges and open issues[J].Internet of Things,2020,12:100289.

[58] Ahmad, A.H. Scalability,et al.Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers[J]. Netw Syst Manage 29, 9 (2021).

[59] C. Tang, C. Zhu, N. Zhang, et al.SDN-Assisted Mobile Edge Computing for Collaborative Computation Offloading in Industrial Internet of Things[J].in IEEE Internet of Things Journal, 2022,23(9):24253-24263.

[60] DU T,LI C,LUO Y.Latency-aware computation offloading and DQN-based resource allocation approaches in SDN-enabled MEC[J].Ad Hoc Networks,2022,135:102950.

[61] SELLAMI B,HAKIRI A,YAHIA S B.Deep Reinforcement Learning for energy-aware task offloading in join SDN-Blockchain 5G massive IoT edge network[J].Future Generation Computer Systems,2022,137:363-379.

[62] ZHU X,ZHANG T,ZHANG J,et al.Deep reinforcement learning-based edge computing offloading algorithm for software-defined IoT[J].Computer Networks,2023,235:110006.

[63] LIN L, ZHANG L. Joint Optimization of Offloading and Resource Allocation for SDN-Enabled IoV[J]. Wireless Communications and Mobile Computing, 2022, 2022: 1-13.

[64] LI M, LI Y, TIAN Y, et al. AppealNet: An Efficient and Highly-Accurate Edge/Cloud Collaborative Architecture for DNN Inference[C]//2021 58th ACM/IEEE Design Automation

Conference (DAC). San Francisco, CA, USA: IEEE, 2021: 409-414.

[65] XIAO W, MIAO Y, FORTINO G, et al. Collaborative Cloud-Edge Service Cognition Framework for DNN Configuration Toward Smart IIoT[J]. IEEE Transactions on Industrial Informatics, 2022, 18(10): 7038-7047.

[66] CHEN X, ZHANG J, LIN B, et al. Energy-Efficient Offloading for DNN-Based Smart IoT Systems in Cloud-Edge Environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(3): 683-697.

[67] 刘先锋,梁赛,李强,等.基于深度强化学习的云边协同 DNN 推理[J].计算机工程,2022,48(11):30-38.

[68] 郭晓东,郝思达,王丽芳.基于深度强化学习的车辆边缘计算任务卸载方法[J].计算机应用研究,2023,40(09):2803-2807+2814.

[69] WANG Z, BAO W, YUAN D, et al. Accelerating on-device DNN inference during service outage through scheduling early exit[J]. Computer Communications, 2020, 162: 69-82.

[70] REN P, QIAO X, HUANG Y, et al. Edge-Assisted Distributed DNN Collaborative Computing Approach for Mobile Web Augmented Reality in 5G Networks[J]. IEEE Network, 2020, 34(2): 254-261.

[71] DONG C, HU S, CHEN X, et al. Joint Optimization With DNN Partitioning and Resource Allocation in Mobile Edge Computing[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 3973-3986.

[72] SONMEZ C, OZGOVDE A, ERSOY C. EdgeCloudSim: An environment for performance evaluation of Edge Computing systems[C]//2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). Valencia, Spain: IEEE, 2017: 39-44.

[73] DING Y, FANG W, LIU M, et al. JMDC: A joint model and data compression system for deep neural networks collaborative computing in edge-cloud networks[J]. Journal of Parallel and Distributed Computing, 2023, 173: 83-93.

[74] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[75] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[76] SZEGEDY C, LIU W, JIA Y,et al. Going Deeper with Convolutions[M]. arXiv, 2014.

[77] 高月红, 陈露. 移动边缘计算中任务卸载研究综述[J]. 计算机科学, 2022, 49(S02):

807-813.

[78] Chen S., Li Q., Zhou M., et al. Recent Advances in Collaborative Scheduling of Computing Tasks in an Edge Computing Paradigm[J]. Sensors, 2021, 21(3): 779-789.

[79] TORRES D R, MARTÍN C, RUBIO B, et al. An open source framework based on Kafka-ML for Distributed DNN inference over the Cloud-to-Things continuum[J]. Journal of Systems Architecture, 2021, 118: 102214.

[80] Na J, Zhang H, Lian J, et al. Genetic Algorithm-Based Online-Partitioning BranchyNet for Accelerating Edge Inference[J]. Sensors. 2023; 23(3):1500.

[81] WANG M, RASOULINEZHAD S, LEONG P H W, et al. NITI: Training Integer Neural Networks Using Integer-Only Arithmetic[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(11): 3249-3261.

[82] LI Y, WU Y, ZHANG X, et al. Energy-Aware Adaptive Multi-Exit Neural Network Inference Implementation for a Millimeter-Scale Sensing System[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2022, 30(7): 849-859.

[83] ZHANG L L, HAN S, WEI J, et al. nn-Meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices[C]//Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services. Virtual Event Wisconsin: ACM, 2021: 81-93.

[84] 陈铿浩.基于强化学习的智能小车控制研究[D].广东工业大学,2021.

[85] SON J, HE T, BUYYA R. CloudSimSDN-NFV: Modeling and simulation of network function virtualization and service function chaining in edge computing environments[J]. Software: Practice and Experience, 2019, 49(12): 1748-1764.

[86] NA J, ZHANG H, LIAN J, et al. Genetic Algorithm-Based Online-Partitioning BranchyNet for Accelerating Edge Inference[J/OL]. Sensors, 2023, 23(3): 1500.

[87] BONATO V, BOUGANIS C S. Class-specific early exit design methodology for convolutional neural networks[J]. Applied Soft Computing, 2021, 107: 107316.