·热点与综述·　　文章编号：1000-3428(2022)11-0030-09　　文献标志码：A　　中图分类号：TP391

基于深度强化学习的云边协同DNN推理

刘先锋，梁　赛，李　强，张　锦

（湖南师范大学 信息科学与工程学院，长沙 410081）

摘　要：现有基于云边协同的深度神经网络(DNN)推理仅涉及边缘设备同构情况下的静态划分策略，未考虑网络传输速率、边缘设备资源、云服务器负载等变化对DNN推理计算最佳划分点的影响，以及异构边缘设备集群间DNN推理任务的最佳卸载策略。针对以上问题，提出基于深度强化学习的自适应DNN推理计算划分和任务卸载算法。以最小化DNN推理时延为优化目标，建立自适应DNN推理计算划分和任务卸载的数学模型。通过定义状态、动作空间和奖励，将DNN推理计算划分和任务卸载组合优化问题转换为马尔可夫决策过程下的最优策略问题。利用深度强化学习方法，从经验池中学习动态环境下边缘设备与云服务器间DNN推理计算划分和异构边缘集群间任务卸载的近似最优策略。实验结果表明，与经典DNN推理算法相比，该算法在异构动态环境下的DNN推理时延约平均降低了28.83%，能更好地满足DNN推理的低时延需求。

关键词：边缘计算；深度神经网络；深度强化学习；推理计算划分；任务卸载

开放科学(资源服务)标志码(OSID)：

中文引用格式：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.

英文引用格式：LIU X F，LIANG S，LI Q，et al. Cloud-edge collaborative DNN inference based on deep reinforcement learning[J]. Computer Engineering，2022，48(11)：30-38.

Cloud-Edge Collaborative DNN Inference Based on Deep Reinforcement Learning

LIU Xianfeng，LIANG Sai，LI Qiang，ZHANG Jin

（College of Information Science and Engineering，Hunan Normal University，Changsha 410081，China）

【Abstract】In the existing Deep Neural Network（DNN） inference based on cloud-edge collaboration，only the static partition strategy is considered in the homogeneous case of edge devices. However，the influence of the network transmission rate，edge device resources，cloud server load on the optimal partition point of DNN inference computation and the optimal offloading strategy of DNN inference task are not considered in heterogeneous edge device clusters. To solve these problems，this study presents an adaptive DNN inference computation partition and task offloading algorithm based on Deep Reinforcement Learning（DRL）. The aim is to minimize the DNN inference delay，and a mathematical model of adaptive task offloading and DNN inference computation partition is established. The state，action space，and reward are defined to transform task offloading and DNN inference computation partition combination optimization problems into the optimal policy problem under the Markov decision process. DRL is used to learn from the experience pool about the approximate optimal strategy of DNN inference computation partition between edge devices and cloud servers and task offloading between heterogeneous edge clusters in a dynamic environment. The experimental results show that compared with several classical DNN inference algorithms，the DNN inference delay of the proposed algorithm in a heterogeneous dynamic environment is reduced by approximately 28.83% on average，proving that the low latency requirement of DNN inference is met in a better manner.

【Key words】edge computing；Deep Neural Network（DNN）；Deep Reinforcement Learning（DRL）；inference computation partition；task offloading

DOI：10. 19678/j. issn. 1000-3428. 0063579

基金项目：湖南省自然科学基金(2021JJ30456)；湖南省科技计划(2021GK5014，2019SK2161，2018TP1018)。

作者简介：刘先锋(1964—)，男，教授、博士，主研方向为边缘计算、云计算、人工智能；梁　赛，硕士研究生；李　强(通信作者)，副教授、博士；张　锦，教授、博士。

收稿日期：2022-03-31　　**修回日期：**2022-06-05　　**E-mail：**liqiang@hunnu.edu.cn

0　概述

近年来，随着计算机与通信技术的高速发展，网络边缘设备数量及其所产生的数据量呈现爆炸式增长趋势。深度神经网络（Deep Neural Network，DNN）[1]作为支持现代智能移动应用的关键技术，因其高精度和可靠的推理性能，在计算机视觉[2]、自然语言处理[3]、机器翻译[4]等大规模数据处理场景中得到广泛应用与研究。由于DNN是计算密集型网络，通常包含十个以上的网络层及大量的神经元节点，对存储和计算资源要求很高，难以部署在资源受限的边缘设备。现有基于云计算的解决方案存在带宽资源消耗大、通信延迟无法预测和图像数据隐私泄露等问题。为了解决此类问题，文献[5-6]提出基于云边协同的DNN分布式推理，通过将DNN推理部分计算从云端下推到移动边缘，以缓解云服务器的负载压力，实现DNN推理的低延迟和高可靠性[7]。然而，现有工作仅考虑同构设备下的静态划分策略，未考虑网络传输速率、边缘设备资源和云服务器负载等变化对DNN推理计算最佳划分点的影响以及异构边缘设备集群间DNN推理任务的最佳卸载策略。

针对动态环境下边缘设备与云服务器间DNN推理计算划分以及异构边缘设备集群（Edge Device Cluster，EDC）间DNN推理任务卸载问题，本文提出基于深度强化学习（Deep Reinforcement Learning，DRL）的自适应DNN推理计算划分和任务卸载（DNN Inference Computation Partition and Task Offloading Based on Deep Reinforcement Learning，DPTO）算法。DPTO算法对DNN模型各网络层的计算资源需求和输出数据大小进行分析。根据模型结构特征，在资源受限的边缘设备和云服务器间建立分布式DNN协作推理框架，以此优化DNN推理时延。在DNN推理计算划分的基础上，考虑推理任务在异构边缘集群间的卸载问题，对DNN推理计算划分和任务卸载的优化目标和约束条件进行分析，建立DNN推理计算划分和任务卸载的数学模型。DPTO算法具有自适应学习能力，能根据当前环境为DNN推理计算划分和推理任务卸载选择近似最优策略。最终选取3种常用的DNN模型、不同类型的边缘设备和网络传输速率验证DPTO算法的有效性。

1　相关工作

目前，为在算力和存储等资源受限的边缘设备上部署和实现低时延DNN推理，研究人员提出了一系列关于DNN推理的优化技术。现有优化技术包括设计轻量级模型、模型压缩方法和模型提前退出机制。文献[8]提出SqueezeNet小型DNN模型，其将参数量缩减为原来的1/50，极大降低了模型复杂度。文献[9-10]提出模型压缩方法，通过参数共享、剪枝不敏感或冗余的权重参数，降低模型复杂度和资源需求，实现在资源受限边缘设备上的低延迟和低能耗DNN模型推理。文献[11-12]提出模型分支退出机制，通过仅处理较为靠前的若干网络层来加速DNN推理。以上技术虽然能有效降低资源需求、加速DNN推理速度，但同时也会带来模型精度的损失，难以适应实际应用场景中DNN推理愈加严格的精度要求。

研究人员尝试在边缘设备上部署和运行完整的DNN模型，主要分为边边协同和云边协同。边边协同通过将多个边缘设备作为计算节点参与DNN推理以满足单个边缘设备的计算需求，提升系统的整体计算能力。DNN模型通常由卷积层、池化层和全连接层组成，通常卷积层消耗的算力最多。针对这一特性，文献[13]提出DeepThings，利用FTP（Fused Tile Partition）划分方案在空间上将卷积层划分成多个分区任务，并分配给不同的移动边缘设备。文献[14]提出一种寻找最优网络层的划分方式和设备分区配置机制。在良好的网络环境下，文献[13-14]研究有效提高了DNN推理速度，然而DNN卷积层划分之后，相邻分区之间存在重叠数据，将导致设备间高频的重叠数据交互，当通信带宽降低时，DNN推理时延显著增加。

本文通过在Intel Mini PC CPU 1.9 GHz上执行VGG19模型[15]来展示DNN模型的计算和通信特征。图1给出了VGG19各层的输出数据内存大小和计算时延，其中，input表示输入层、conv表示卷积层、fc表示全连接层、pool表示池化层。从图1中可知：1）VGG19各层有显著不同的输出数据内存大小，卷积层增加数据，池化层减少数据，池化层的计算时延较小，卷积层和全连接层的计算时延较高；2）每一层的计算时延和输出数据内存大小不成正比关系，计算时延大的层不一定有较大的输出值。基于以上结论，将DNN模型划分成两部分，计算量小、传输时延大的前几层放在边缘设备端处理，计算量大、传输时延小的后几层放在云端处理。该方法充分利用边缘设备和云服务器的资源，不同层级的计算设备承担不同算力需求的任务，有效权衡了移动边缘设备和云服务器的计算量和通信量。针对基于云边协同的DNN分布式推理，文献[5]提出Neurosurgeon，从最小化时延和能耗的角度出发，基于回归模型估算DNN模型每一层执行时延，结合当前网络带宽返回边缘设备和云服务器间DNN推理计算的最优划分点，以此达到时延或能耗最优。文献[6]利用边缘设备和云服务器的可用资源，为给定应用程序提供有效的模型部署方案。

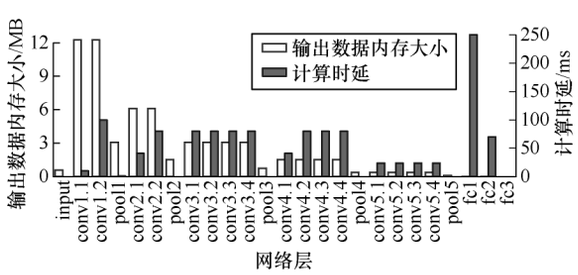

图1　VGG19每层输出数据内存大小及计算时延

Fig.1　Memory size of output data and computation delay of each layer in VGG19

虽然目前已有基于云边协同的DNN分布式推理研究,但是这些研究存在以下不足:

1)对于边缘设备与云服务器间DNN推理计算的划分,网络传输速率、边缘设备资源以及云服务器负载等变化都会直接影响到DNN推理计算的最佳划分。以VGG19为例,图2~图4分别给出了DPTO算法在不同网络传输速率、边缘设备和云服务器负载下,以每层作为划分点的端到端推理时延以及当前环境下最佳划分点的选取,其中:第一列input表示以输入层作为划分点,将DNN推理计算全部卸载到云服务器执行;最后一列fc3表示以输出层作为划分点,将DNN推理计算全部放在边缘设备上执行;其他列表示以该层作为划分点,输入层到该层的推理计算放在边缘设备执行,剩余网络层的推理计算放在云服务器执行;箭头表示当前环境下的最佳划分策略。

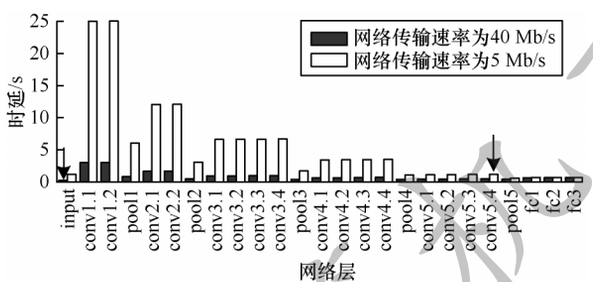

图2 VGG19在不同网络传输速率下以每层作为划分点的端到端推理时延

Fig.2 End-to-end inference delay of VGG19 at different partition points under different network transmission rates

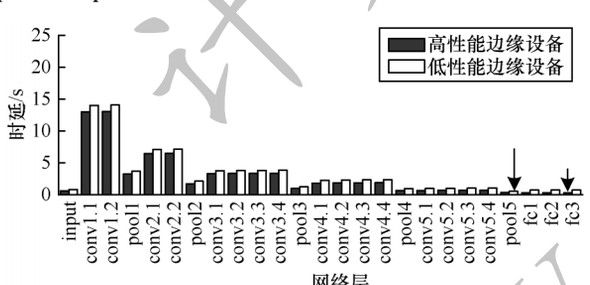

图3 VGG19在不同边缘设备下以每层作为划分点的端到端推理时延

Fig.3 End-to-end inference delay of VGG19 at different partition points under different edge devices

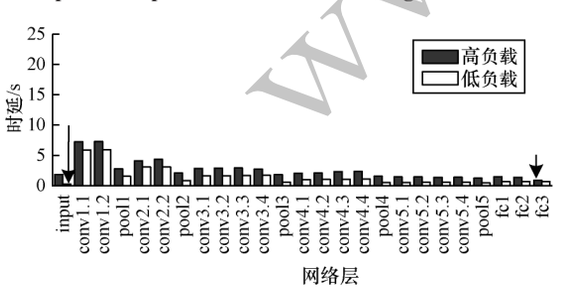

图4 VGG19在不同云服务器负载下以每层作为划分点的端到端推理时延

Fig.4 End-to-end inference delay of VGG19 at different partition points under different cloud server loads

2)已有基于云边协同的DNN模型推理仅考虑了单个边缘设备与云服务器的协作,在实际场景中通常有多个异构边缘设备,当把所有任务全部卸载到单个边缘设备时,会使该边缘设备超出承载范围,造成较大的响应延迟以及其他边缘设备计算资源的浪费。因此,在多个异构边缘设备上进行任务卸载,实现边缘集群内部的协作以此提高计算性能是一个亟待解决的问题。

异构边缘集群中共有m台边缘设备,DNN模型含有n层,在异构边缘集群上进行DNN推理计算划分和任务卸载的所有情况为$m\times(n+1)$。求解空间随着DNN模型层数和边缘设备数量的增加而增大。传统解决方法包括遗传学算法、启发式算法、迭代搜索算法等,这些算法在解决此类问题时取得了一定成果。为了得到最优解,这些算法的设计需要专家知识的辅助,这会导致算法缺乏灵活性、不稳定、效果提升不明显。随着深度学习技术的发展,DRL[16-17]将深度学习与强化学习[18]相结合作为一种人工智能算法,被广泛应用于组合优化、多方博弈等各种复杂决策求解问题,具有重要研究价值。DRL的主要思想是在一个交互环境中利用创建的智能体(agent)不断与环境互动,通过环境反馈的奖励或惩罚信息,逐步逼近最优结果。相比于传统算法,DRL具有以下优势:1)与许多一次性优化方法相比,深度强化学习可以随环境变化调整策略,更好地适应复杂环境的变化;2)DRL在学习过程中不需要了解网络状态随时间变化规律的相关先验知识。DRL通过不断与环境进行交互,能够学习不同状态下应该采取的最优策略。基于以上特点,深度强化学习是解决复杂环境下决策问题的有效方法[19-20]。为此,本文提出基于DRL的自适应DNN推理计算划分和任务卸载算法,对于不同的推理任务,该算法能有效降低推理任务执行总时延。

2 系统框架和推理时延

2.1 系统框架

基于深度强化学习的云边协同DNN推理框架如图5所示。该框架由3个部分组成:1)时间预估模型,基于当前边缘设备和云服务器资源,在不同边缘设备和云服务器上建立不同类型DNN层的时间预估模型;2)DNN推理计算划分和任务卸载,结合当前网络传输速率、边缘设备资源和云服务器负载,通过DPTO算法选择DNN推理计算划分和任务卸载的最佳策略;3)DNN分布式推理,根据DNN推理计算划分和任务卸载策略,将划分点之前的DNN推理计算卸载到相应的边缘设备上执行,划分点之后的DNN推理计算卸载到云服务器执行。

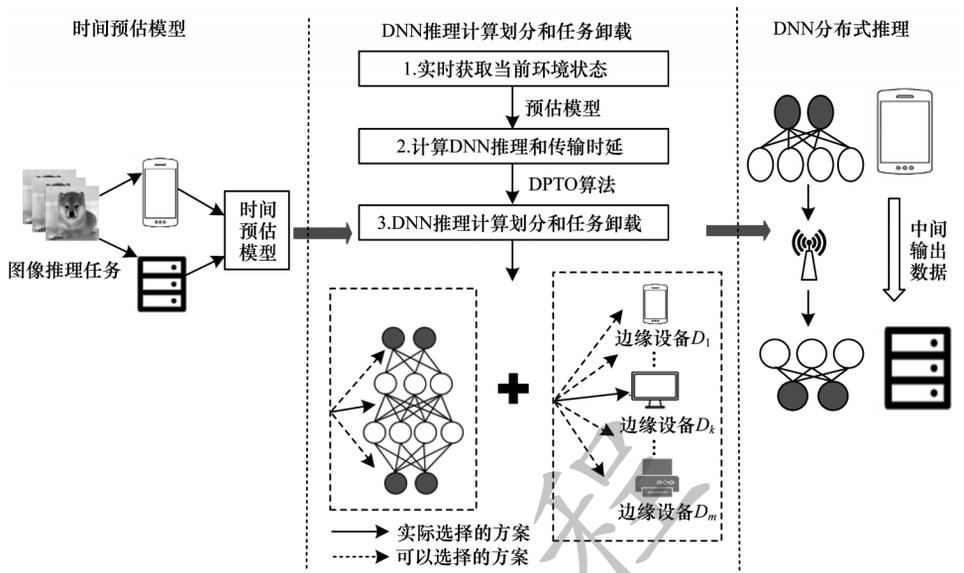

图5　基于深度强化学习的云边协同DNN推理框架

Fig.5　Framework of cloud-edge collaborative DNN inference based on deep reinforcement learning

2.2　时间预估模型

在DNN推理计算过程中，主要计算集中于卷积层和全连接层。卷积层与全连接层的计算时间与每秒浮点运算次数（Floating Point Operations，FLOPs）具有相关性，通过计算FLOPs可以预估卷积层和全连接层的计算时间，为DNN推理计算划分和任务卸载提供决策依据。文献[21]介绍了卷积层和全连接层的FLOPs计算公式，如式（1）和式（2）所示：

$$F_{\mathrm{FLOPs}}=2HW(C_{\mathrm{in}}K^2+1)C_{\mathrm{out}} \tag{1}$$

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

其中：H和W表示输入特征图的高和宽；C_{in}和C_{out}表示卷积计算的输入和输出通道数；K表示卷积核的大小；I、O表示全连接层的输入和输出维数。

假设FLOPs与计算时间的预估模型计算公式如式（3）~式（5）所示：

$$y=kx+b \tag{3}$$

$$k=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2} \tag{4}$$

$$b=\bar{y}-k\bar{x} \tag{5}$$

其中：x表示FLOPs；k表示设备计算能力；y表示计算时间；b表示卷积层或全连接层计算的固有时间开销。通过设置不同的输入特征图（$H\times W$）、输入输出通道数（C_{in}，C_{out}）以及输入和输出维数（I，O）在边缘设备上分别做卷积和全连接运算，求得FLOPs和平均计算时间。记录多组FLOPs与计算时间的值，使用最小二乘法求得预估模型。

2.3　DNN推理时延

DNN模型含有n层，将DNN推理计算在边缘设备和云服务器上进行划分，所有可选划分策略$p=\{0,1,\cdots,n\}$，其中$p=n$和$p=0$为特殊的分区点，$p=n$表示DNN推理计算全部放在边缘设备上处理，$p=0$表示DNN推理计算全部上传到云服务器处理。$0<p<n$表示输入层到第p层在边缘设备上执行，剩余的网络层传输到云服务器执行，网络层第p层的输出数据大小为D_p。当采用云边协同方式执行DNN推理时，单个图片的推理时延$T_{\text{总}}^k$主要由以下3个部分组成：1）边缘设备上的执行时延T_{d}；2）中间输出数据上传到云服务器的传输时间T_{t}；3）云服务器上的执行时延T_{c}。由于推理结果往往很小，其返回的传输延迟通常忽略不计。

$$T_{\text{总}}^k=T_{\mathrm{d}}+T_{\mathrm{t}}+T_{\mathrm{c}} \tag{6}$$

基于式（6）得到的预估模型预测各层在边缘设备和云服务器上执行所需时间，每一张图片的推理时延可以表示如下：

$$T_{\text{总}}^k=\sum_{j=0}^{p}E_{\mathrm{ED}_j}+\frac{D_P}{B}+\sum_{j=p+1}^{n}C_{\mathrm{CS}_j} \tag{7}$$

其中：E_{ED_j}表示第j层在移动设备上执行所需的时间；D_p表示第p层的输出数据大小；B表示网络传输速率；C_{CS_j}表示第j层在云服务器上的执行时间。任务T中含有k张图片，则处理该任务所需的总时延如下：

$$T_{\text{总}}=\sum_{k=1}^{k}T_{\text{总}}^k=\sum_{k=1}^{k}\left(\sum_{j=0}^{p}E_{\mathrm{ED}_j}+\frac{D_p}{B}+\sum_{j=p+1}^{n}C_{\mathrm{CS}_j}\right) \tag{8}$$

综上，优化目标如下：

$$\begin{aligned}&\min\ T_{\text{总}}\\&\text{s.t.}\ \ C_1:\sum_{i=1}^{t}m_{\mathrm{mem}_i}\le M\\&\qquad C_2:\sum_{i=1}^{t}c_{\mathrm{cpu}_i}\le C\\&\qquad C_3:l_j\in p\ \text{and}\ E_{\mathrm{ED}_m}\in E_{\mathrm{EDC}}\end{aligned} \tag{9}$$

其中：C_1、C_2、C_3为优化目标的约束条件。在约束条件C_1和C_2中，M和C分别表示边缘设备的内存和

CPU资源，t表示正在执行的任务数，只有当所有任务所需的内存和CPU资源总和少于边缘设备的总资源时，该边缘设备才可处理新的任务。约束条件C_3表示所选边缘设备和DNN推理计算划分点的策略在所有可选策略中。

3 自适应DNN推理计算划分和任务卸载

3.1 马尔可夫决策过程

为利用DRL方法来寻找最优解，需要把问题转化为强化学习的基本要素。首先将问题建模为马尔可夫决策过程（Markov Decision Process，MDP），MDP可以表示为一个四元组$\{S,A,P,R\}$，其中，S表示环境中的状态空间，A表示动作空间，P表示状态转移函数，$P(s'|s,a)$表示在状态s上执行动作a之后转变为状态s'的概率，R为奖励值函数。

1）状态空间：$S=\{s|s=(B,E_{\mathrm{ED}},C_{\mathrm{CS}},D)\}$，其中，$B$表示当前网络传输速率，$E_{\mathrm{ED}}=(E_{\mathrm{ED}_1},E_{\mathrm{ED}_2},\cdots,E_{\mathrm{ED}_m})$为具有$m$个元素的序列，表示当前$m$个边缘设备中各设备的计算资源，$C_{\mathrm{CS}}$表示云服务器的计算资源，$D=(D_{\mathrm{input}},D_1,D_2,\cdots,D_n)$表示原始输入数据以及DNN模型各层的输出数据大小。

2）动作空间：由所有可选的决策方案组成。对于DNN推理计算划分和任务卸载而言，动作就是在边缘设备和云服务器间选择DNN推理计算划分点以及执行该部分任务的边缘设备。基于当前状态，DNN模型有n+1个推理计算划分点，边缘集群中有m个边缘设备。动作空间$A=\{P_0,P_1,\cdots,P_n;E_{\mathrm{ED}_1},E_{\mathrm{ED}_2},\cdots,E_{\mathrm{ED}_m}\}$，其中，$\{P_0,P_1,\cdots,P_n\}$表示DNN推理计算划分点的集合，$\{E_{\mathrm{ED}_1},E_{\mathrm{ED}_2},\cdots,E_{\mathrm{ED}_m}\}$表示当前所有可用的边缘设备。

3）奖励值：环境会根据智能体执行的动作反馈相应的奖励值。奖励值越大，表示选择的动作越好。以任务推理时延为优化目标，设置任务执行响应时延的负值作为奖励值。

$$r=-T_{\text{总}} \tag{10}$$

其中：r表示奖励值大小。

3.2 基于深度Q网络的DNN推理计算划分和任务卸载

Q-Learning（QL）是无监督的自适应学习算法，从与环境的交互作用中学习知识，适用于求解马尔可夫决策问题。在Q-Learning算法中使用二维矩阵存储每一个状态下所有动作的Q值，即状态-动作值函数$Q(s,a)$。当状态空间和动作空间较大时，受计算机内存限制，会导致Q表爆炸[22]。本文提出基于深度Q网络（Deep Q-Network，DQN）[23]的任务卸载和DNN推理计算划分算法，DQN算法是DRL中的一种具体算法，使用神经网络替换Q表，将Q表更新转化为函数拟合问题。如图6所示，DQN由两个结构相同但参数不同的神经网络构成，在DQN算法的训练过程中，Main网络用来计算当前状态动作对的估计价值$Q(s,a,w)$，w表示Main网络的参数，Target网络用于计算下一状态s'在所有动作下的$Q(s',a',w')$，并选取最大值所对应的动作。根据所选动作a'计算目标价值函数，如式（11）所示：

$$Q_{\text{target}}=r+\gamma\max_{a'}Q(s',a';w') \tag{11}$$

其中：r表示奖励值；γ表示折扣因子。在神经网络的参数更新过程中，Main网络使用最新参数，在每次计算当前状态的估计价值后都会更新其参数。在每隔δ步骤之后，复制Main网络的参数到Target网络以更新Target网络。

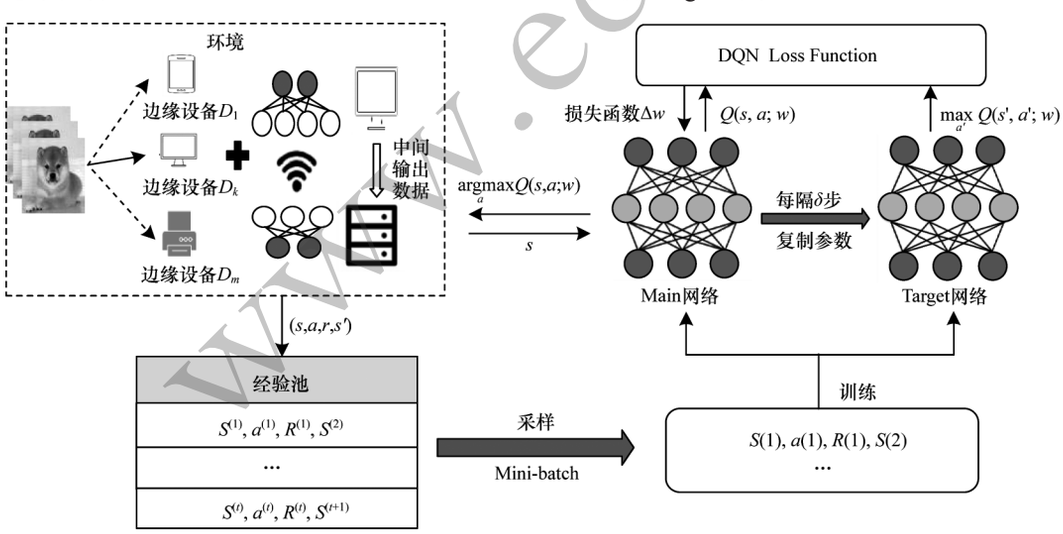

图6 基于DQN的自适应DNN推理计算划分和任务卸载框架

Fig.6 Framework of adaptive DNN inference computation partition and task offloading based on DQN

由于强化学习的动作选择是一个连续的过程，因此前后数据关联性很大，而神经网络的训练通常要求数据样本是静态独立分布。在DQN算法中，除了构建两个神经网络降低状态间的相关性外，还使用经验回放方法解决相关性及非静态分布问题，具体做法为：对于状态s，用ε-greedy[24]策略选取动作

a，与环境交互后生成一个样本(s,a,r,s')，将其储存在经验池中。当经验池中的元组数目达到一定值（例如数量N）时，随机选择经验样本组成Mini-batch来训练DQN。当经验池容量大于N时，则会删除最老的经验样本，仅保留最后的N个经验样本。

DQN算法训练的本质是使得当前Q值无限接近于目标Q值，最终达到收敛状态。损失函数定义如式（12）所示：

$$\mathrm{Loss}(w)=E[(r+\gamma\max_{a'}Q(s',a';\ w')-Q(s,a;\ w))^2] \tag{12}$$

在得到损失函数后，通过损失的反向传播机制调整神经网络参数w，同时Main网络会利用Adam优化器来不断地降低神经网络的损失函数，提高神经网络精度。

3.3　DPTO算法

DPTO算法用来解决异构边缘集群下的任务卸载以及边缘设备与云服务器之间的DNN推理计算划分问题。DPTO首先实时获取各状态特征值，agent根据当前状态输出任务卸载和DNN推理计算划分策略。DPTO算法的伪代码如算法1所示。

算法1　DPTO算法

输入　当前网络传输速率B，当前各边缘设备和云服务器资源$R_S=\{E_{ED_1},E_{ED_2},\cdots,E_{ED_n},C_{CS}\}$，原始输入数据以及DNN模型各层输出数据大小$D=\{D_{input},D_1,D_2,\cdots,D_n\}$，推理任务$T$

输出　DNN推理计算划分和任务卸载策略

1.初始化DQN模型；
2.初始化实验参数；
3.for episode in T do：
4.for each task do：
5.获取DNN推理计算划分点和可用边缘设备集合Ω；
6.设置各特征值得到状态s并传输给agent；
7.使用DQN方法在Ω中选择DNN推理计算划分点和执行该任务的边缘设备；
8.if DNN推理计算划分点为n、边缘设备计算节点为设备m then
9.任务仅在边缘设备m上执行；
10.计算任务执行时延；
11.end
12.else if DNN推理计算划分点为0 then
13.任务卸载到云服务器上执行；
14.计算任务执行时延；
15.end
16.else
17.DNN推理计算划分到边缘设备和云服务器上共同处理；
18.根据式（7）和式（8）计算任务执行时延；
19.end else
20.根据式（9）计算奖励值；
21.更新DPTO算法的最优策略；
22.end for
23.返回到步骤3；
24.end for

DPTO算法的空间复杂度为：$2d(S)+O(1)+H+E+L$，其中：$d(S)$为神经网络参数的维度，存储两个参数相同的神经网络内存空间为$2d(S)$；$O(1)$为其他参数所占的内存空间，例如学习速率、折扣因子等；H为存储所有可选动作的内存空间；E为经验池所占内存空间；L为从经验池中采样的数据样本数目的内存空间。

DPTO算法前向推理的计算复杂度计算如下：在DPTO算法中输入层的神经元个数为所有状态的集合s，网络的第1个隐藏层和第2个隐藏层的节点个数分别设为k_1和k_2，输出层的神经元个数为所有可选动作的集合$m\times n$，则输入层到第1个隐藏层的矩阵运算为$[k_1\times s]\times[s\times 1]$，计算复杂度为$O(k_1\times s^2)$。以此类推，第1个隐藏层到第2个隐藏层的计算复杂度为$O(k_2\times k_1^2)$，第2个隐藏层到输出层的计算复杂度为$O(m\times n\times k_2^2)$。对于T个任务，DPTO算法的总计算复杂度为$O(T(k_1\times s^2+k_2\times k_1^2+m\times n\times k_2^2))$。

4　实验结果与分析

为测试DPTO算法的可行性和有效性，本节搭建仿真实验环境，通过对比实验衡量DPTO算法中基于深度强化学习的DNN推理计算划分和任务卸载在时延方面的优化效果。

4.1　实验平台

实验使用阿里云服务器提供云计算支持，以Intel Mini PC CPU 900 MHz、Intel Mini PC CPU 2.4 GHz、树莓派3B+（1.2 GHz ARM Cortex-A53处理器，1 GB内存）模拟异构边缘设备。边缘设备和云服务器都加载已训练好的DNN模型，边缘设备与云服务器的通信采用TCP/IP协议，Wondershaper设置边缘节点网络传输速率。DNN模型推理采用PyTorch1.3框架，Python3.7.4编程语言。在DPTO算法中设置学习速率为0.01，折扣因子为0.9，训练批次大小为32，经验池大小为500。DPTO算法收敛结果如图7所示。

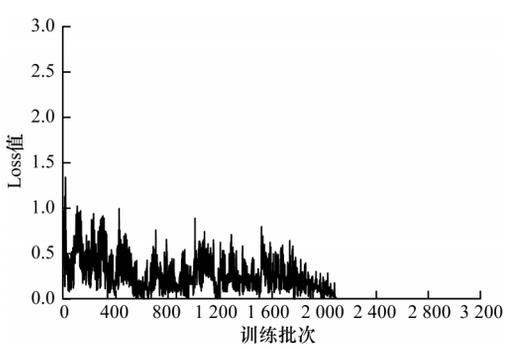

图7　DPTO算法收敛结果

Fig.7　Convergence result of DPTO algorithm

4.2　DNN推理计算划分算法评估

通过在不同环境下执行VGG19[15]、AlexNet[25]、YOLOv2[26]3种DNN模型，对比以下3种经典DNN推理算法来衡量基于深度强化学习的DNN推理计

算划分(DNN Computation Partition Based on Deep Reinforcement Learning,DP)算法对DNN推理时延的优化效果:

1)纯云计算(EC):DNN推理计算全部卸载到云服务器执行,云服务器将推理结果返回给移动设备。

2)纯边缘计算(EM):DNN推理计算全部放在边缘设备上执行。

3)离线划分(OF):通过在离线阶段多次测量所有DNN推理计算划分点的执行时延,选择具有最小平均时延的划分点作为云边协同的DNN推理计算划分点。

4.2.1 不同网络传输速率下DNN推理时延评估

图8给出了EC、EM、OF、DP 4种DNN推理算法在不同网络传输速率下处理单一图片推理所消耗的时间。对于VGG19、AlexNet、YOLOv2 3种不同类型的DNN模型,DP明显优于EC、EM、OF。与EC相比,DP的DNN推理时延减少了约23.45%~60.32%。在EC中,所有数据需上传到云服务器,当网络传输速率过低时,其传输时延过大,导致整个DNN推理时延增大。与EM相比,DP的DNN推理时延减少了约31.86%~62.77%,这是由于边缘设备资源有限,DNN推理计算都在边缘执行,响应时延较大。与OF相比,DP的模型推理时延减少了约0.29%~22.87%,因为在不同网络传输速率下,DNN推理计算的最佳划分点不同。在OF中,固定的DNN推理计算划分点不一定是当前环境下的最佳划分点。当传输速率较高时,DP的推理时延接近于EC,其选择将整个DNN推理计算全部上传到云服务器执行。在传输速率较低时,云边协同DP通过结合当前环境选择最佳DNN推理计算划分点,以使DNN推理时延最小化。

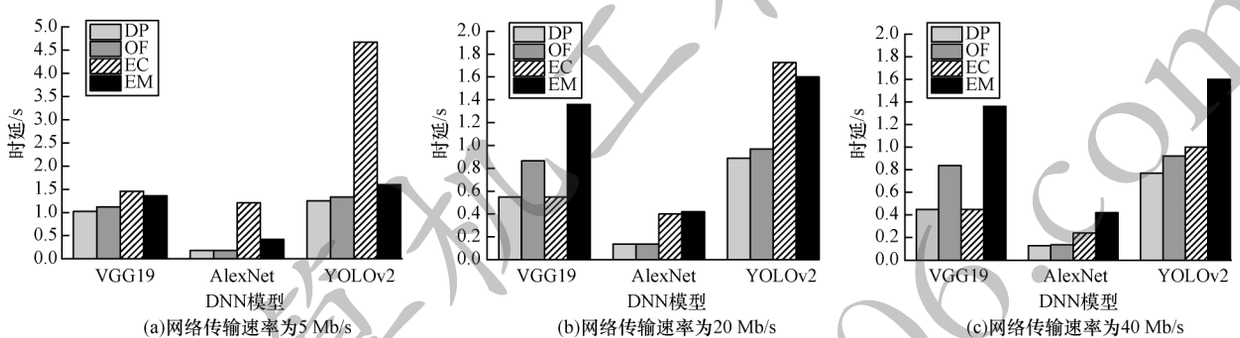

图8 不同网络传输速率下4种算法的DNN推理时延

Fig.8 DNN inference delay for four algorithms under different network transmission rates

4.2.2 不同边缘设备下DNN推理时延评估

VGG19、AlexNet、YOLOv2 3种DNN模型在不同推理算法和边缘设备(高性能边缘设备Intel Mini PC CPU 900 MHz和低性能边缘设备Intel Mini PC CPU 2.4 GHz)下的推理时延如图9所示。DP的DNN推理时间相比于EM减少了约18.35%~39.36%,相比于EC减少了约46.48%~61.35%。当边缘设备计算能力较强时,DP的推理时延接近于EM,其选择将整个DNN推理计算全部放在边缘设备上执行,避免数据传输带来较大的响应延迟。当边缘设备计算能力较弱时,DP在边缘设备和云服务器之间自适应分配DNN推理计算或将整个DNN推理计算全部上传到云服务器执行。由于DP能在动态环境下做出最优决策,DNN推理时间相比于OF减少了约6.22%~10.29%。

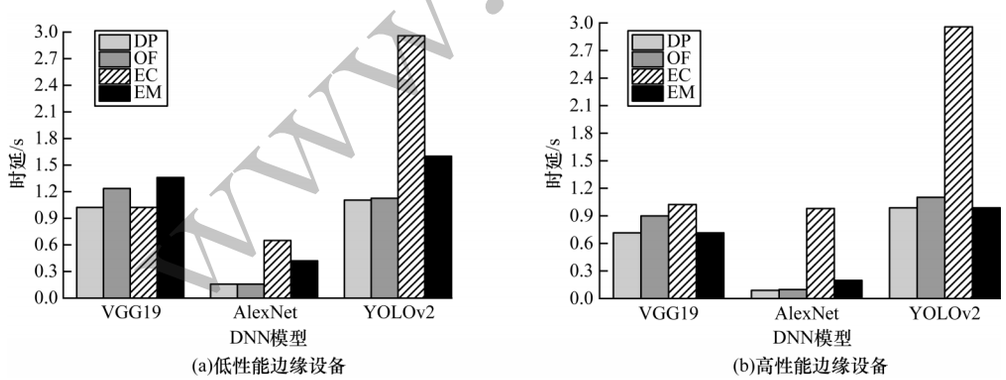

图9 不同边缘设备下4种算法的DNN推理时延

Fig.9 DNN inference delay for four algorithms under different edge devices

4.2.3 不同云服务器负载下DNN推理时延评估

图10给出了4种算法在不同云服务器负载下的DNN推理时延。DP的DNN推理时延相比于EM减少了约0.24%~55.39%,相比于EC减少了约28.12%~48.88%。当云服务器负载相对较高时,DP的DNN推理时延接近于EM,其选择将整个DNN推理计算全部放在边缘设备上执行,避免云服务器负载过高带来巨大的响应延迟。当云服务器负载相对较低且

网络状况良好时,DP将DNN推理计算全部上传到云服务器执行。在其他情况下,DP将DNN推理计算划分成两部分,分别在边缘设备和云服务器上共同执行。DP的推理时延相比于OF减少了约14.41%~27.12%,因为DP能在动态环境下自适应选择最佳DNN推理计算划分点。

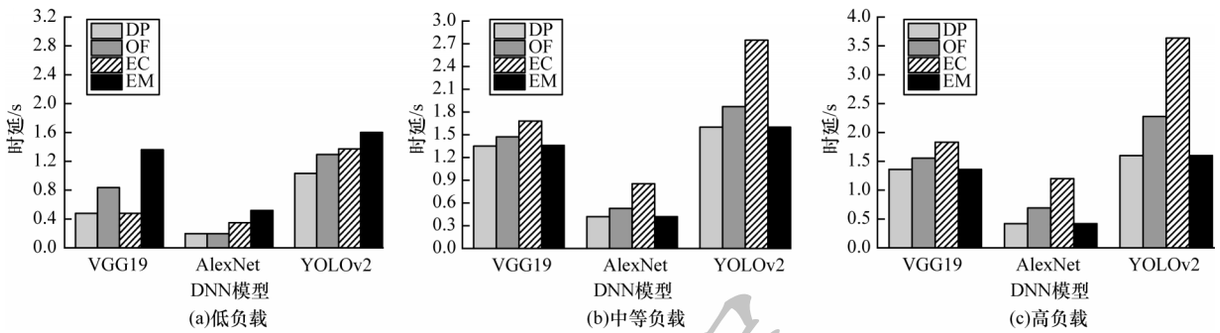

图10　不同云服务器负载下4种算法的DNN推理时延

Fig.10　DNN inference delay of four algorithms under different cloud server loads

4.3　DPTO算法评估

在4.2节中验证了基于深度强化学习的DNN推理计算划分对DNN推理时延的优化效果,根据不同任务的推理情况,其中任务1、任务2、任务3、任务4分别包含1、30、50、100张不同图片,对比以下3种算法来评估DPTO算法在DNN推理时延上的进一步优化效果:

1)DP:仅考虑DNN推理计算的划分,未考虑DNN推理任务在异构边缘集群上的卸载优化,其将所有任务分配给当前计算资源最多的边缘设备处理。

2)轮询(RR):将任务随机分配给边缘设备进行处理。

3)QL:具有良好学习效果的经典强化学习算法,基于Q表选择卸载策略。

图11给出了4种不同任务在不同算法下的执行时间。对于4种不同的任务,DPTO的任务执行时间相比于DP减少了48.08%,在DP中把所有任务全部卸载到单个边缘设备时,会导致该边缘设备出现资源竞争问题,任务等待时间变长。而其他边缘设备上的计算资源处于空闲状态,造成响应时延的增加和计算资源的浪费。与RR相比,DPTO的任务执行时间减少了29.18%,由于RR没有考虑边缘设备计算能力的差异性,边缘设备之间的负载不平衡导致任务执行产生较大时延。与QL相比,DPTO的任务执行时间减少了3.99%,QL基于Q表选择卸载策略,Q表采用二维数组存储每一状态下的动作值函数,随着任务和边缘设备数量的增加,QL中的状态和动作数量增多,通过遍历整个Q表寻找最优卸载策略的开销时延增大,从而导致任务执行时延也增大。综上,不同的卸载策略对任务的执行时延有着较大的影响。当只有一个任务时,边缘设备的计算压力较小,选择最好的边缘设备就能应对该任务。但随着任务的不断增加,根据任务的大小选择合适的边缘设备计算节点对时延的优化具有重要的作用。

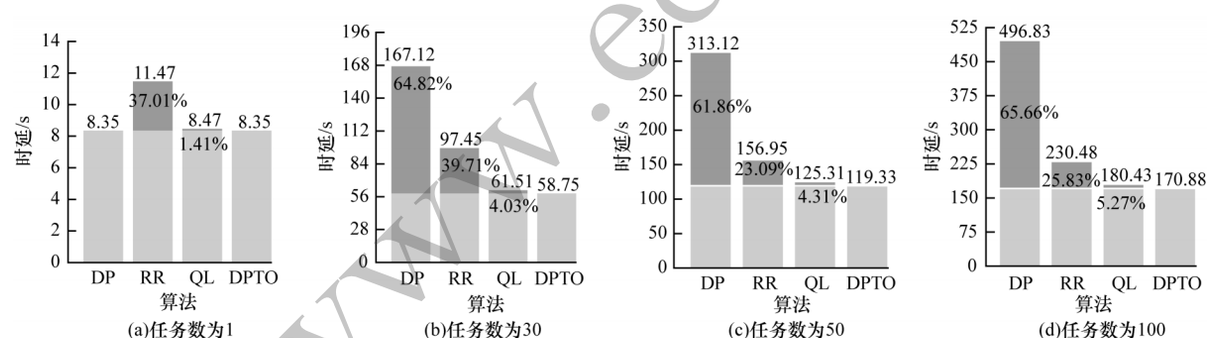

图11　不同任务数量下4种算法的执行总时延

Fig.11　Total execution delay of four algorithms under different number of tasks

5　结束语

目前,深度神经网络发展迅速,广泛应用于计算机视觉、自然语言处理和机器翻译等智能任务。为了优化DNN推理性能,本文提出基于深度强化学习的云边协同DNN推理算法。将动态环境下边缘设备和云服务器间DNN推理计算划分以及异构边缘集群间任务卸载转换为马尔可夫决策过程下的最优策略确定问题,利用深度强化学习方法在经验池中学习异构动态环境下DNN推理计算划分和任务卸载最佳策略。实验结果表明,对于不同的DNN推理任务和DNN模型,DPTO算法在不同环境下相比于已有算法推理时延平均降低了约28.83%。下一步将对边缘设备宕机和恶意攻击等异常情况进行研究,提出高效的容错解决方案,以保证云边协同DNN推理的可靠性和安全性。

参考文献

[1] LIU W B, WANG Z D, LIU X H, et al. A survey of deep neural network architectures and their applications[J]. Neurocomputing, 2017, 234: 11-26.

[2] 姚相坤,万里红,霍宏,等. 基于多结构卷积神经网络的高分遥感影像飞机目标检测[J]. 计算机工程,2017,43(1):259-267.
YAO X K, WAN L H, HUO H, et al. Airplane object detection in high resolution remote sensing imagery based on multi-structure convolutional neural network [J]. Computer Engineering, 2017, 43(1): 259-267. (in Chinese)

[3] YOUNG T, HAZARIKA D, PORIA S, et al. Recent trends in deep learning based natural language processing[J]. IEEE Computational Intelligence Magazine, 2018, 13(3): 55-75.

[4] POPEL M, TOMKOVA M, TOMEK J, et al. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals[J]. Nature Communications, 2020, 11(1): 1-15.

[5] KANG Y, HAUSWALD J, GAO C, et al. Neurosurgeon: collaborative intelligence between the cloud and mobile edge[J]. ACM SIGARCH Computer Architecture News, 2017, 45(1): 615-629.

[6] 赵鹏程,高尚,于洪梅. 基于多智能体深度强化学习的空间众包任务分配[J]. 吉林大学学报(理学版),2022,60(2):321-331.
ZHAO P C, GAO J, YU H M. Spatial crowdsourcing task allocation based on multi-agent deep reinforcement learning [J]. Journal of Jilin University(Science Edition), 2022, 60(2): 321-331. (in Chinese)

[7] 施巍松,张星洲,王一帆,等. 边缘计算:现状与展望[J]. 计算机研究与发展,2019,56(1):69-89.
SHI W S, ZHANG X Z, WANG Y F, et al. Edge computing: state-of-the-art and future directions[J]. Journal of Computer Research and Development, 2019, 56(1): 69-89. (in Chinese)

[8] IANDOLA F N, HAN S, MOSKEWICZ M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0. 5 MB model size[EB/OL]. [2022-02-05]. https://arxiv. org/abs/1602. 07360.

[9] LIU S, LIN Y, ZHOU Z, et al. On-demand deep model compression for mobile devices: a usage-driven model selection framework[C]//Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. New York, USA: ACM Press, 2018: 389-400.

[10] HAN S, MAO H Z, DALLY W J. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding[EB/OL]. [2022-02-05]. https://arxiv. org/abs/1510. 00149.

[11] LI E, ZENG L K, ZHOU Z, et al. Edge AI: on-demand accelerating deep neural network inference via edge computing[J]. IEEE Transactions on Wireless Communications, 2020, 19(1): 447-457.

[12] TEERAPITTAYANON S, MCDANEL B, KUNG H T. BranchyNet: fast inference via early exiting from deep neural networks[C]//Proceedings of the 23rd International Conference on Pattern Recognition. Washington D. C., USA: IEEE Press, 2016: 2464-2469.

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

[14] ZHOU L, SAMAVATIAN M H, BACHA A, et al. Adaptive parallel execution of deep neural networks on heterogeneous edge devices[C]//Proceedings of the 4th ACM/IEEE Symposium on Edge Computing. New York, USA: ACM Press, 2019: 195-208.

[15] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[EB/OL]. [2022-02-05]. https://arxiv. org/abs/1409. 1556.

[16] HUANG L, BI S, ZHANG Y J A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks [J]. IEEE Transactions on Mobile Computing, 2019, 19(11): 2581-2593.

[17] 杨思明,单征,丁煜,等. 深度强化学习研究综述[J]. 计算机工程,2021,47(12):19-29.
YANG S M, SHAN Z, DING Y, et al. Survey of research on deep reinforcement learning[J]. Computer Engineering, 2021, 47(12): 19-29. (in Chinese)

[18] 杨锦翔,熊焰,黄文超. 基于强化学习的安全协议形式化验证优化研究[J]. 计算机工程,2021,47(12):141-146.
YANG J X, XIONG Y, HUANG W C. Research on optimization of security protocol formal verification based on reinforcement learning[J]. Computer Engineering, 2021, 47(12): 141-146. (in Chinese)

[19] ZOU J, HAO T, YU C, et al. A3C-DO: a regional resource scheduling framework based on deep reinforcement learning in edge scenario [J]. IEEE Transactions on Computers, 2021, 70(2): 228-239.

[20] YAN J, BI S Z, ZHANG Y J A. Offloading and resource allocation with general task graph in mobile edge computing: a deep reinforcement learning approach[J]. IEEE Transactions on Wireless Communications, 2020, 19(8): 5404-5419.

[21] MOLCHANOV P, TYREE S, KARRAS T, et al. Pruning convolutional neural networks for resource efficient transfer learning[EB/OL]. [2022-02-05]. https://arxiv. org/abs/1611. 06440.

[22] KAN N, ZOU J, TANG K, et al. Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming [C]//Proceedings of 2019 IEEE International Conference on Acoustics, Speech and Signal Processing. Washington D. C., USA: IEEE Press, 2019: 4030-4034.

[23] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529-533.

[24] MIGNON A, ROCHA R L. An adaptive implementation of ε-greedy in reinforcement learning[J]. Procedia Computer Science, 2017, 109: 1146-1151.

[25] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImagNet classification with deep convolutional neural networks[J]. Advances in Neural Information Processing Systems, 2012, 25(2): 1097-1105.

[26] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: unified, real-time object detection [C]// Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Press, 2016: 779-788.

编辑 陆燕菲