数据处理单元赋能的智算中心网络拥塞控制机制

陈锦前，郭少勇，刘畅，亓峰，邱雪松
（北京邮电大学网络与交换技术全国重点实验室，北京 100876）

摘　要：针对智算中心集群间交互频繁造成网络拥塞频发导致智能业务实时性难以保障的问题，以数据处理单元（DPU）为核心载体构建了深度强化学习算法驱动的拥塞控制模型，利用剪枝与量化融合的方式对模型进行压缩，并通过知识蒸馏方法生成高效梯度增强决策树，实现调速动作与网络实时状态的精准匹配。仿真结果表明，所提机制在泛化能力和控制效果方面均优于现有方法，在多个压力测试场景中提升网络有效吞吐率与公平性指标JAIN10.8%和8.9%以上，降低P99端到端时延与丢包率17.31%和11.47%以上，降低并行计算场景下数据流传输任务完成时间11.23%以上，且具备应对网络状态突变的快速响应能力。

关键词：拥塞控制；多智能体深度强化学习；智算中心网络；远程直接内存访问网络；数据处理单元
中图分类号：TP393
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2025027

DPU empowered intelligent congestion control mechanism for the intelligent computing center network

CHEN Jinqian, GUO Shaoyong, LIU Chang, QI Feng, QIU Xuesong
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Addressing the issue of frequent network congestion due to high-frequency interactions between intelligent computing center clusters, which compromised the real-time performance of intelligent services, a congestion control model driven by deep reinforcement learning algorithm was constructed with the data processing unit (DPU). By integrating pruning and quantization, the model was lightweighted. Moreover, the model was transformed into the efficient gradient-boosted decision tree through knowledge distillation method, allowing for precise matching of control actions with real-time network conditions. Simulation results show that the proposed mechanism is demonstrated to outperform existing methods in terms of generalization capability and control effectiveness. The network's effective throughput and fairness index JAIN are increased by more than 10.8% and 8.9%, respectively, across various experimental scenarios. P99 end-to-end latency and packet loss rate are reduced by more than 17.31% and 11.47%, respectively. The completion time of data flow transfer tasks in parallel computing scenarios is decreased by more than 11.23%. Additionally, rapid response capabilities to sudden changes in network status are exhibited.

Keywords: congestion control, multi-agent deep reinforcement learning, intelligent computing center network, remote direct memory access network, data processing unit

收稿日期：2024–12–31；**修回日期：**2025–02–09
通信作者：亓峰，qifeng@bupt.edu.cn
基金项目：国家自然科学基金资助项目（No.62322103）；北京市自然科学基金资助项目（No.4232009）；中央高校基本科研业务费专项资金资助项目（No.2023ZCTH11）
Foundation Items: The National Natural Science Foundation of China (No.62322103), The Natural Science Foundation of Beijing (No.4232009), The Foundation of Central University Basic Research Projects (No.2023ZCTH11)

0 引言

随着生成式人工智能技术飞速发展导致算力需求激增，算力中心正在向集约化方向发展，数据中心从“云化时代”转向“算力时代”[1]。智算中心是服务于人工智能的数据计算中心，将算力资源全面解耦，追求计算资源和存储资源极致的弹性供给和利用[2]，网络提供中央处理器（CPU, central processing unit）、图形处理单元（GPU, graphics processing unit）和存储之间总线级的高速连接，因此网络性能成为提升智算中心算力效率的关键要素[3]。远程直接内存访问（RDMA, remote direct memory access）协议通过绕过操作系统网络协议栈进而缩短I/O排队时延，实现低时延、高吞吐率的数据传输，因此逐渐在智算中心网络中得到实际应用[4]。在无损网络中，RDMA协议采用优先级流量控制（PFC, proiority flow control）机制通过暂停流传输的方式防止数据包丢失，但该方式已经被研究者证明极易导致拥塞蔓延、死锁等问题，造成网络整体性能大幅度下降[5-6]。在有损网络中，RDMA协议利用重传机制保障数据准确传输至目的地，但会导致网络时延增加和有效吞吐量下降。目前拥塞控制算法已被证实可最大限度地减少无损网络中PFC帧激活，并减少有损网络中数据包重传概率，从而提高网络整体性能与效率[7-8]，因此成为工业界和学术界的关注点。拥塞控制机制通过持续监测网络的拥塞情况，根据网络的整体性能，包括吞吐量、时延、丢包率等指标，动态调整发送方的数据发送速率，使网络的负载保持在合理的范围内[9]。

然而，智算中心网络流量时变性极强，当流量突然停止传输或者大量新流量加入传输队列时，已有基于规则的拥塞控制算法如Blot[10]、DCQCN[11]、Swfit[12]和HPCC[13]等由于缺乏学习历史经验的能力，难以对拥塞控制策略进行精准调整，并将网络收敛至新的平衡。因此，陆续有研究者提出了基于机器学习的拥塞控制算法以提升其泛化与自优化能力[14]。文献[15]提出了一种基于在线学习的拥塞控制算法PCC Vivace，该算法能够持续优化做出更匹配网络流量特征的拥塞控制动作，但该算法的在线学习过程难以保持稳定，无法保障往更优模型演化。文献[16]对PCC Vivace算法的奖励函数进行了优化扩展，并提出了深度强化学习驱动的拥塞控制框架Aurora，但该框架只适用于稳态网络，同时难以适应复杂且不对称的网络拓扑结构。文献[17]以软件定义网络为基础，构建了多任务深度强化学习支撑的拥塞控制算法，从而实现网络整体性能优化。文献[18]面向异构网络提出了一种基于迁移学习的拥塞控制算法，针对不同制式网络构建差异化控制模型以匹配流量特征，然而该算法对网络设备硬件资源要求极高。文献[19]提出了适用于多类业务数据流的智能拥塞控制算法，能在保障业务服务质量的同时最大化网络吞吐量，但该算法在动态场景中的收敛速度存在不足。文献[20]提出了基于改进型确定性梯度策略算法的深度强化学习拥塞控制算法，提升了算法收敛速度，并实现了传输速率和带宽利用率的最佳平衡。总体而言，基于机器学习的拥塞控制方案通过离线训练或在线学习的方式迭代更新策略函数，求得在约束条件下使奖励函数最大的速率控制决策动作，但是策略函数通常需要神经网络进行拟合。因此，尽管基于机器学习的拥塞控制方案应用前景很好，但由于神经网络存在极高的计算开销从而干扰数据路径传输性能，现有的解决方案通过降低推理决策频率来解决该问题。例如，文献[21]将拥塞控制任务分解为不同时间尺度的2个子任务，利用深度强化学习算法在控制器中生成粗粒度控制策略参数，并辅助网络调谐器完成细粒度调速决策，该方案有效降低了拥塞控制算法的部署资源开销和决策时间，但该方案存在滞后性与粗粒度问题，难以与智算中心网络复杂多变的流量特征进行匹配。同时，为了保障拥塞控制算法有效运行，拥塞控制算法决策时间必须小于数据包往返时间（RTT, round-trip time）[22]，部署RDMA协议的智算中心网络RTT通常在10 μs左右，而基于神经网络的拥塞控制算法决策时间远远大于该指标。因此，亟须提出一种细粒度、低开销、智能化的拥塞控制机制以满足日益增长的智算中心网络传输需求。

数据处理单元（DPU, data processing unit）作为以数据为中心构建的可编程处理器，能高效承担网络协议处理和数据流传输管控等工作负载[23]。同时DPU能根据网络状态信息快速做出响应，进一步提升拥塞控制效果，为智算中心网络部署智能拥塞控制算法提供了解决方案。数据处理单元参考设计架构如图1所示，主要包含以下核心组件。1) 控制平面组件。由通用处理器核构成，

负责执行各类流控算法完成数据流细粒度传输控制。2) 数据平面组件。由可编程硬件加速引擎（如FPGA、ASIC）构成，主要包括特定数据包加速处理、网络协议解析、流量过滤与转发、安全加密等功能。3) I/O接口组件。主要包括主机侧I/O接口、高速网络I/O接口和主存I/O接口，主机侧I/O接口负责完成DPU与终端设备的集成，高速网络I/O接口负责实现DPU与高速网络的连接，主存I/O接口负责借助片上内存完成控制平面组件与数据平面组件之间的信息交互。

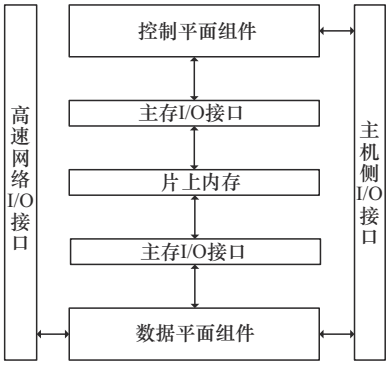


图1　数据处理单元参考设计架构

综上所述，本文研究工作如下。

1) 提出了一种数据处理单元赋能的拥塞控制机制（DECC, data process unit empowered congestion control mechanism），为保证网络状态以及多数据流特征支撑拥塞控制决策的精确性与实时性，利用DPU加速带内遥测数据包处理实现网络状态的高效精准感知，并构建多智能体深度强化学习协同的拥塞控制模型。采用分布式决策与集中式训练的方式，根据带内遥测结果完成发送端数据流速率的快速调整以及模型的迭代优化更新，最终实现有效吞吐率、丢包率、数据包平均时延和公平性各方面指标的全局最优。

2) 提出了一种融合模型压缩和知识蒸馏技术的深度神经网络模型轻量化方法，实现拥塞控制模型与资源受限DPU的融合适配，并有效降低拥塞控制算法决策时间。首先，采用多层自适应统一剪枝技术将训练完成的深度神经网络模型中的非关键神经元删除，降低模型权重数量；其次，采用通道级统一量化机制将剪枝模型中的浮点数权重和激活值转换为低精度整数，进一步降低模型计算复杂度；最后，以梯度增强树为学生模型进行知识蒸馏，将二值神经网络模型转换为对计算量要求极小的决策树结构，在将推理时间缩短近原来的$\frac{1}{300}$的同时，保障网络拥塞控制效果。

3) 仿真结果表明，DECC在泛化能力和控制效果方面均优于现有算法，在多个实验测试场景中提升网络有效吞吐率和公平性指标JAIN10.8%和8.9%以上，降低P99端到端时延与丢包率17.31%和11.47%以上，降低并行计算场景下数据流传输任务完成时间11.23%以上，且具备应对网络状态突变的快速响应能力，实现了智算中心网络海量数据流高效公平传输。

1　系统框架

针对智算中心叶脊网络架构提出DPU赋能的智能拥塞控制机制系统框架，该框架包含三大关键组件，分别为支持带内遥测功能的可编程交换机、内嵌DPU的计算节点和DPU拥塞控制智能体，如图2所示。DPU拥塞控制智能体从带内遥测数据包中提取到数据流特征信息后实时生成发包速率调整动作。内嵌DPU的计算节点从拥塞控制智能体中获取经验，完成对拥塞控制模型的迭代更新优化与轻量化，并将当前最优模型下发至拥塞控制智能体。智算中心网络拥塞控制机制主要包括以下关键组件。

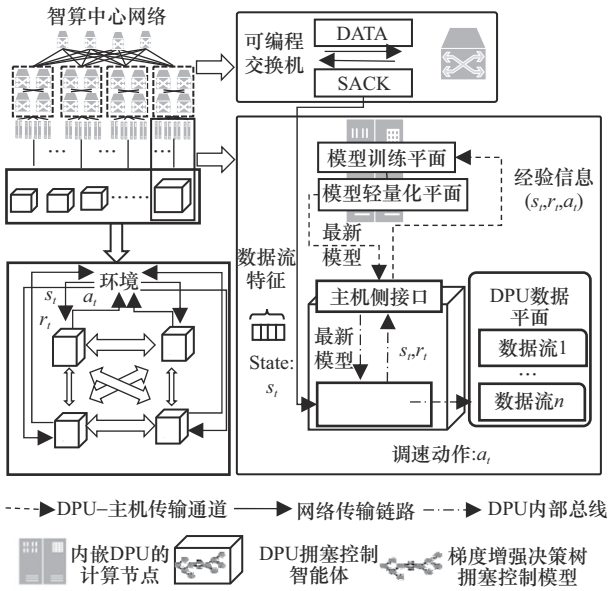


图2　智算中心网络拥塞控制机制系统框架

1) 内嵌DPU的计算节点。计算节点主要由模型训练平面和模型轻量化平面两大关键组件构成，

其中模型训练平面通过结合自学习与经验回放的方式对拥塞控制模型进行持续训练，以提升模型的泛化与持续进化能力。模型轻量化平面利用结合模型压缩和知识蒸馏的方式将训练完成模型转化为梯度增强决策树，并将该决策树下发至DPU控制平面。

2) 可编程交换机。可编程交换机采用P4语言集成实现带内遥测功能，数据包经过可编程交换机转发后会自动生成SACK（Switch ACK）包并向发送方反馈网络链路状态和数据流信息，包括队列长度、链路带宽利用率、队列长度变化率、数据流尚未确认字节数等，为部署在发送端DPU内的拥塞控制智能体提供了精准丰富的环境状态信息。

3) DPU拥塞控制智能体。智能体部署在DPU内部控制平面的通用处理器中，根据带内遥测获取的数据流信息运行主机侧下发的拥塞控制决策树对数据平面的多个流量速率进行及时调控，将基于带内遥测信息生成精准奖励值，并将经验上传至主机侧的模型训练平面，使智能拥塞控制模型能够自进化和自学习，从而具备更强的泛化能力与场景适配效果，不断提升智算中心网络流量传输效率与可靠性。

4) DPU−主机传输通道。该传输信道负责完成主机侧与DPU拥塞控制智能体之间的信息交互，DPU利用该通道将经验信息发送至主机侧模型训练平面，使智能拥塞控制模型能完成优化更新。同时主机侧模型轻量化平面可将迭代更新并提炼后的梯度增强决策树拥塞控制模型通过该通道发送至DPU拥塞控制智能体。

5) 梯度增强决策树拥塞控制模型。深度强化学习模型能够根据环境状态及策略函数生成最佳动作。由于智算中心网络具备严格的时间尺度，复杂的深度强化学习模型计算量大导致推理时间长，因此需要智能体快速完成决策。本文通过结合模型压缩和知识蒸馏的方式将深度强化学习模型提炼成梯度增强决策树模型，从而提升模型的适配性和推理速度。

数据处理单元赋能的智算中心拥塞控制机制工作流程如下。DPU拥塞控制智能体以监控时间段（MTP, monitoring time period）的粒度收集数据包级别统计信息并将其作为输入参数利用梯度增强决策树拥塞控制模型自适应调整数据平面子流发送速率，并根据带内遥测结果输出智能体动作与环境之间的互动信息（即经验信息），将经验信息发送至主机侧模型训练平面。主机侧模型训练平面利用智能体提供的经验样本和全局状态信息对深度神经网络模型进行迭代训练，并定期将优化好的模型转发至模型轻量化平面对深度神经网络进行压缩处理，最终主机侧将提炼得到的梯度增强决策树拥塞控制模型发送至DPU控制平面，以持续提升拥塞控制算法的效果、泛化能力以及场景匹配度。

2 多智能体强化学习拥塞控制算法设计

2.1 目标分析

智算中心网络拥塞控制可转换为多智能体决策问题，假设智算中心网络中包含n个计算节点，则需要n个相应的智能体协同决策调整其数据流的发送速率，拥塞控制的主要目标是优化以下指标，其中↑ / ↓分别表示越高/越低越好。

1) 有效吞吐率（↑）。单位时间内实际有效传输的有效数据量与网络链路带宽的比值。

2) 数据包时延（↓）。智算中心网络中数据包从源端传输到目的地所需的时间。

3) 丢包率（↓）。由拥塞导致的数据包数量占数据流完成时传输的数据包数量的百分比。

4) 公平性（↑）。共享网络链路的数据流之间带宽占用率相近性的度量。

然而这些目标之间可能存在矛盾性，一方面，提高链路带宽利用率可以有效利用网络资源，提高网络整体性能。但是，当链路带宽利用率过高时，网络中的数据包可能会因为网络资源不足而产生大量排队现象，导致数据包的传输时延增加。另一方面，降低链路带宽利用率可以降低数据包的传输时延，提高网络的响应速度。但是，过低的链路带宽利用率可能会导致网络资源浪费，从而降低网络整体性能。因此，拥塞控制多目标方案呈现出帕累托前沿现象，即最优化一个目标可能造成另一个目标次优。为此，本文构建了综合考虑全局目标的奖励函数，以达到公平性、有效吞吐率、丢包率和数据包时延各方面指标的总体最优。

2.2 问题建模

智算中心网络具备可集中控制的特征，这使得在整个网络体系中部署强化学习智能体，同时智能体之间共享有效信息针对共同的全局目标进行协同优化成为可能。智算中心拥塞控制过程中速率调整动作生成面临的挑战是严格的时间尺度以及复杂动

态的网络环境，因此需要性能高、推理速度快、收敛时间短和适用连续动作空间的机器学习算法。在智算中心网络多流场景中，本文首先将拥塞控制问题转换为协作式多智能体强化学习问题，其中每个智能体分别对其所在的数据发送节点进行控制，与网络中所有智能体进行协作以实现全局目标，并利用全局状态信息通过多智能体确定性策略梯度函数对拥塞控制模型进行迭代训练优化，实现模型自学习、自进化和自适应。首先，利用部分可观察马尔可夫博弈过程[24]对智算中心网络拥塞控制问题进行抽象描述，包含以下部分：1) 该博弈过程的参与方由智算中心网络中负责拥塞控制的异步交互智能体组成；2) 智能体之间共享相同的策略和奖励目标。对于第i个数据流的每个时间步t，将全局状态信息定义为所有活动数据流的局部状态集合$(s_t^1,\cdots,s_t^n)$以及网络状态信息，智能体从带内遥测信息中获取数据流的本地环境状态s_t，并根据策略函数$\pi(a|s)$生成相应动作a_t，同时智能体在采取动作a_t后，数据流状态到达s_{t+1}，并通过动作价值函数生成的奖励信息对策略函数进行优化更新。深度强化学习的目标是迭代优化策略函数$\pi(a|s)$使得在有限的训练时间内奖励期望值到达最高。奖励期望值如式(1)所示。

$$\mathcal{J}=\mathbb{E}_{\left((s^1,a^1),\cdots,(s^T,a^T)\right)\sim p^{\pi}}\left(\sum_{t=0}^{T}\gamma^t r_t\right) \tag{1}$$

其中，T为深度强化学习的结束时刻，a^t和s^t分别为t时刻的动作和状态，γ^t为t时刻深度强化学习奖励值的折扣因子，p^{π}为所有智能体遵循策略函数π的轨迹（即动作a与状态s的组合）分布。

2.3　状态空间

状态空间包括全局状态空间与本地状态空间，分别供深度强化学习模型完成集中式训练优化和DPU拥塞控制智能体进行分布式快速决策。在训练过程中，DPU将带内遥测数据包组装成全局状态空间并发送至主机侧模型训练平面对深度强化学习模型进行协同更新。在推理过程中，DPU拥塞控制智能体仅根据自身数据流信息生成动作，实现高效拥塞控制决策。带内遥测获取的数据流统计结果包含有效吞吐量thr、时延lat、丢包率loss、尚未确认的字节数$\mathrm{pkt}_{\mathrm{flight}}$和数据流速率$P_{\mathrm{rate}}$，内嵌DPU的智能体在决策过程中使用以上特征信息构成本地状态空间。

全局状态空间则通过对所有本地数据流统计向量进行聚合以减少模型训练时的输入特征维度，从而实现更为高效的训练。综合利用最小、最大和平均指标对不同数据流的传输状态进行评估，同时收集网络链路和节点信息，以更好地对网络全局状态进行表征，最终将全局状态信息应用于智能拥塞控制算法的训练阶段。全局状态空间参数及含义如表1所示。

表1　　全局状态空间参数及含义

参数	含义
ovr_thr	当前所有活动数据流的总体有效吞吐量
min_thr	当前所有活动数据流的最小有效吞吐量
max_thr	当前所有活动数据流的最大有效吞吐量
avg_lat	当前所有活动数据流的平均时延
min_cwnd	当前所有活动数据流的最小拥塞窗口大小
max_cwnd	当前所有活动数据流的最大拥塞窗口大小
avg_cwnd	当前所有活动数据流的平均拥塞窗口大小
loss_radtio	当前所有活动数据流的平均丢包率
num_flow	当前所有活动数据流的条数
d_0	网络链路端到端基础时延
buf	网络设备缓冲区大小
c	网络链路带宽

2.4　动作空间

采用线性增减的方式进行流速快速调整以适应状态多变的智算中心网络，a_t^j为时刻t智能体对子数据流flow_j的拥塞窗口大小cwnd_t的调节动作，如式(2)所示。

$$\mathrm{cwnd}_{t+1}=\begin{cases}\mathrm{cwnd}_t\left(1+\alpha a_t^j\right), & a_t^j\geqslant 0\\ \dfrac{\mathrm{cwnd}_t}{1-\alpha a_t^j}, & a_t^j<0\end{cases} \tag{2}$$

其中，$a_t^j\in(-0.5,0.5)$，α为比例系数，以避免动作空间过大导致深度强化学习算法出现过拟合现象，同时数据处理单元根据调整后的数据流拥塞窗口大小cwnd_{t+1}对其数据流速率P_{rate}进行更新，即$P_{\mathrm{rate}}=\dfrac{\mathrm{cwnd}_{t+1}}{\mathrm{RTT}_{\mathrm{avg}}}$，$\mathrm{RTT}_{\mathrm{avg}}$为当前数据流统计的平均RTT。

2.5　奖励函数

本文将奖励函数R定义为综合反映拥塞控制子目标的线性组合函数，包括传输效率、稳定性和公平性。假设网络中活动的数据流条数为n，则根据

带内遥测数据包获取的数据流信息按以下方式计算总体奖励值。吞吐量指标R_{thr}为当前有效吞吐量与链路带宽c的比值，丢包率指标R_{loss}为当前各个数据流的丢包量与有效吞吐量的平均值，具体如式(3)所示。

$$R_{\text{thr}}=\frac{\sum_{i=1}^{n}\text{thr}_i}{c},R_{\text{loss}}=\frac{1}{n}\sum_{i=1}^{n}\frac{\text{loss}_i}{\text{thr}_i} \tag{3}$$

其中，thr_i为第i条数据流的有效吞吐量，c为网络链路带宽，loss_i为第i条数据流的丢包量。对于时延指标R_{lat}，可忽略由于并发数据流同时提高发送速率引起的较小排队时延增加，以实现网络链路带宽的高效利用，因此时延指标如式(4)所示。

$$R_{\text{lat}}=\left[(1+\beta)d_0-\frac{\sum_{i=1}^{n}\text{lat}_i}{n}\right]_{+}P_{\text{rate\_max}} \tag{4}$$

其中，d_0为基础时延，$[]_+$运算符代表结果大于0取原值，小于0取0，β为比例系数，参照Aurora算法框架设置为0.1，$P_{\text{rate_max}}$为当前所有活动数据流的最大发送速率，当因提速引发的时延增加量低于(1+β)d_0时，R_{lat}为0。同时，为了保障网络数据传输的公平性和稳定性，本文利用多个数据流平均有效吞吐量之间的标准差评估公平性指标R_{fair}，并根据当前所有活动数据流的实时吞吐量标准差评估稳定性指标R_{stab}，具体如式(5)和式(6)所示。

$$R_{\text{fair}}=-\sqrt{\frac{\sum_{i=1}^{n}(\text{avg\_thr}_i-\frac{1}{n}\sum_{i=1}^{n}\text{avg\_thr}_i)^2}{n}} \tag{5}$$

$$R_{\text{stab}}=-\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{\sum_{j=0}^{w-1}\left(\text{thr}_{i,T-j}-\text{avg\_thr}_i\right)^2}{w\text{avg\_thr}_i^2}} \tag{6}$$

其中，w为历史观测时间取样窗口数量，T为当前时间窗口，$\text{thr}_{i,t}$为第i个数据流在第t个历史观测窗口的有效吞吐量，avg_thr_i为第i个数据流历史观测到的平均有效吞吐量，如式(7)所示。

$$\text{avg\_thr}_i=\frac{1}{w}\sum_{j=0}^{w-1}\text{thr}_{i,t-j} \tag{7}$$

当R_{fair}和R_{stab}为0时，网络中数据流达到最佳公平性和稳定性，通过将以上指标进行组合构建深度强化学习奖励函数R，如式(8)所示。

$$R=c_0R_{\text{thr}}-c_1R_{\text{lat}}-c_2R_{\text{loss}}-c_3R_{\text{fair}}-c_4R_{\text{stab}} \tag{8}$$

本文对奖励函数的值域进行限制以避免陷入局部最优解，使其在每个观测时间窗口处于(−0.1,0.1)的范围，通过构建综合考虑全局优化目标的奖励函数，同时对c_0、c_1、c_2、c_3和c_4这些系数进行调整以实现子优化目标之间的权衡，使智能拥塞控制算法在训练过程中对能实现高吞吐量、公平和稳定网络的动作进行探索。

2.6　多智能体深度强化学习模型训练方法

强化学习算法利用策略函数$\pi(a_t|s_t)$输出在状态s_t下执行动作a_t的概率，神经网络理论上可以拟合任何任意函数，为此本文利用深度神经网络完成对策略函数的拟合，所设计的深度神经网络架构开始由3个输入全连接层（输入→128→64→132）组成，然后是一个长短期记忆（LSTM, long short-term memory）网络层（32→32），最后是2个输出全连接层（32→16→11），输入是当前时刻t第j个数据流的本地状态s_t^j。LSTM层使策略函数能够在进行决策与更新时充分考虑当前数据流的本地历史特征信息。多智能体深度强化学习神经网络训练过程存在非平稳特征，最终收敛奖励取决于所有数据流之间的交互作用，使动作的潜在回报估计变得不准确，传统的Q学习收敛理论在多智能体训练环境中不再适用[25]。为此，本文以多智能体确定性策略梯度算法[26]为基础，并利用演员（actor）−评论家（critic）[27]框架对深度强化学习模型进行持续训练，完成对智能体策略函数的精准更新。多智能体深度强化学习模型训练框架如图3所示。

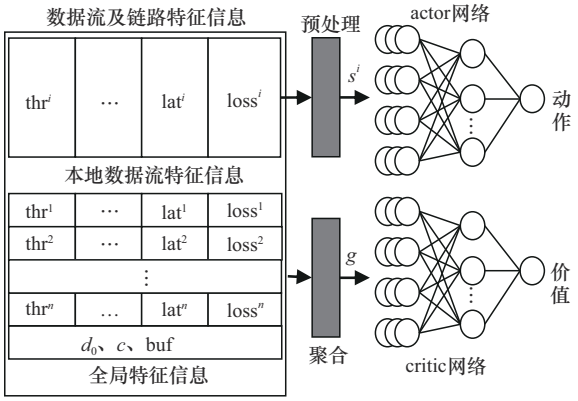


图3　多智能体深度强化学习模型训练框架

该框架由actor网络和critic网络两部分组成，其中智能体的actor网络根据本地数据流状态利用

策略函数π_θ生成动作，critic网络则使用动作−价值函数ω根据全局特征信息输出智能体当前选择动作的预期价值，如式(9)所示。

$$Q_\omega^{\pi_\theta}(g,s,a)=\mathbb{E}_{s_t,a_t\sim p^{\pi_\theta}}\left[\sum_{t=0}^{T}\gamma^t r_t|g,a,s\right] \tag{9}$$

即智能体在时间步t时在全局状态空间g下执行动作a的预期未来奖励回报。在具备足够的状态特征信息时，critic网络可为actor网络生成精准的动作价值估计，actor网络只需根据本地状态空间生成最优策略，并根据critic网络反馈的动作价值迭代更新策略函数。训练目标是使critic网络能根据当前状态和动作更精准地预测奖励，并使actor网络生成的动作更接近预期最大奖励。这2个网络均可由上述深度神经网络拟合，智算中心网络中所有智能体之间共用同类模型。本文利用多智能体确定性梯度策略计算目标函数梯度对actor网络参数进行更新，如式(10)所示。

$$\nabla_\theta\mathcal{J}(\theta)=\mathbb{E}_{s,a\sim p^{\pi_\theta}}\left[\nabla_\theta\log\pi_\theta(a|s)Q_\omega^{\pi_\theta}(g,s,a)\right] \tag{10}$$

其中，$Q_\omega^{\pi_\theta}(g,s,a)$由critic网络计算得出，拥塞控制智能体使用梯度$\nabla_\theta\mathcal{J}(\theta)$对actor网络参数进行更新，使动作策略函数向奖励增加方向优化。同时利用标准时间差分（TD, temporal-difference）方法[28]对critic网络参数进行更新，使其能做出更精准的动作价值评估，最小化目标如式(11)所示。

$$\mathcal{L}(\omega)=\mathbb{E}_{s,a,r,s'}\left[(Q_\omega^{\pi_\theta}(g,s,a)-r+\gamma Q_\omega^{\pi_\theta}(g',s',a')|_{a'=\pi_\theta(s')})^2\right] \tag{11}$$

其中，a、g和s分别表示当前时间步的动作、全局状态和本地状态，a'、g'和s'分别表示执行当前动作后下一时间步的预测动作、全局状态和本地状态，r表示当前时间步奖励值。式(11)衡量了预测动作−价值与实际奖励值之间的相对差异，通过使用梯度下降最小化$\mathcal{L}(\omega)$，critic网络能为actor网络提供更精准的动作−价值估计，使actor网络向正确的方向优化。actor网络和critic网络均利用采样经验进行更新。在训练过程中，通过接收多元组经验(g,s,a,g',s',r)并计算$\mathcal{J}(\theta)$和$\mathcal{L}(\omega)$的梯度对actor网络和critic网络参数进行更新，拥塞控制多智能体训练算法如算法1所示。

算法1　拥塞控制多智能体训练算法

输入　学习率α和η，每一次模型权重更新时所使用的样本数量（批大小）B_{sample}，训练回合数T

输出　训练完成的模型参数θ和ω

1) 初始化actor网络π_θ和critic网络$Q_\omega^{\pi_\theta}$

2) for t=0:1:T−1

3)　从环境中采取数量为B_{sample}的经验样本(g,s,a,g',s',r)

4)　计算actor网络的梯度：

$$\nabla_\theta\mathcal{J}(\theta)=\mathbb{E}_{s,a\sim p^{\pi_\theta}}\left[\nabla_\theta\log\pi_\theta(a|s)Q_\omega^{\pi_\theta}(g,s,a)\right]$$

5)　计算critic网络的梯度：

$$\mathcal{L}(\omega)=\mathbb{E}_{s,a,r,s'}\left[\left(Q_\omega^{\pi_\theta}(g,s,a)-r+\gamma Q_\omega^{\pi_\theta}(g',s',a')|_{a'=\pi_\theta(s')}\right)^2\right]$$

6)　更新actor网络π_θ和critic网络$Q_\omega^{\pi_\theta}$：

$$\theta\leftarrow\theta+\alpha\nabla_\theta\mathcal{J}(\theta),\omega\leftarrow\omega-\eta\nabla_\omega\mathcal{L}(\omega)$$

7) end for

3　拥塞控制模型与DPU的融合部署方案

3.1　部署限制性分析

目前基于强化学习的拥塞控制算法计算量庞大，且对硬件资源要求极高，因此难以直接部署在资源受限的DPU上。同时，RDMA网络RTT通常在10 μs左右，而上述神经网络结构计算量极大导致决策时间远超上限。目前主流的RDMA网络拥塞控制策略（如DCQCN和HPCC）的决策时间通常在2 μs内，因此研究目标是在智能拥塞控制策略与DPU有机融合的基础上，保证决策时间在2 μs左右。DPU引入了可编程引擎对数据传输平面进行高效流速控制，但是DPU的片上内存空间有限，同时存在浮点数运算能力不足的缺陷，且尚未全面支持深度学习激活函数库。上述支撑拥塞控制策略的原始神经网络所需浮点计算量已超过40 000次，导致DPU难以在严格的决策时间限制下完成推理，同时该神经网络有上万个权重参数值存储在内存中，已远超DPU资源上限。综上所述，原始神经网络架构难以在DPU上完成部署，下文将具体阐述如何对神经网络进行压缩与知识蒸馏，使其同时满足决策时间与硬件资源开销的要求。

3.2　融合剪枝和量化技术的神经网络压缩方法

现有神经网络压缩方法的剪枝掩码和量化器在微调过程中通常需要手动配置或者迭代优化，造成压缩过程长、计算量庞大的问题，为此，本文提出

了一次性剪枝-量化方法对深度神经网络进行高效压缩，压缩模块在微调过程中始终保持固定。对于训练完成的模型，首先构建统一剪枝误差分析机制，用于计算每层的剪枝比例并生成相应的分层剪枝掩码，完成对非关键冗余神经网络结构快速删减。然后，构建分层划分的通道量化器，将通道级权重转化为位数更低的低精度整数，从而在最大化保留模型精度的同时减少内存占用和计算量，该方法的流程如图4和算法2所示。

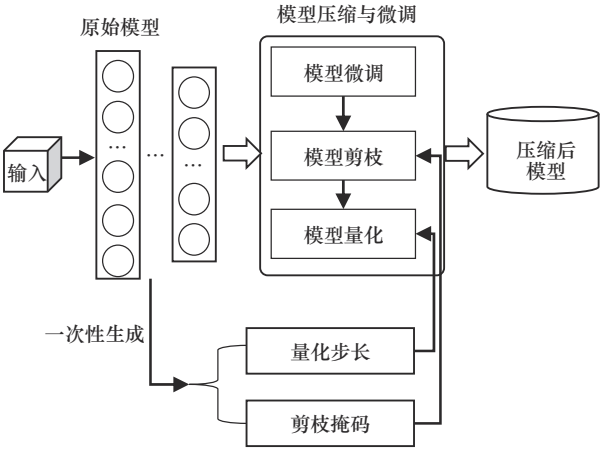


图4 融合剪枝和量化技术的神经网络压缩方法流程

算法2 融合量化和剪枝技术的神经网络压缩方法

输入 L层FP32原始神经网络，目标剪枝率p^*，目标量化字节位宽B，批大小N_b，最大回合数N_e，经验样本池T_S

输出 微调完成后的压缩模型

1) 计算出模型各层的剪枝掩码$\{M_i\}_{i=1}^{L}$

2) 计算出统一通道量化器$\{\Delta_i\}_{i=1}^{L}$

3) for 1,2,⋯,N_e do

4) 从经验样本池T_S中采样批大小为N_b的经验样本

5) 利用经验样本采用增量学习的方式对模型进行微调

6) end for

7) 利用层级剪枝掩码($\{M_i\}_{i=1}^{L}$)和统一通道量化器($\{\Delta_i\}_{i=1}^{L}$)对微调后的模型进行压缩

令$\mathcal{W}=\{W_i\}_{i=1}^{L}$为层数为$L$的神经网络FP32（float32，32位浮点数）权重张量，其中W_i为第i层的权重张量。为了简化表述，令W_{ij}（j=1,2,⋯,N_i）为W_i的第j个元素，W_{ijk}（k=1,2,⋯,N_{ij}）为神经网络第i层第j个通道的第k个元素，N_i为W_i中的权重个数，N_{ij}为神经网络第i层第j个通道的权重个数。本文所设计的神经网络的每一层权重都满足零附近的对称分布（即拉普拉斯分布[29]）概率，定义神经网络第i层权重的概率密度函数为$f_i(x)$。该方案在模型微调过程中的剪枝掩码（$\{M_i\}_{i=1}^{L}$）和统一通道量化器（$\{\Delta_i\}_{i=1}^{L}$）保持固定，从而加速了模型压缩效率。对于给定的预训练完成模型，剪枝-量化过程可用式$\hat{W}=M\circ\left(\Delta\left\lfloor\frac{W}{\Delta}\right\rfloor\right)$表示，其中$W$为预训练模型的某一个权重，$\hat{W}$为压缩后的模型权重，$\circ$为Hadamard乘积，$M$为剪枝掩码通用表示符号，$\Delta\left\lfloor\frac{W}{\Delta}\right\rfloor$为量化步长为$\Delta$的权重量化操作。为使拥塞控制智能模型能持续优化以适应动态变化的智算中心网络，在采用增量学习[30]的方式对模型进行微调后再进行量化和剪枝，同时保留当前最新原始神经网络以供模型后续优化。剪枝掩码($\{M_i\}_{i=1}^{L}$)和通道量化器($\{\Delta_i\}_{i=1}^{L}$)的计算方式在3.2.1节和3.2.2节详细描述。

3.2.1 统一层级权重参数裁剪方法

本节构建了一个统一方法对给定预训练神经网络的每一层权重参数张量W_i进行裁剪，模型剪枝问题的关键是确定可以删除的权重[31]，因此可将模型剪枝问题转换为寻找所有层的剪枝比$\{p_i\}_{i=1}^{L}$，其中p_i为第i层被删除的小幅度（即零附近的小绝对值）权重百分比。具体来说，其删除了第i层对称范围$[-\beta_i,\beta_i]$的所有权重参数，β_i为正值标量，模型剪枝率p可由式(12)计算。

$$p=\frac{1}{N}\sum_{i=1}^{L}\int_{-\beta_i}^{\beta_i}N_i f_i(x)\mathrm{d}x=\frac{2}{N}\sum_{i=1}^{L}\int_{0}^{\beta_i}N_i f_i(x)\mathrm{d}x \quad (12)$$

其中，N为模型所有权重参数个数，N_i为第i层的权重参数个数，$f_i(x)$为神经网络第i层权重参数概率密度函数。因此，第i层的剪枝误差如式(13)所示。

$$\mathcal{L}_i^{\beta}=\sum_{j=1}^{N_i}\left(W_{ij}\right)^2\Big|_{|W_{ij}|\leqslant\beta_i}=2\int_{0}^{\beta_i}N_i x^2 f_i(x)\mathrm{d}x \quad (13)$$

其中，W_{ij}为第i层的第j个权重参数，$f_i(x)$为神经网络第i层权重参数概率密度函数，β_i为第i层的剪枝范围。模型剪枝的目标是最小化权重参数删除引发的误差，使剪枝后的模型性能与原始模型性能尽可能保持一致，为此定义目标函数如式(14)所示。

$$\beta_1^*,\cdots,\beta_L^* = \arg\min_{\beta_1,\beta_2,\cdots,\beta_L} \frac{1}{N}\sum_{i=1}^{L}\mathcal{L}_i^{\beta} = \arg\min_{\beta_1,\beta_2,\cdots,\beta_L} \frac{2}{N}\sum_{i=1}^{L}\int_0^{\beta_i} N_i x^2 f_i(x)\mathrm{d}x \quad (14)$$

对于给定目标剪枝率p^*，可以通过拉格朗日乘子法求解式(14)的最小化问题，得到式(15)。

$$\mathcal{L}\left(\beta_1,\cdots,\beta_L,\lambda\right) = \sum_{i=1}^{L}\frac{2}{N}\int_0^{\beta_i} N_i x^2 f_i(x)\mathrm{d}x - \lambda\left(\frac{2}{N}\sum_{i=1}^{L}\int_0^{\beta_i} N_i f_i(x)\mathrm{d}x - p^*\right) \quad (15)$$

其中，λ为拉格朗日乘子，当拉格朗日函数$\mathcal{L}\left(\beta_1,\cdots,\beta_L,\lambda\right)$对$\beta_i$的偏导数为0时，可得到$\beta_i$的值即$\beta_i^* = \beta_2^* = \cdots = \beta_L^* = \sqrt{\lambda}$。同理，可以将拉格朗日函数$\mathcal{L}\left(\beta_1,\cdots,\beta_L,\lambda\right)$对$\lambda$的偏导数设为0，计算得到$\lambda$，如式(16)所示。

$$\frac{\partial\mathcal{L}\left(\beta_1,\cdots,\beta_L,\lambda\right)}{\partial\lambda} = \frac{2}{N}\sum_{i=1}^{L}\int_0^{\sqrt{\lambda}} N_i f_i(x)\mathrm{d}x - p^* = 0 \quad (16)$$

本文所设计神经网络第i层权重参数概率密度函数$f_i(x)$可被定义为拉普拉斯概率密度函数$\frac{\mathrm{e}^{\frac{-|x|}{\tau_i}}}{2\tau_i}$，其中$\tau_i$为第$i$层的一个标量权重参数，因此其概率分布函数可以表示为

$$F(x) = \int_0^x f_i(y)\mathrm{d}y = \int_0^x \frac{1}{2\tau_i}\mathrm{e}^{\frac{-|y|}{\tau_i}}\mathrm{d}y = -\frac{1}{2}\mathrm{e}^{-\frac{y}{\tau_i}}\Big|_0^x \quad (17)$$

则式(16)可转换为

$$\frac{\partial\mathcal{L}\left(\beta_1,\cdots,\beta_L,\lambda\right)}{\partial\lambda} \approx \frac{1}{N}\sum_{i=1}^{L} N_i\left(1-\mathrm{e}^{-\frac{\sqrt{\lambda}}{\tau_i}}\right) - p^* = 0 \quad (18)$$

利用 Levenberg-Marquardt[32]算法可对每层的权重参数概率密度分布函数$F(x)$进行拟合，从而推导出层级标量参数τ_i，同时利用 Newton-Raphson[33]算法求解式(18)中的拉格朗日乘子λ，并代入$\beta_i^* = \beta_2^* = \cdots = \beta_L^* = \sqrt{\lambda}$得到$\{\beta_i\}_{i=1}^{L}$，利用$p_i = \int_{-\beta_i}^{\beta_i} N_i f_i(x)\mathrm{d}x$计算出每一层剪枝比，同时通过幅度阈值法得到剪枝掩码$M_i \in \{0,1\}^{|W_i|}$。通过$\hat{W}_i = M_i \circ W_i$删除每一层非关键权重，剩余未修剪权重为稀疏张量，可以通过利用差异化索引和较低的位数进行编码并存储，从而降低模型的内存占用率和计算量，提升后续模型的知识蒸馏效率，完成拥塞控制模型的高效迭代优化。

3.2.2　统一通道级权重参数量化方法

与现有通道级权重参数量化方法通道之间使用差异化量化步长不同，本文所提统一通道级权重参数量化方法使模型单个层的所有通道共享公共量化步长Δ，令k_{ij}为第i层第j个通道的量化区间数，即该通道未剪枝权重参数所需的量化中心点数量，令α_{ij}为第i层第j个通道的权重参数最大值（即$W_{ijk} \in [-\alpha_{ij},\alpha_{ij}]$），则区间$[-\alpha_{ij},\alpha_{ij}]$可划分为$k_{ij}$个相等的量化区间。因此，量化区间数$k_{ij}$可由式(19)表示。

$$k_{ij} = \left\lceil \frac{2\alpha_{ij}}{\Delta_i} \right\rceil \quad (19)$$

其中，Δ_i为第i层的公共量化步长，$\lceil\cdot\rceil$为取整运算符，$\alpha_{ij} = \max\{|W_{ijk}||k = 1,2,\cdots,\overline{N}_{ij}\}$，$|\cdot|$为绝对值运算符，$W_{ijk}$为神经网络模型第$i$层第$j$个通道的第$k$个权重参数，$\overline{N}_{ij}$为神经网络模型第$i$层第$j$个通道的未裁剪权重参数数量，则第$i$层所需的量化区间数由式(20)表示。

$$K_i = \frac{1}{\overline{N}_i}\sum_{j=1}^{C_i}\overline{N}_{ij}K_{ij} \quad (20)$$

其中，C_i为神经网络模型第i层的通道数量，$\overline{N}_i = \sum_{j=1}^{C_i}\overline{N}_{ij}$为神经网络模型第$i$层未裁剪权重参数数量，因此整个神经网络模型的量化区间数量如式(21)所示。

$$\frac{1}{\overline{N}}\sum_{i=1}^{L}K_i N_i\int_{\beta_i}^{+\infty} f_i(x)\mathrm{d}x = \frac{1}{\overline{N}}\sum_{i=1}^{L}\sum_{j=1}^{C_i}\overline{N}_{ij}K_{ij} = 2^B \quad (21)$$

其中，B为存储模型所有未裁剪权重参数所需的目标最小位宽，$\overline{N} = \sum_{i=1}^{L}\overline{N}_i$为模型中所有未裁剪权重参数数量。利用给定公共量化步长Δ_i，相应层的每个权重参数可以被均匀量化至相应的离散级别，神经网络模型同一层通道共享量化函数为$Q_i(x)$，如式(22)所示。

$$Q_i(x) = \mathrm{sgn}(x)\Delta_i\left\lceil\frac{|x|}{\Delta_i}\right\rceil \quad (22)$$

其中，sgn(x)为符号函数，$\lceil\cdot\rceil$为取整运算符。因此由量化引发的模型均方误差可以表示为

$$\mathcal{L}_q = \sum_{i=1}^{L}\frac{1}{\overline{N}_i}\sum_{j=1}^{C_i}\sum_{k=1}^{S_j}N_{ij}\left(W_{ijk} - Q_i\left(W_{ijk}\right)\right)^2 \quad (23)$$

其中，C_i为神经网络模型第i层的通道数量，S_j为神经网络模型第i层第j个权重参数数量，W_{ijk}为模型第i层第j个通道的第k个权重参数。由于$f_i(x)$为对称分布函数，因此可以将式(23)简化为式(24)。

$$\mathcal{L}_q = 2\sum_{i=1}^{L}\frac{1}{\bar{N}_i}\int_{\beta_i}^{+\infty} N_i f_i(x)\left(x - Q_i(x)\right)^2 \mathrm{d}x \approx \sum_{i=1}^{L}\frac{\Delta_i^2}{12} \tag{24}$$

其中，$\bar{N}_i = 2N_i\int_{\beta_i}^{+\infty} f_i(x)\mathrm{d}x$为模型第$i$层所有未裁剪权重参数数量，$N_i$为原始模型第$i$层所有权重参数数量。为了在量化位数限制条件下最小化$\mathcal{L}_q$，同样采用拉格朗日乘子法求解在式(21)约束下的最优量化步长，如式(25)所示。

$$\mathcal{L}\left(\Delta_1,\cdots,\Delta_L,\lambda\right) = \sum_{i=1}^{L}\frac{1}{12}\Delta_i^2 + \lambda\left(\frac{1}{\bar{N}}\sum_{i=1}^{L}\sum_{j=1}^{C_i}\bar{N}_{ij}\frac{2\alpha_{ij}}{\Delta_i} - 2^B\right) \tag{25}$$

其中，C_i为神经网络模型第i层的通道数量，$\bar{N}_{ij}$为模型第i层第j个通道的未裁剪权重参数数量，α_{ij}为模型第i层第j个通道的权重参数最大值，类似3.2.1节中关于剪枝比例的拉格朗日推导，得到Δ_i和λ分别如式(26)和式(27)所示。

$$\Delta_i = \sqrt[3]{\frac{12\lambda\sum_{j=1}^{C_i}\bar{N}_{ij}\alpha_{ij}}{\bar{N}}} \tag{26}$$

$$\lambda = \left(\frac{1}{2^{B-1}\bar{N}}\sum_{i=1}^{L}\sqrt[3]{\frac{\bar{N}}{12}}\frac{\sum_{j=1}^{C_i}\bar{N}_{ij}\alpha_{ij}}{\sqrt[3]{\sum_{j=1}^{C_i}\bar{N}_{ij}\alpha_{ij}}}\right)^3 \tag{27}$$

通过联合式(26)和式(27)，可以得到统一通道量化器（$\{\Delta_i\}_{i=1}^{L}$）并对深度神经网络所有权重参数进行量化操作，从而降低模型的计算复杂度和拟合难度。为了验证本文所提压缩方法的优势，以文献[20]中的神经网络模型为基准模型，分别利用目前主流算法包括剪枝方法（CoFi）[34]、量化方法（VecQ）[35]、剪枝-量化联合方法（OBC）[36]以及本文方法进行压缩操作，并进行压缩效果分析。最终，以文献[20]的测试数据集为输入得到输出差异度，评估不同压缩方法对模型精度的损失值影响，实验结果如表2所示。

表2　压缩方法效果对比

压缩方法	输出差异度	剪枝比例	量化位数/位	加速效果/倍
CoFi	6.14%	95.31%	32	23.52
VecQ	3.49%	—	2	16
OBC	2.31%	60.12%	8	15.12
本文方法	4.12%	65.01%	4	30.15

仿真结果表明，本文方法较目前主流压缩方法在保证尽量不损失模型精度的同时，实现模型压缩效果的最大化。尽管通过剪枝和量化得到的二值轻量化深度神经网络可在资源受限的数据处理单元内部署，但其依旧存在计算量大导致推理时间超过RTT的缺陷。为此需要进一步利用知识蒸馏的方式将深度神经网络模型转化为计算量极小的二叉决策树模型，从而进一步压缩拥塞控制决策时间。

3.3　梯度增强树驱动的神经网络模型知识蒸馏方法

不同于传统直接对原始深度神经网络进行知识蒸馏的方式，本文在得到量化和剪枝后的二值轻量化深度神经网络基础上进一步将其转换为轻量级二叉决策树模型。一方面，可以提升模型迭代优化与蒸馏速度，加强模型对智算中心网络的自适应与自进化能力；另一方面，经过剪枝与量化的模型去除了重要性低的权重，使其拟合难度降低，可有效提升模型蒸馏速度。在深度神经网络训练过程中，智能体需要与环境进行持续交互，并通过随机梯度上升/下降算法更新模型参数，从而得到一个最优的深度强化学习策略函数。当深度神经网络模型训练完成并完成量化与剪枝后，如何使一个轻量化模型在相同输入下尽可能输出与其相似的动作成为有监督学习问题，即模型知识蒸馏过程。现有的模型知识蒸馏方式[37]在智算中心网络场景下面临2个主要挑战。一是知识蒸馏方式所需的经验样本数据有限，导致其在数据量稀缺的条件下难以提炼出高精度学生模型。二是由于网络设备的计算资源、内存和所支持运算类型受限，因此对学生模型的选择具备严格限制，即只能部署基于树的模型（如决策树、随机森林等）。文献[38]提出了Metis框架，通过将深度神经网络策略转换为多个基于树的可解释轻量级控制器。然而受DPU内存限制，利用该Metis框架拟合出来的小规模决策树森林存在模型精度与性能显著下降的缺陷。

梯度增强树作为一种集成学习方法，其通过将多个弱学习器（决策树）组合起来构建一个性能强大的机器学习模型，具备确定性的特征并会对前序预测器的错误进行及时调整，因此可以使模型整体表现出较强的健壮性。本文引入梯度增强树将拥塞控制神经网络提炼成等效模型，梯度增强树通过梯度下降的方式最小化损失函数，从而构建一系列的决策树，每个新的决策树用于修正前一个决策树的残差，最终将这一系列决策树完成组合以构建强大的集成模型，过程如图5所示。

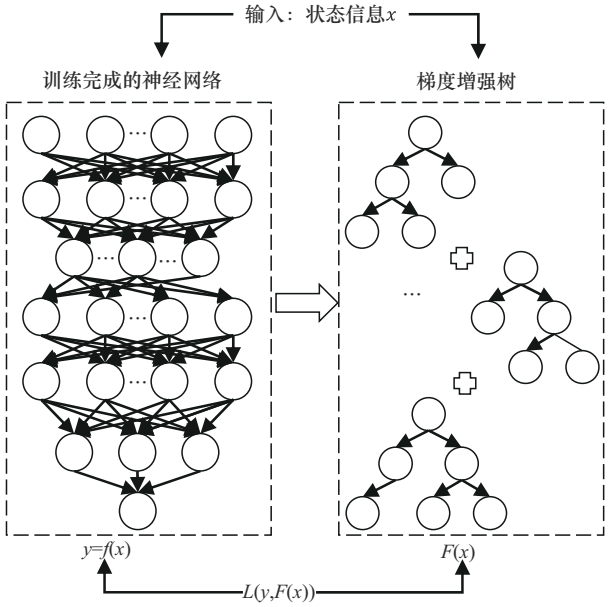


图5　梯度增强树驱动的神经网络模型知识蒸馏方法

与深度强化学习策略函数类似，模型蒸馏任务的关键点是构建一个函数$F^{*}=\mathbb{R}^{m}\rightarrow\mathbb{R}$，将所输入的状态信息映射为输出动作空间，即回归问题。为了获取模型知识蒸馏所需的训练经验样本数据集，可利用所收敛的深度强化学习策略收集多条轨迹$U=\{U_1,U_2,U_3,\cdots,U_n\}$,$U_i=(s_i,a_i)$（即历史动作$a$与历史状态$s$的组合）分布，并记录观测到的输入特征和预测动作，最终利用构建的训练经验样本数据集指导二叉梯度增强树学生模型输出与深度神经网络教师模型期望值最为接近的动作，该过程为监督学习，即最小化二叉梯度增强树训练过程的损失函数$L(y,F(x))=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-F(x_i))^2}$，其中$n$为轨迹数量，$y$为深度神经网络教师模型输出结果，$F(x)$为二叉梯度增强树预期输出结果，具体如式(28)所示。

$$\hat{F}=\arg\min_{F}\mathbb{E}\left[L(y,F(x))\right] \tag{28}$$

其中，$\hat{F}$为通过迭代监督学习训练构建的决策树预测器序列。在每一个迭代时间段t内，有$F^{t}=F^{t-1}+\alpha h^{t}$，$\alpha$为学习率（步长），本文设置该监督学习的学习率为0.02，$h^{t}:\mathbb{R}^{m}\rightarrow\mathbb{R}$为基础预测器序列，$h^{t}$的计算式如式(29)所示。

$$h^{t}=\arg\min_{h\in H}\mathbb{E}\left[L(y,F^{t-1}(x)+h(x))\right] \tag{29}$$

本文采用CatBoost[39]作为原型梯度增强树，对剪枝和量化后的拥塞控制深度神经网络进行知识蒸馏。CatBoost作为主流先进的梯度增强决策树框架，能够将基础预测器序列h^{t}拟合成多个二叉决策树，可有效提升智能拥塞控制算法的速度和准确度。模型压缩效果对比如表3所示，通过将拥塞控制策略函数从深度神经网络模型转换为二叉决策树模型的方式，可大幅降低拥塞控制算法决策时延，以满足智算中心网络数据流高速传输场景下严格的时间窗口大小要求。

表3　　模型压缩效果对比

模型种类	浮点数运算次数/次	决策时延/μs
原始深度神经网络	46 256	458
二值神经网络	1 326	17
梯度增强决策树	—	1.53

同时，本文参照文献[20]初步搭建仿真测试环境，将二值神经网络教师模型和基于梯度增强决策树的学生模型性能进行仿真对比，实验结果如表4所示。结果表明，使用模型知识蒸馏的方式所提炼得到的学生模型能够极为接近地模仿教师模型动作，从而实现表现相似的拥塞控制效果。

表4　　模型性能对比

模型种类	数据流数量/条	吞吐率	端到端时延/μs
二值神经网络	32	99.61%	5.29
	256	98.89%	6.61
	2 048	97.18%	9.25
梯度增强决策树	32	99.52%	5.32
	256	98.82%	6.67
	2 048	97.12%	9.32

4 仿真分析

DPU具备专用流速控制引擎，因此可将基于梯度增强决策树的拥塞控制算法转换为简单if-else逻辑结构并完成部署。为了验证本文算法的有效性，使用NS3仿真软件模拟构建多类场景的训练环境，将网络链路速率设置为400 Gbit/s，网络协议设置为RoCEv2，通过模拟仿真方式构建各种多对一和多对多传输场景以搭建智能拥塞控制算法训练环境，并通过NS3-AI[40]接口对智能拥塞控制算法进行迭代训练优化，采用异步的方式完成多智能体与网络环境的实时交互，利用经验回放以及增量学习机制迭代训练多智能体强化学习拥塞控制模型，最后将通过模型压缩和知识蒸馏提炼得到的二叉梯度增强决策树部署至DPU内部并在实际场景下进行性能验证，参照文献[20-21]对拥塞控制算法训练参数进行设置，并利用网格搜索方法对最优压缩参数进行探索，得到最终结果如表5所示，DPU型号为英伟达Blue Field 2。

表5 拥塞控制算法训练与压缩参数

参数	值
学习率(α, η)	0.001
历史观测取样窗口ω	5
奖励函数折扣因子γ	0.98
批大小	192
模型更新时间间隙/s	5
模型更新步长	20
动作控制系数	0.85
奖励系数c_0	0.1
奖励系数c_1	0.02
奖励系数c_2	1
奖励系数c_3	0.02
奖励系数c_4	0.01
目标剪枝率p*	0.65
目标量化位宽B	4

针对稳态环境下的算法性能验证，本文在两级Fat-Tree拓扑集群内部多种传输场景下进行了流量拥塞控制效果测试，该Fat-Tree拓扑集群包含4个机架顶部（ToR, top of the rack）交换机，每个ToR交换机与16个计算节点连接，如图6所示。针对动态场景下（如长短流测试场景）的算法性能验证，本文在包含单交换机集群和16个计算节点的单机架网络集群中进行不同拥塞控制算法的效果测试。在多对一传输、多对多传输和长短流传输场景下的测试过程中，通过持续向接收方发布64 KB的RDMA写请求来生成数据流，并对算法性能进行量化评估。

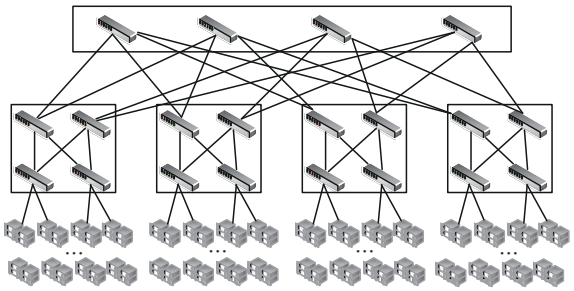


图6 Fat-Tree拓扑集群

本文将DPU赋能的拥塞控制算法（DECC）与基于规则的拥塞控制算法（如Bolt、HPCC）以及基于强化学习的算法（如ADPG、SPINE）进行比较，分别在保持网络稳态和应对网络状态突变能力方面进行对比验证。为保障网络性能稳定，本文在多对一和多对多测试场景下从时延、吞吐量、丢包率和公平性4个方面评估所提算法的优越性，利用OSU（Ohio supercomputer center university）全对全测试[41]评估本文算法在保障并行计算场景下消息传递接口（MPI, message passing interface）集合通信效率上的性能。此外，通过长短流测试对本文算法应对网络状态突变的响应能力开展评估。采用消融实验在多对多测试场景下分析本文机制的每个步骤（多智能体强化学习模型构建、模型剪枝-量化、模型蒸馏）对拥塞控制算法总性能的影响程度，并开展了本文算法在硬件资源开销方面的对比测试。

4.1 多对一数据传输场景

在多对一数据传输场景下，多个智算中心计算节点向单一汇聚节点持续发送数据，将DECC与多种拥塞控制算法进行对比，在4种不同数据流规模下，对有效吞吐率、P99端到端时延（即升序排列后排在99%位置的端到端时延）、公平性指标JAIN和丢包率进行测量，其中，有效吞吐率为数据流有效传输速率（即单位时间为实际成功传输的有效数据量）与链路带宽的比值，公平性指标JAIN如式(30)所示，实验结果如图7所示。

$$J\left(x_1,x_2,\cdots,x_n\right)=\frac{\left(\sum_{i=1}^{n}x_i\right)^2}{n\sum_{i=1}^{n}\left(x_i\right)^2} \tag{30}$$

其中，x_i为第i个计算节点的平均带宽占用量，n为节点数量。公平性指标JAIN取值范围为$\left[\frac{1}{n},1\right]$，越接近1表示带宽资源分配越公平。

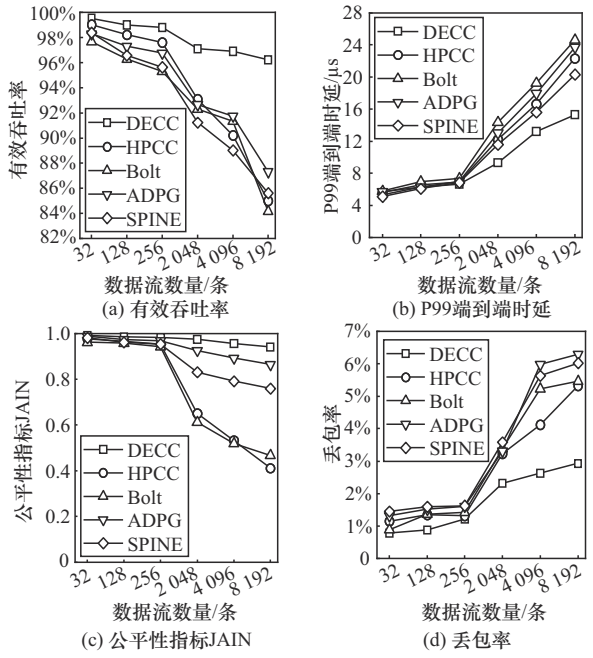


图7　多对一数据传输场景实验结果

由图7可得，当数据流数量较少时，实验所对比的拥塞控制算法在测试指标方面上相差不大，然而当数据流数量大于2 048条时，HPCC和Bolt难以应对数据流相互影响的复杂网络环境，且缺乏数据流发送方间协同解决拥塞的机制，因此所做出的拥塞控制决策可能陷入局部最优解，造成拥塞死锁，引发重传现象频发，导致丢包率大幅上升并对有效吞吐率造成巨大影响。而基于机器学习的拥塞控制算法（如ADPG、SPINE）依赖于深度神经网络，存在推理时间长的缺陷，从而引发决策滞后问题，造成所做出的拥塞控制动作不再适配当前网络状态。同时现有的这两类算法缺乏在公平性指标JAIN方面的考虑，因此在所设计的关键测试指标上均劣于本文所提DECC算法。当数据流数量为8 192条时，DECC算法的有效吞吐率、P99端到端时延、公平性指标JAIN和丢包率分别为96.23%、15.32 μs、0.941和2.93%。在有效吞吐率方面，表现次优的ADPG算法为86.32%，相较DECC算法提升11.48%。在P99端到端时延方面，表现次优的SPINE算法为20.34 μs，相较DECC算法降低24.68%。在公平性指标JAIN方面，表现次优的ADPG算法为0.864，相较DECC算法提升8.9%。在丢包率方面，表现次优的HPCC算法为5.32%，相较DECC算法降低44.92%。

4.2　多对多数据传输场景

在多对多数据传输场景下，多个智算中心计算节点并行运行多个数据包广播发送进程，实现集群内节点相互通信，在测试过程中所有计算节点的数据包广播发送进程数量保持一致并逐步增加，在该场景下拥塞点分布极为广泛，如何在保障有效吞吐率的同时最大化降低传输时延成为难题，同4.1节的测试指标一致，多对多数据传输场景的实验结果如图8所示。

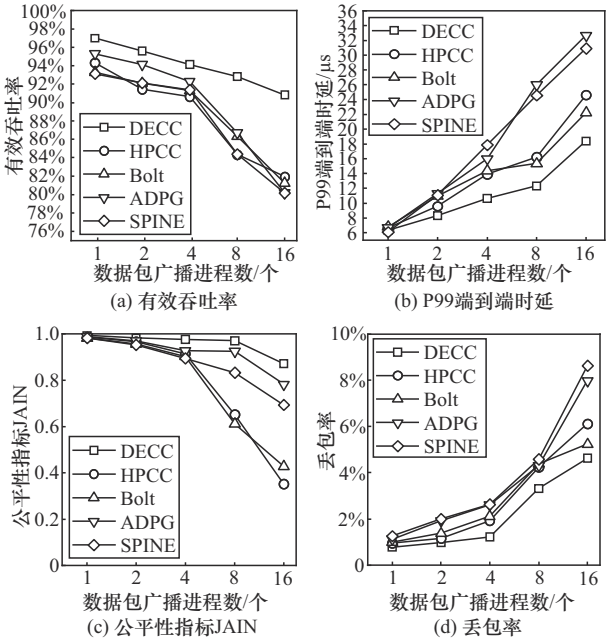


图8　多对多数据传输场景实验结果

由图8可得，当数据包广播进程数增加至8个时，所对比的4种算法在四大关键指标下均产生了大幅下降，具体原因分析如下。1) HPCC算法需保障网络中未确认的字节数始终低于阈值以避免发出PFC帧，但该阈值难以随网络状态精准变化造成大规模高并发数据传输场景下的实际丢包率以及P99端到端时延大幅上升。2) Bolt算法在数据流数量达到一定规模后会引发网络带宽的无序竞争，导致资源分配长时间排队造成有效吞吐率大幅下降以及P99端到端时延上升，使数据流之间的带宽资源占用量产生巨大差异。3) ADPG和SPINE算法需要计算量和内存占用量极大的深度神经网络支撑，当拥

塞控制决策任务所需的硬件资源过大时会造成数据处理单元过载，极大影响数据传输任务效率，导致测试指标均劣于基于规则的拥塞控制算法以及本文算法，尽管ADPG和SPINE算法均对数据流传输公平性指标JAIN进行了考虑，但依然会存在决策滞后的问题，导致公平性指标JAIN存在一定程度的下降。当数据包广播进程数为16个时，DECC算法的有效吞吐率、P99端到端时延、公平性指标JAIN和丢包率分别为90.82%、17.32 μs、0.871和4.63%。在有效吞吐率方面，表现次优的HPCC算法为81.95%，相较DECC算法提升10.82%。在P99端到端时延方面，表现次优的Bolt算法为20.94 μs，相较DECC算法降低17.31%。在公平性指标JAIN方面，表现次优的ADPG算法为0.782，相较DECC算法提升11.38%。在丢包率方面，表现次优的Bolt算法为5.23%，相较DECC算法降低11.47%。

4.3　OSU全对全传输场景

在OSU全对全传输场景中，需要测量智算中心网络进行MPI集合通信下随消息数据量大小的增加下平均消息传输完成时间的变化，初始消息数据量大小设置为32 KB，实验结果如图9所示。

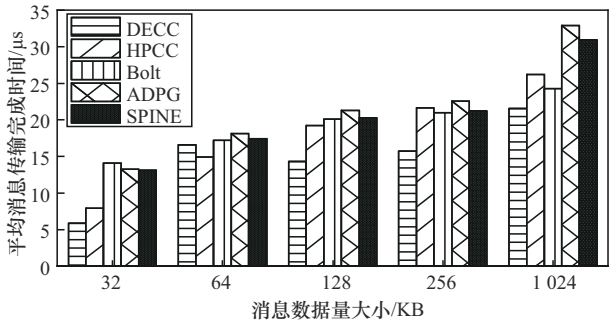


图9　OSU全对全传输场景实验结果

由图9可得，随着消息数据量大小的增加，平均消息传输完成时间也逐步增加，但DECC算法始终保持相对较低水平。当消息数据量大小增加至一定量后，智算中心网络发送拥塞节点呈爆炸式增长，HPCC和Bolt算法尽管能在短时间内迅速做出降速调整避免排队时延大幅增长，但难以保证所做决策保持最优，造成平均消息传输完成时间增加；ADPG算法存在推理时间过长的缺陷，导致所做拥塞控制动作存在滞后性，难以在短时间内消除拥塞，造成平均消息传输完成时间始终保持相对最大值；尽管SPINE算法能在短时间内做出发送速率调整动作，但依旧难以保证动作的优越性；DECC算法在部署运行阶段可采集一系列经验轨迹，并通过离线增量学习的方式对拥塞控制模型参数进行优化，最终通过上述的模型压缩方式构建新的梯度增强决策树实现持续更新。同时DECC算法具备极快的推理速度，因此能做出更为匹配网络环境的相对较优拥塞控制动作。在消息数据量大小为1 024 KB时，采用DECC算法的智算中心网络平均消息传输完成时间为21.59 μs，而表现次优的Bolt算法为24.32 μs，相较降低11.23%。

4.4　长短流数据传输场景

在长短流数据传输场景下，主要测试网络状态突变时拥塞控制算法的响应能力。在该测试场景中，长流不间断进行大规模数据传输，大量数据传输量极小的短流随机加入网络。本文在包含单交换机集群和16个计算节点的单机架网络集群中进行不同拥塞控制算法的效果测试，主要评估指标为长流的平均带宽利用率和降速所需时间。一方面，当短流加入网络后，长流必须适当降低其发送速率以保证短流能正常完成数据传输，长流相对反应时间通过其降速所需时间与RTT的比值进行评估。另一方面，当短流完成数据传输任务后，长流应快速恢复至其原有速率，因此更高的长流带宽平均利用率代表所使用拥塞控制算法有着更优的响应能力，长流相对反应时间和平均带宽利用率的实验结果分别如图10和图11所示。

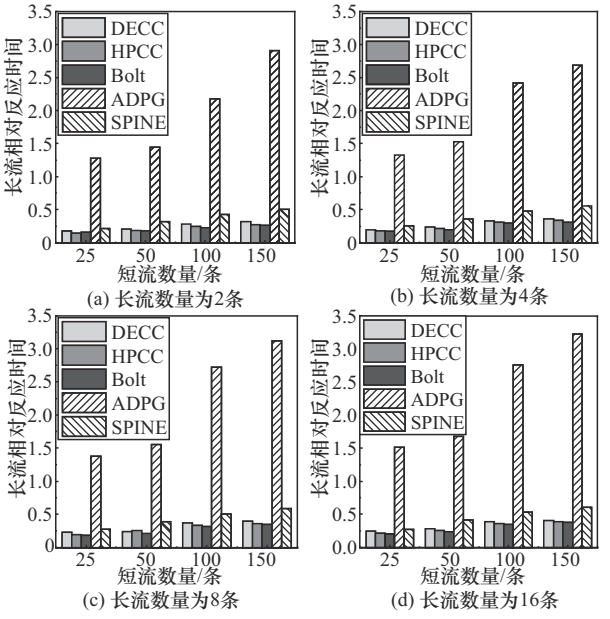


图10　长流相对反应时间实验结果

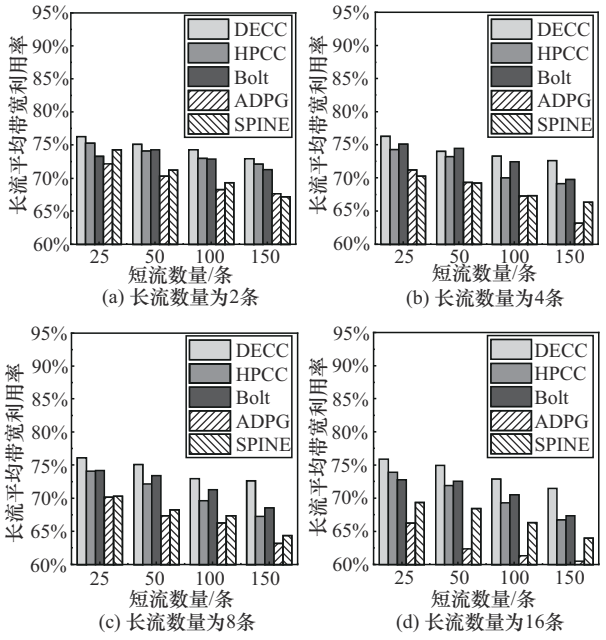


图11　长流平均带宽利用率实验结果

由图10和图11可得，在多个测试条件下，DECC算法的网络长流相对反应时间与HPCC和Bolt算法基本保持一致，并具备最高的长流平均带宽利用率；ADPG算法始终保持着极高长流相对反应时间与较低长流平均带宽利用率；尽管SPINE算法较ADPG算法的表现有着大幅提升，但依旧逊色于DECC、HPCC和Bolt算法。这是由于DECC算法具备相对较低的推理时间能够快速响应网络状态突变情况，同时其采用线性增长调速策略能实现长流传输速率快速恢复。而ADPG算法推理时间过长难以应对网络状态突变情况，SPINE算法难在短时间内将拥塞控制参数调整至最优值从而引发带宽资源浪费。在长流数量为16条、短流数量为150条时，DECC算法的长流相对反应时间和平均带宽利用率分别为0.41和71.45%，采用Bolt算法的网络长流相对反应时间和平均带宽利用率分别为0.38和67.35%，由此可得，DECC算法能快速响应，且具备更为优秀的网络恢复能力。

4.5　优化步骤对总体性能影响情况测试

本文方法包括3个主要优化步骤：利用多智能体深度强化学习生成拥塞控制神经网络（简称为神经网络）、经过剪枝和量化得到二值轻量化深度神经网络（简称为二值网络）和通过知识蒸馏提炼轻量级二叉决策树模型（简称为决策树）。为分析不同优化步骤对拥塞控制算法总体性能的影响程度，在多对多数据传输场景（数据包广播进程数为8个）中进行仿真消融实验，得到如表6所示结果。

表6　消融实验结果

优化步骤	有效吞吐率	公平性指标JAIN	P99端到端时延/μs	丢包率
HPCC	84.34%	0.651	16.23	4.23%
神经网络	72.32%	0.815	26.31	8.38%
二值网络	86.37%	0.901	20.01	5.69%
决策树	92.83%	0.956	12.32	3.32%

由表6可得，利用多智能体深度强化学习生成拥塞控制神经网络尽管较HPCC算法在公平性指标JAIN上有所提升，但由于其占用大量计算资源及存在决策滞后问题导致有效吞吐率、P99端到端时延和丢包率显著下降；经过剪枝和量化得到的二值轻量化深度神经网络对计算资源占用大幅降低，尽管决策时间依旧较长，但总体性能指标相较HPCC算法已实现提升；轻量级二叉决策树模型将拥塞控制效果实现了最大化。

4.6　硬件资源开销情况测试

为进一步验证DECC算法的轻量化压缩水平以及与DPU的融合情况，本文在如图6所示的网络拓扑上分别部署了DECC、HPCC、ADPG和Aurora算法，并测试不同算法在多对多数据传输场景下随数据包广播进程数增加对数据处理单元CPU和内存占用率的变化情况，实验结果如图12所示。

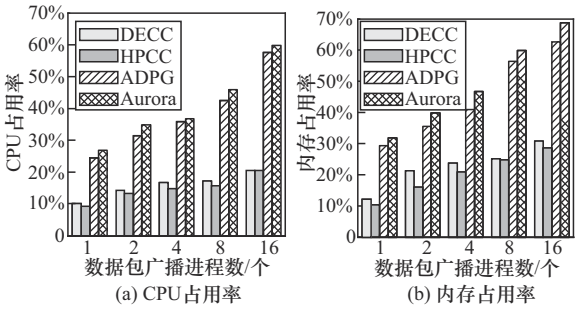


图12　硬件资源开销情况实验结果

由图12可得，随着数据包广播进程数逐渐增加，4种拥塞控制算法的CPU占用率和内存占用率均逐渐增加，ADPG和Aurora算法在数据流达到一定规模后会导致内存和计算资源爆炸式增长，DECC算法由梯度增强决策树驱动，因此计算量与所需内存相对较小，在硬件资源开销方面大体与基

于规则的HPCC算法保持一致。在数据包广播进程数为16个时，DECC算法的CPU占用率与内存占用率分别为21.9%和30.9%，HPCC算法分别为20.6%和28.6%，相较DECC算法仅提高了6.3%和8.0%；ADPG算法分别为57.6%和62.6%，相较DECC算法下降了61.9%和50.6%。

5 结束语

灵活高效的拥塞控制算法对保障智算中心网络的强大性能至关重要，基于机器学习的拥塞控制算法能够从大量的网络监测数据中提取到有效经验知识，做出更为优秀的发送速率调整决策，但其存在计算开销大和推理时间长的缺陷，难以与现有网络设备适配。为此，本文提出了一种多智能体深度强化学习驱动的拥塞控制算法，通过模型压缩和知识蒸馏的方式将负责深度神经拥塞控制模型转换为轻量化梯度增强二叉决策树模型，极大地减少了智能算法的硬件资源需求和决策时间。实验结果表明，本文算法可在数据处理单元内高效实时运行，从而提升网络的吞吐量、传输速率和稳定性，并表现了优秀的公平性，进一步保障了海量人工智能业务高效可靠运行。基于梯度增强树的智能拥塞控制算法是实现轻量化AI拥塞控制的第一步，但在本文中模型更新采用离线学习的方式，存在更新滞后的缺陷，且尚未考虑对训练数据进行特征提取，从而会对模型训练速度造成一定影响，同时对流量特征的提取与压缩尚未进行优化，导致拥塞控制算法的输入难以保证最优。为此，在未来工作中，笔者将进一步研究网络状态的精细感知技术，以更好为拥塞控制算法提供精准输入，完成更为优秀的调速动作决策，同时对深度神经网络的构建及训练方式进行优化改进，以适应不同网络环境。

参考文献：

[1] 段晓东, 程伟强, 王瑞雪, 等. 面向新型智能计算中心的全调度以太网技术[J]. 中兴通讯技术, 2023, 29(4): 57-63.
DUAN X D, CHENG W Q, WANG R X, et al. Global scheduling Ethernet for new intelligent computing center[J]. ZTE Technology Journal, 2023, 29(4): 57-63.

[2] 赵俊峰, 李芳, 叶晓峰, 等. 面向广域RDMA的确定性网络需求与技术[J]. 电信科学, 2023, 39(11): 39-51.
ZHAO J F, LI F, YE X F, et al. Research on deterministic networking requirements and technologies for RDMA-WAN[J]. Telecommunications Science, 2023, 39(11): 39-51.

[3] 韩博文, 徐博华, 曹畅, 等. 智算中心高性能网络流量调度技术研究及实践[J]. 邮电设计技术, 2024(4): 12-19.
HAN B W, XU B H, CAO C, et al. Research and practice of high-performance network traffic scheduling technology in intelligent computing center[J]. Designing Techniques of Posts and Telecommunications, 2024(4): 12-19.

[4] MITTAL R, SHPINER A, PANDA A, et al. Revisiting network support for RDMA[C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2018: 313-326.

[5] BAI W, ABDEEN S S, AGRAWAL A, et al. Empowering azure storage with RDMA[C]//20th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Berkeley: USENIX Association, 2023: 49-67.

[6] GANGIDI A, MIAO R, ZHENG S B, et al. RDMA over Ethernet for distributed training at meta scale[C]//Proceedings of the ACM SIGCOMM 2024 Conference. New York: ACM Press, 2024: 57-70.

[7] BUI V P, CHIEN T V, LAGUNAS E, et al. Robust congestion control for demand-based optimization in precoded multi-beam high throughput satellite communications[J]. IEEE Transactions on Communications, 2022, 70(10): 6918-6937.

[8] AGARWAL S, KRISHNAMURTHY A, AGARWAL R. Host congestion control[C]//Proceedings of the ACM SIGCOMM 2023 Conference. New York: ACM Press, 2023: 275-287.

[9] LI Z, SHEN X D, XUN H, et al. CoopCon: cooperative hybrid congestion control scheme for named data networking[J]. IEEE Transactions on Network and Service Management, 2023, 20(4): 4734-4750.

[10] ARSLAN S, LI Y L, KUMAR G, et al. Bolt: sub-RTT congestion control for ultra-low latency[C]//20th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Berkeley: USENIX Association, 2023: 219-236.

[11] ZHU Y B, ERAN H, FIRESTONE D, et al. Congestion control for large-scale RDMA deployments[C]//Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. New York: ACM Press, 2015: 523-536.

[12] KUMAR G, DUKKIPATI N, JANG K, et al. Swift: delay is simple and effective for congestion control in the datacen-ter[C]//Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM Press, 2020: 514-528.

[13] LI Y L, MIAO R, LIU H H, et al. HPCC: high precision congestion control[C]//Proceedings of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2019: 44-58.

[14] XIA Z C, WU L B, WANG F, et al. Glider: rethinking congestion control with deep reinforcement learning[J]. World Wide Web, 2023, 26(1): 115-137.

[15] DONG M, MENG T, ZARCHY D, et al. PCC Wivace: online-learning congestion control[C]//15th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Berkeley: USENIX Association, 2018: 343-356.

[16] JAY N, ROTMAN N H, GODFREY P B, et al. A deep reinforcement learning perspective on Internet congestion control[C]//36th International Conference on Machine Learning (ICML). New York: PMLR, 2019: 3050-3059.

[17] LEI K, LIANG Y Z, LI W. Congestion control in SDN-based networks via multi-task deep reinforcement learning[J]. IEEE Network, 2020, 34(4): 28-34.

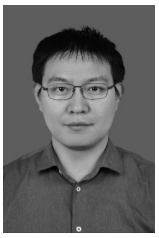
[18] YEN C Y, ABBASLOO S, CHAO H J. Computers can learn from the heuristic designs and master Internet congestion control[C]//Proceedings of the ACM SIGCOMM 2023 Conference. New York: ACM Press, 2023: 255-274.

[19] ABBASLOO S, YEN C Y, CHAO H J. Classic meets modern: a pragmatic learning-based congestion control for the Internet[C]//Proceedings of the Annual Conference of the ACM Special Interest Group on

Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM Press, 2020: 632-647.
[20] TESSLER C, SHPIGELMAN Y, DALAL G, et al. Reinforcement learning for datacenter congestion control[J]. ACM SIGMETRICS Performance Evaluation Review, 2022, 49(2): 43-46.
[21] TIAN H, LIAO X D, ZENG C L, et al. Efficient DRL-based congestion control with ultra-low overhead[J]. IEEE/ACM Transactions on Networking, 2024, 32(3): 1888-1903.
[22] AGARWAL A, ARUN V, RAY D, et al. Towards provably performant congestion control[C]//21st USENIX Symposium on Networked Systems Design and Implementation (NSDI). Berkeley: USENIX Association, 2024: 951-978.
[23] 刘忠沛, 吕高锋, 王继昌, 等. 专用数据处理器综述[J]. 计算机工程与科学, 2023, 45(2): 215-227.
LIU Z P, LYU G F, WANG J C, et al. Review on data processing unit[J]. Computer Engineering & Science, 2023, 45(2): 215-227.
[24] LI G, CHI Y J, WEI Y T, et al. Minimax-optimal multi-agent RL in Markov games with a generative model[C]//Proceedings of the 36th International Conference on Neural Information Processing Systems. New York: ACM Press, 2022: 15353-15367.
[25] SHEN S Q, QIU M W, LIU J, et al. ResQ: a residual Q function-based approach for multi-agent reinforcement learning value factorization[C]// Neural Information Processing Systems, 2022: 5471-5483.
[26] LOU X Z, ZHANG J G, NORMAN T J, et al. TAPE: leveraging agent topology for cooperative multi-agent policy gradient[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2024, 38(16): 17496-17504.
[27] YU Z S, ZHANG X H. Actor-critic alignment for offline-to-online reinforcement learning[J]. Proceedings of Machine Learning Research, 2023, 202: 40452-40474.
[28] BORDELON B, MASSET P, KUO H, et al. Loss dynamics of temporal difference reinforcement learning[C]//Proceedings of the 37th International Conference on Neural Information Processing Systems. New York: ACM Press, 2023: 14469-14496.
[29] DAXBERGER E, KRISTIADI A, IMMER A, et al. Laplace redux-effortless Bayesian deep learning[C]//Proceedings of the 35th International Conference on Neural Information Processing Systems. New York: ACM Press, 2021: 20089-20103.
[30] LANGE M D, ALJUNDI R, MASANA M, et al. A continual learning survey: defying forgetting in classification tasks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 44(7): 3366-3385.
[31] HUH M, CHEUNG B, AGRAWAL P, et al. Straightening out the straight-through estimator: overcoming optimization challenges in vector quantized networks[C]//International Conference on Machine Learning, 2023. Berlin: Springer, 14096-14113.
[32] RANGANATHAN A. The levenberg-marquardt algorithm[J]. Tutoral on LM algorithm, 2004，11(1): 101-110.
[33] YUAN R, LAZARIC A, GOWER R M. Sketched Newton: raphson[J]. SIAM Journal on Optimization, 2022, 32(3): 1555-1583.
[34] XIA M Z, ZHONG Z X, CHEN D Q. Structured pruning learns compact and accurate models[C]//Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL Press, 2022: 1513-1528.
[35] GONG C, CHEN Y, LU Y, et al. VecQ: minimal loss DNN model compression with vectorized weight quantization[J]. IEEE Transactions on Computers, 2021, 70(5): 696-710.
[36] FRANTAR E, ALISTARH D. Optimal brain compression: a framework for accurate post-training quantization and pruning[J]. Advances in Neural Information Processing Systems, 2022, 35: 4475-4488.
[37] XIE G R, LI Q, DONG Y T, et al. Mousika: enable general in-network intelligence in programmable switches by knowledge distillation[C]// Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2022: 1938-1947.
[38] ZHANG Z, HUANG Y, DUAN G, et al. Metis: understanding and enhancing in-network regular expressions[C]//37th International Conference on Neural Information Processing Systems (NIPS). New York: ACM Press, 2023: 77867-77881.
[39] PROKHORENKOVA L, GUSEV G, VOROBEV A, et al. CatBoost: unbiased boosting with categorical features[J]. arXiv Preprint, arXiv: 1706.09516, 2017.
[40] YIN H, LIU P Y, LIU K S, et al. NS3-AI: fostering artificial intelligence algorithms for networking research[C]//Proceedings of the 2020 Workshop on NS-3. New York: ACM Press, 2020: 57-64.
[41] PANDA D K, SUBRAMONI H, CHU C H, et al. The MVAPICH project: transforming research into high-performance MPI library for HPC community[J]. Journal of Computational Science, 2021, 52: 101208.

[作者简介]

陈锦前（1999– ），男，湖南永州人，北京邮电大学博士生，主要研究方向为网络传输优化技术、强化学习、DPU等。



郭少勇（1985– ），男，河北邢台人，博士，北京邮电大学教授、博士生导师，主要研究方向为DPU、区块链应用、边缘智能等。

刘畅（1996– ），女，河北石家庄人，北京邮电大学博士生，主要研究方向为区块链技术、联邦学习等。

亓峰（1971– ），男，山东济南人，博士，北京邮电大学教授、博士生导师，主要研究方向为通信软件。

邱雪松（1973– ），男，江西上饶人，博士，北京邮电大学教授、博士生导师，主要研究方向为网络与业务管理、物联网与区块链。