周陈静, 骆淑云. 基于深度强化学习的实时视频边缘卸载策略[J]. 智能计算机与应用,2024,14(8):32-39. DOI:10.20169/j.issn.2095-2163.240806

基于深度强化学习的实时视频边缘卸载策略

周陈静, 骆淑云
(浙江理工大学 计算机科学与技术(人工智能)学院, 杭州 310018)

摘 要: 视频边缘计算架构将视频任务部分或全部卸载至靠近网络的边端进行处理,为实时视频处理提供了一个解决方案。为加速视频处理,本文搭建了一个视频边缘计算系统。针对具体的疵点检测视频任务,考虑视频处理的实时性和准确度要求,对计算卸载问题进行建模;为使卸载策略更好地适应动态的网络环境,将问题转换为马尔可夫决策过程,提出了一种基于深度强化学习和队列预测的视频计算卸载策略。实验结果表明,该卸载策略相较于基准策略能够在保证准确度的情况下,最大程度地降低系统时延。

关键词: 视频边缘计算; 实时视频处理; 深度强化学习
中图分类号: TP183/TN929.5 **文献标志码**: A **文章编号**: 2095-2163(2024)08-0032-08

Computation offloading decision in video edge computing based on deep reinforcement learning

ZHOU Chenjing, LUO Shuyun
(School of computer science and technology(Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: The video edge computing architecture provides a solution for real-time video processing by offloading partial or all of the video to the edge end close to the network for processing. A video edge computing system is built to accelerate video processing. The computational offloading problem is modelled for a specific fault detection video task, taking into account the real-time and accuracy requirements of video processing. In order to adapt to the dynamic network environment, the problem is transformed into a Markovian decision process and a deep reinforcement learning-based video computational offloading policy is proposed. The experimental results indicate that the offloading strategy is able to minimize system latency with guaranteed accuracy compared to other baseline strategy.

Key words: video edge computing; real-time video processing; deep reinforcement learning

0 引 言

智能制造、智慧城市等概念的迅速推进,使得各类摄像头、移动设备和可穿戴设备遍布各地,产生了数以亿计的视频数据亟待处理[1-2]。基于深度学习的工业质检算法以其高效、准确等优势被引入工厂,推动智能制造的发展,同时也对通信技术与终端设备提出了挑战[3-5]。以织物疵点检测为例,在织物的织造、印染加工等生产工序中对织物疵点进行自动化检测时,工业摄像头通常会产生大量视频数据,且视频数据需要实时处理,及时发现疵点,但此类终端设备通常受到自身硬件条件限制,难以实时处理视频任务。

传统的云计算架构将视频传输至云服务器集中处理,由于视频任务的爆炸式增长,常常会出现网络拥塞、任务队列堆积过大等问题,无法满足任务实时性要求,且由于云计算中心与移动设备存在一定传输距离,无法保证数据在传输时的安全性。移动边缘计算(Mobile Edge Computing,MEC)作为一种新型计算架构,将任务卸载至网络边缘端处理,能够很好地解决云计算架构中高延迟和数据安全问题[6-8]。

目前,视频边缘计算相关研究多围绕视频分析处理、自适应视频流等问题。Ran 等[9]提出一个用于边缘视频分析的深度学习框架 DeepDecision,能

基金项目: 浙江省尖兵研发攻关计划项目(2023C01041)。
作者简介: 周陈静(1998-),女,硕士研究生,主要研究方向:边缘计算,强化学习。
通讯作者: 骆淑云(1986-),女,博士,讲师,主要研究方向:工业智能边缘计算,网络经济。Email:Shuyunluo@ zstu.edu.cn
收稿日期: 2023-05-22

够在云、边缘或本地远程执行深度学习，并考虑了模型精度、视频质量和网络条件等因素确定卸载策略；Chen 等[10]提出了一个基于边缘计算的智能视频监控系统，将深度学习网络部署在边缘计算环境之中，将计算工作负载从网络中心迁移至网络边缘，以减少网络通信开销。

上述研究均涉及视频分析处理领域，考虑了深度模型部署问题，其卸载策略通常采用粗粒度的二进制卸载策略，即本地处理或完全卸载处理。而自适应视频流计算卸载研究通常使用部分卸载策略。视频流具有边传输边播放的流式传输特点，常常应用于视频播放、视频会和在线直播等系统中。这类研究主要提高用户视频观看体验，有些用户偏向于视频清晰度更高，有些用户能够容忍视频的低清晰度，但无法容忍视频卡顿。自适应视频流能够按照用户偏好来决定编码方式，满足不同用户需求。

Tran 等[11]研究了 MEC 中自适应比特率视频缓存与处理问题，提出了一种联合协作缓存和处理框架，将视频缓存和请求调度问题转换成整数线性规划问题，目的是最小化视频检索的延迟成本，同时提出了启发式自适应码率（Adaptive Bitrate Video Streaming，ABR）感知主动缓存放置算法以及在线低复杂度视频请求调度算法来寻求最优解。Wang 等[12]考虑到动态网络条件，提出了一种基于边缘计算范式的自适应无线视频转码框架，并提出了带宽调整算法，以提高带宽利用率；Liu 等[13]则针对基于区块链的视频流边缘计算系统，设计了一种激励机制，提出了一种基于区块链的视频流块大小自适应方案；Zhou 等[14]则针对 3D 视频流的问题，结合 MEC 和软件定义网络提出了一种新的网络架构，同时提出一种基于 AC（Actor-Critic）的深度强化学习（Deep Reinforcement Learning，DRL）算法用于体验质量优化，并引入长短期记忆递归神经网络（Long Short Term Memory，LSTM）进行带宽预测。综上，现有视频计算卸载系统主要以加速视频编解码任务为主。在用于加速视频分析任务的视频计算卸载系统中，卸载方案为二进制卸载策略，相较于部分卸载，处理效率不高，无法保证系统实时性要求，同时未考虑视频检测中准确度的影响。

本文针对视频计算卸载的特定场景，考虑检测准确度对卸载决策所带来的影响，对计算卸载问题进行建模，并提出了一种基于深度强化学习和队列预测的计算卸载策略，以满足实时性和准确度的需求；在所提策略中，加入了准确度奖励和额外奖励引导智能体做出更优的决策；为避免智能体追求高准确度而不断向服务器卸载任务，在双延迟深度确定性策略梯度（Twin Delayed Deep Deterministic Policy Gradient，TD3）算法的基础上加入时序信息，预测边缘服务器的任务队列拥塞程度，给予智能体一定的时序信息，使其更好的感知周围环境以优化卸载策略。实验表明，所提卸载策略相较于其他卸载策略，能够在保证系统准确度的前提下，最大程度地降低系统时延。

1　视频任务边缘计算卸载系统模型

为了加快视频处理速度，本文搭建了一个视频边缘计算系统，如图 1 所示。

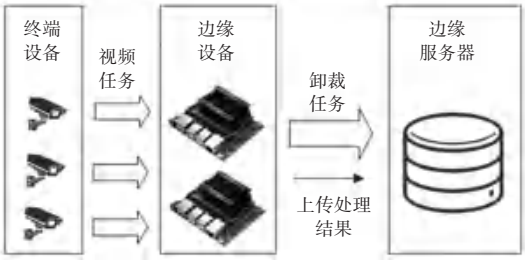

图 1　视频边缘计算系统

Fig. 1　Video edge computing system

该系统由多个终端设备、边缘设备（Edge Devices，EDs）以及边缘服务器（Edge Servers，ESs）组成。终端设备生成并发送视频任务至 EDs；EDs 接收待处理视频任务后，可将其分割成多个时长相同的视频块部分卸载至 ESs 进行处理；最终 EDs 将结果发送至 ESs，将结果存储至服务器，以方便用户调取。假设现有 N 个边缘设备 $n=\{1,2,\cdots,N\}$ 以及 M 个边缘服务器 $m=\{1,2,\cdots,M\}$，在每个时隙 $t=\{1,2,\cdots,T\}$ 开始时，边缘设备 n 接收视频任务 $V_n(t)$ 并将其分割成 $D_n(t)$ 个等长的视频块；边缘设备 n 根据卸载策略 $A=\{\alpha_n(t),m\}$，$\alpha_n(t)\in[0,1]$，确定卸载服务器以及卸载视频块数量。

待卸载视频块数量：

$$N_n^O(t)=\lceil D_n(t)\times\alpha_n(t)\rceil \tag{1}$$

其中，$\lceil\cdot\rceil$ 表示向上取整。

本地处理视频块数量：

$$N_n^L(t)=D_n(t)-N_n^O(t) \tag{2}$$

1.1　本地处理模型

本地处理的视频块需要编解码，将格式转换为 RGB 格式，再进行视频推理，编解码的速率取决于边缘设备的计算能力，将 f_{cpu}^e 表示边缘设备的 CPU 频率，则其对于边缘设备 n 在本地处理的第 i 个视频块，

其编解码时延如式(3)所示:

$$L_{n,i}^{e\mathrm{RTR}}(t)=\frac{Q_n^{e,\mathrm{RTR}}(t)}{f_{\mathrm{cpu}}^e}C^{e,\mathrm{RTR}}+\frac{i}{f_{\mathrm{cpu}}^e}C^{e,\mathrm{RTR}} \tag{3}$$

其中,$Q_n^{e,\mathrm{RTR}}(t)$ 表示在时隙 t 时编解码队列长度,$C^{e,\mathrm{RTR}}$ 表示单个视频块进行编解码操作所需的计算资源。

视频块完成视频编解码后,进入至视频推理队列,对于本地处理的第 i 个视频块,在完成视频编解码后,需等待推理队列中排在之前的任务完成推理后,才能开始推理。因此对于边缘设备 n 在本地处理的第 i 视频块,其处理时延如式(4)所示:

$$L_{n,i}^{e,\mathrm{Inf}}(t)=\max(L_{n,i}^{e,\mathrm{RTR}}(t),L_{n,i-1}^{e,\mathrm{Inf}}(t))+\frac{C^{\mathrm{Inf}}}{f_{\mathrm{gpu}}^e} \tag{4}$$

其中,f_{gpu}^e 表示边缘设备视频推理处理频率。

视频处理结果上传至服务器的时延忽略不计,则任务 $V_n(t)$ 在本地处理的总时延即第 $N_n^L(t)$ 个视频块在本地处理完成后的时延,如式(5)所示:

$$L_n^e(t)=L_{n,N_n^L(t)}^{e,\mathrm{Inf}}(t)=\max(L_{n,N_n^L(t)}^{e,\mathrm{RTR}}(t),L_{n,N_n^L(t)}^{e,\mathrm{Inf}}(t))+\frac{C^{\mathrm{Inf}}}{f_{\mathrm{gpu}}^e} \tag{5}$$

1.2 卸载处理模型

卸载至边缘服务器处理的视频块,需要进行编解码,将格式转换为流媒体格式,传输至服务器后,再进行视频推理。在时隙 0 时,边缘设备生成编解码队列以及传输队列,边缘服务器生成视频推理队列。

在时隙 t 时,编解码队列以及传输队列长度分别为 $Q_n^{e,\mathrm{RTF}}(t)$ 和 $Q_n^{\mathrm{Trans}}(t)$,此时对于边缘设备 n 卸载处理的第 i 视频块,其编解码时延,如式(6)所示:

$$L_{n,i}^{e,\mathrm{RTH}}(t)=\frac{Q_n^{e,\mathrm{RTF}}(t)}{f_{\mathrm{cpu}}^e}C^{e,\mathrm{RTF}}+\frac{i}{f_{\mathrm{cpu}}^e}C^{e,\mathrm{RTF}} \tag{6}$$

视频块完成视频编解码可进行视频传输。边缘设备通过无线链路将视频块上传至服务器,传输时延由视频数据量以及传输时延决定。在时隙 t 时,终端设备与边缘服务器的无线传输速率 $\tau(t)$ 表示为式(7):

$$\tau(t)=B(t)\log_2\left(1+\frac{P\cdot h(t)}{\sigma^2}\right) \tag{7}$$

其中,σ^2 表示噪声功率;P 表示发射功率;$B(t)$ 和 $h(t)$ 分别表示时隙 t 的边缘服务器和终端设备之间的上行带宽和信道增益。

针对卸载处理的 $N_n^o(t)$ 个视频块,每个视频块按帧进行传输,因此无需等待编解码完全完成,只需视频块开始编解码产生了一定量的帧数据后,视频块便可开始传输,因此,单个视频块的传输时延如式(8)所示:

$$L^{e,\mathrm{trans}}=\max\left(\frac{C^{e,\mathrm{RTF}}}{f_{\mathrm{cpu}}^e},\frac{V_n(t)}{D_n(t)\tau(t)}\right) \tag{8}$$

其中,$\frac{V_n(t)}{D_n(t)\tau(t)}$ 表示传输单个已编解码完毕的视频块所需的传输耗时。

由于视频块可以边解码边传输,因此其传输时延由其编解码速率和传输速率决定。如编解码速率较慢,则传输速率更快,其传输时延主要由编解码速率决定;同理,如传输速率更慢,则传输时延主要由传输速率决定,因此第 i 个视频块的传输时延如式(9)所示:

$$L_{n,i}^{\mathrm{trans}}(t)=L_{n,i}^{\mathrm{wait}}(t)+L^{e,\mathrm{trans}} \tag{9}$$

其中,$L_{n,i}^{\mathrm{wait}}(t)$ 表示视频块的等待时延。

视频块需在自身开始编码后且前 $i-1$ 个视频块已被传输后才开始传输,其等待时延可表示为式(10):

$$L_{n,i}^{\mathrm{wait}}(t)=\max(L_{n,i-1}^{\mathrm{trans}}(t),L_{n,i-1}^{e,\mathrm{RTF}}(t)) \tag{10}$$

视频块传输至服务器后,其在服务器中的编解码时延较小且处理速度稳定,可忽略不计。在时隙 t 时,第 m 个服务器的推理队列长度为 $Q_m^{s,Inf}(t)$,因此当前时隙 t 时服务器中的视频推理队列中的剩余任务完成时延 $L_{m,0}^{s,\mathrm{inf}}(t)$ 可表示为式(11):

$$L_{m,0}^{s,\mathrm{inf}}(t)=\frac{Q_m^{s,\mathrm{Inf}}(t)\cdot C^{\mathrm{Inf}}}{f_{\mathrm{gpu}}^s} \tag{11}$$

其中,f_{gpu}^s 表示边缘服务器视频推理处理频率。

在编解码之后,视频块进入视频推理队列,则第 i 个视频块的处理时延,式(12):

$$L_{m,i}^{s,\mathrm{Inf}}(t)=\max(L_{n,i}^{\mathrm{trans}}(t),L_{m,i-1}^{s,\mathrm{Inf}}(t))+\frac{C^{\mathrm{Inf}}}{f_{\mathrm{gpu}}^s} \tag{12}$$

因此任务 $V_n(t)$ 卸载处理的总时延即第 $N_n^O(t)$ 个视频块在服务器处理完成后的时延,式(13):

$$L_m^s(t)=L_{m,N_n^O(t)}^{s,\mathrm{Inf}}(t)=\max(L_{n,N_n^O(t)}^{\mathrm{trans}}(t),L_{m,N_n^O(t)-1}^{s,\mathrm{Inf}}(t))+\frac{C^{\mathrm{Inf}}}{f_{\mathrm{gpu}}^s} \tag{13}$$

综上,任务 $V_n(t)$ 的总处理时延如公式(14)所示:

$$L_n(t)=\max(L_n^l(t),L_m^s(t)) \tag{14}$$

1.3 视频处理准确度模型

在视频检测任务中,准确度是衡量是否正确完成检测任务的指标[15]。模型参数越复杂,其准确度相对越高。部署复杂模型需要较多的计算资源和存储资源,边缘设备的计算和存储资源有限,一般无法

部署过大的模型来满足较高准确度的需求，而边缘服务器通常具备足够的资源部署较大的模型[16]。复杂模型的处理时延相对更长，在一定程度上会对卸载决策产生影响。本文将平均精度均值（Mean Average Precision，mAP）作为模型性能的衡量指标。假设边缘设备中所部署的视频处理模型 mAP 为 a_e，边缘服务器端部署的模型 mAP 为 a_s，则单个边缘设备的平均精度可表示为公式（15）：

$$mAP_n(t) = t \times a_s + (1 - \alpha(t)) \times a_e \quad (15)$$

1.4　问题模型

根据时延模型，整个系统成本可表示为式（16）：

$$C(t) = \max(L(t)) \quad (16)$$

其中，$L(t) = \max(L_1(t), L_2(t), \cdots, L_N(t))$。

为了保证实时性，应使系统时延更小。因此该系统优化问题可表述为 $\arg\min_{A(t)} C(t)$，$A(t)$ 表示各终端设备做出的卸载决策的集合，$t \in \{1, \cdots, T\}$，a 表示系统的目标准确度，$mAP_m \geqslant a, a \in [0,1]$。

2　基于 TD3 和队列预测的计算卸载策略

2.1　马尔科夫决策过程

为了使用强化学习算法来求解计算卸载决策问题，需要将该问题转化为马尔可夫决策过程（Markov Decision Process，MDP）模型[17]。常见的 MDP 模型由一个四元组 $\{S, A, R, P\}$ 构成，S 代表状态，A 代表动作，R 代表奖励，P 为状态转移方程，即 $P(S(t+1) \mid S(t), A(t))$。

系统的状态空间由任务数量、带宽信息以及边缘服务器和边缘设备的队列信息组成。系统的状态空间如式（17）所示：

$$s(t) = \{B(t), Q^{e,\mathrm{RTR}}(t), Q^{e,\mathrm{RTH}}(t), Q^{e,\mathrm{Inf}}(t), Q^{\mathrm{trans}}(t), Q^{s,\mathrm{Inf}}(t), \overline{Q^{s,\mathrm{Inf}}}(t)\} \quad (17)$$

其中，$\overline{Q^{s,\mathrm{Inf}}}(t)$ 表示预测的服务器队列信息，由 LSTM 网络给出。

预测服务器队列在下一时刻的拥塞状态，智能体可根据预测的队列趋势做出相对应的选择。

整体动作空间由边缘设备的卸载决策组成，即卸载率 $\alpha_n(t)$ 以及服务器的选择 $m_n(t)$，因此动作空间可以被定义为式（18）：

$$A(t) = \begin{bmatrix} (\alpha_1(t), m_1(t)), \\ (\alpha_2(t), m_2(t)), \\ \cdots, \\ (\alpha_N(t), m_N(t)) \end{bmatrix} \quad (18)$$

其中，$\alpha_n(t) \in [0,1]$，$m_n(t) \in \{1, 2, \cdots, M\}$。

智能体的目标为最大化长期奖励，而在该问题中，优化目标为最小化系统时延，因此智能体的奖励函数可被定义为系统成本的负相关函数，如式（19）所示：

$$r(t) = -C(t) \quad (19)$$

由于智能体要满足视频处理的准确率要求，因此需要给定准确率奖励，即式（20）：

$$r_{mAP}(t) = \begin{cases} c, & mAP(t) \geqslant a \\ -c, & mAP(t) < a \end{cases} \quad (20)$$

其中，a 表示要求的准确率大小，c 为常数值。

当智能体达到视频处理的准确率要求时，给予 c 的奖励；反之，则给予惩罚。

2.2　基于 LSTM 的服务器队列预测

LSTM 是循环神经网络（Recurrent Neural Network，RNN）的一种，可以充分保留时序信息，解决长期依赖问题[18]。长期依赖问题是指 RNN 在进行长时间序列数据分析时，只能处理上下文较接近的情况，并常常出现梯度消失或梯度爆炸的问题。而 LSTM 可以有效保存长时间序列中的信息以及其中的有效信息，使其不被遗忘。

本文随机采样了连续 400 个时隙，随机策略下的服务器队列长度如图 2 所示，可发现服务器队列数量随着时隙变化而变化，每隔一段时间后会迎来一个高峰。

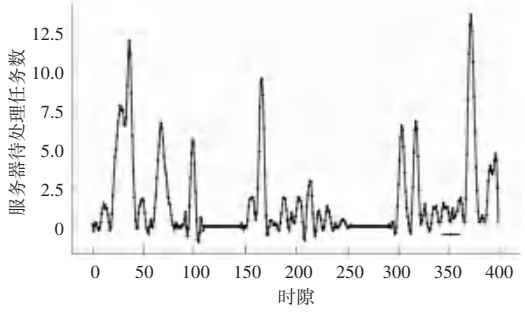

图 2　连续 400 个时隙下的服务器队列长度

Fig. 2　Queue length for servers with 400 time slots

2.3　基于 LSTM 与 TD3 的计算卸载策略

深度强化学习通过智能体与环境交互以优化自身的动作策略。深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法可以解决连续动作控制问题，但该算法会累积误差，使其偏离最优策略[19]。TD3 算法在 DDPG 的基础上提出了双重 Critic 网络以及延迟更新，有效地解决了高偏置问题[20]。

本文在 TD3 的基础之上加入 LSTM 网络以预测服务器队列信息,提出了基于 LSTM 和 TD3 的计算卸载策略(LSTM and TD3 based computational offloading strategy, LSTM-TD3),过程如算法 1 所示。在算法开始时,初始化网络并设置超参数;在每一个时间步中,LSTM 网络会根据当前的服务器状况预测下一时刻的服务器队列数量;智能体将根据环境状态和预测信息做出动作;环境将根据智能体的动作进入下一个状态并给予奖励以及额外奖励;将这一步获得的经验存储至经验回放池中,并根据经验回放池中的信息更新 Critic 网络;当训练达到一定步数时,更新 Actor 网络和目标网络,直到训练结束。

算法 1 **输入** 视频任务信息 $D(t)$,所有队列状态 $Q^{e,\mathrm{RTR}}(t)$, $Q^{e,\mathrm{RTH}}(t)$, $Q^{e,\mathrm{Inf}}(t)$, $Q^{\mathrm{trans}}(t)$, $Q^{s,\mathrm{Inf}}$,带宽信息 $B(t)$

输出 边缘设备卸载决策集合

1 初始化 Critic 网络 Q_{θ_1}、Q_{θ_2} 权重 θ_1、θ_2 以及 Actor 网络 π_ω 权重 ω

2 初始化目标网络 $Q_{\theta_1'}$、$Q_{\theta_2'}$ 权重 $\theta_1' = \theta_1$、$\theta_2' = \theta_2$ 和 Actor 目标网络 $\pi_{\omega'}$ 权重 $\omega' = \omega$

3 初始化经验回放池 B、软更新系数 τ、学习率 α、衰减因子 γ、探索噪声 μ、策略探索噪声 ϵ

4 For each episode do:

5 $s_t \leftarrow s_t$ + LSTM(Q_t^s)

6 Actor 网络根据状态 $S(t)$ 输出动作并给予噪声 $A(t) = \pi_\omega(S(t)) + \mu$

7 执行动作 $A(t)$,获取环境奖励 $R(t)$ 和下一时刻状态 $S(t+1)$

8 将 $(S(t), A(t), R(t), S(t+1))$ 存入经验回放池

9 从经验回放池随机采样 N 个样本 $(S_j(t), A_j(t), R_j(t), S_j(t+1))$, $j = 1, 2, \cdots, N$

10 给目标动作加入噪声 $\widetilde{A_j}(t) = \pi_{\omega'}(S(t)) + \epsilon$, $\epsilon \sim clip(N(0, \sigma), -c, c)$

11 Critic 网络 Q_{θ_1}、Q_{θ_2} 分别计算 Q_1、Q_2,并取最小值作为目标 Q 值 y_j:

$$y_j = R_j(t) + \gamma \min_{i=1,2} Q_{\theta_i'}(S(t+1), \widetilde{a})$$

12 通过均方差损失函数更新 Critic 网络权重:

$$\theta_i = \frac{1}{m}\sum_{j=1}^{m} (y_j - Q_{\theta_i}(S_j(t), A_j(t)))^2$$

13 If t% 延迟更新步长 = 1:

14 通过损失函数更新 Actor 网络权重 ω:

$$J(\theta) = -\frac{1}{m}\sum_{j=1}^{m} Q_{\theta_1}(S_j(t), A_j(t))$$

15 更新 Actor 目标网络, $\omega' \leftarrow \tau\omega + (1-\tau)\omega'$

16 更新 Critic 目标网络, $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$

17 If $S(t+1)$ 为终止状态:

18 End for

3 仿真实验与结果

3.1 参数设置

本文使用 Python 和 Pytorch 进行仿真实验,使用 NVIDIA Jetson Nano B01 (4 GB)作为边缘设备,边缘服务器配备 Intel Core i7-12700KF 和 NVIDIA GeForce GTX 3060。MEC 系统基于 Python 3.7 在 Ubuntu 18.04 LTS 上实现,DRL 算法基于 PyTorch 1.7.0进行训练。在该计算卸载系统中,边缘设备部署的模型准确度在数据集上表现为 0.489,服务器部署的模型准确度为 0.605。在该实验中,将用户所要求的准确率设定为 0.55。每个视频块数据量为 2.4 Mb,假设每个时隙到达的视频块数量为 1-6 个,增加实验的随机性。其余模拟环境参数见表 1。

表 1 视频计算卸载模拟环境参数

Table 1 Video computing offload simulation environment parameters

参数	值
系统时间步长 T	50
系统时隙间隔 Δ/Sec	20
单位视频块数据量 $B_n(t)$/Mb, $\forall n \in N, \forall t \in T$	2.4
单位视频块数量 $D_n(t)$/Sec, $\forall n \in N, \forall t \in T$	1 ~ 6
单位视频块 RTR 转码所需计算资源 $C^{e,\mathrm{RTR}}$/Cycle	3.6×10^9
单位视频块 RTF 转码所需计算资源 $C^{e,\mathrm{RTF}}$/Cycle	7.2×10^9
单位视频块推理所需计算资源 C^{Inf}/Cycle	1.3×10^9
边缘设备 CPU 处理频率 f_{cpu}^e/GHz	1.09
边缘设备 GPU 处理频率 f_{gpu}^e/MHz	900
MEC 服务器 GPU 处理频率 f_{gpu}^s/MHz	13 000

本文将 Actor-Critic 网络学习率(Learning rate, lr)设置为10^{-4},均含有一层 512 个神经元的隐藏层。

网络参数采用 Adam 优化器更新，更新参数时批处理大小为 256。其余参数见表 2。

表 2　算法参数

Table 2　Parameters of the algorithm

参数	值
网络学习率	10^{-4}
隐藏层神经元	512
批处理大小	256
经验回放池	10 000
衰减因子	0.990
探索噪声	0.200
策略噪声	0.100
目标网络软更新程度	0.005

3.2　结果分析

首先对算法的收敛性进行了探索和实验。训练数据为 450 组网络带宽序列，验证集为 50 组网络带宽序列。算法在不同网络学习率下的收敛曲线图如图 3 所示。可以看出，当学习率为10^{-2}与10^{-6}时，算法无法收敛；当学习率为10^{-4}和10^{-3}时，相较于10^{-5}，其收敛速度更快且效果较好；学习率为10^{-3}，在训练后期奖励曲线缓慢下降，出现过拟合现象。因此，在后续实验中选择了10^{-4}作为算法的学习率。

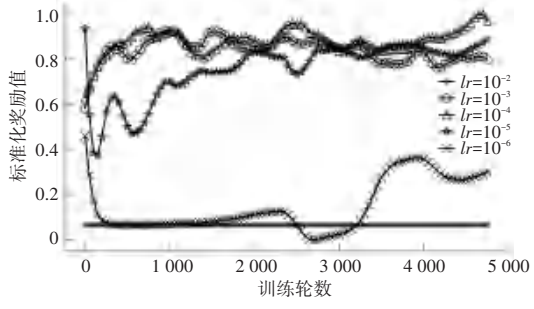

图 3　不同学习率下算法收敛图

Fig. 3　Convergence graph of the algorithm at different learning rates

为评估所提算法的有效性，本文设置如下 5 种卸载策略作为基线策略，并将所提卸载策略与其进行了比较：全本地处理策略（All Local）、全卸载处理策略（All Offload）、随机卸载策略（Random）、基于 DDPG 的卸载策略（DDPG）、基于 TD3 的卸载策略（TD3）。

将系统准确度定义为边缘设备的平均准确度，如式（21）所示：

$$mAP_{\text{sys}} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{n=1}^{N} mAP_n(t) \qquad (21)$$

在不同规模的卸载系统下，每种卸载策略的系统准确度见表 3 和表 4，N 表示边缘设备数量，S 表示边缘服务器数量。不难看出，全本地处理卸载策略所得到的系统准确度最小，全卸载策略获得的系统准确率最大，分别代表系统准确度的上限和下限。基于 DRL 的卸载策略的系统准确度虽不如全卸载策略，但几乎都能高于规定的系统准确度。而本地策略和随机卸载策略，均达不到系统准确度要求。

表 3　不同设备数量（N=5/10）下系统准确度对比

Table 3　Comparison of system accuracy with different number of devices（N=5/10）

卸载策略	系统准确度					
	N5S1	N5S2	N5S3	N10S1	N10S2	N10S3
LSTM-TD3（本文）	0.574	0.569	0.572	0.574	0.573	0.570
TD3	0.563	0.567	0.568	0.566	0.580	0.581
DDPG	0.558	0.572	0.568	0.550	0.595	0.578
Random	0.547	0.546	0.547	0.546	0.548	0.547
All Local	0.489	0.489	0.489	0.489	0.489	0.489
All Offload	0.605	0.605	0.605	0.605	0.605	0.605

表 4　不同设备数量（N=20/25）下系统准确度对比

Table 4　Comparison of system accuracy with different number of devices（N=20/25）

卸载策略	系统准确度					
	N20S1	N20S2	N20S3	N25S1	N25S2	N25S3
LSTM-TD3（本文）	0.552	0.550	0.546	0.552	0.575	0.573
TD3	0.544	0.565	0.562	0.526	0.562	0.560
DDPG	0.549	0.587	0.578	0.518	0.586	0.573
Random	0.546	0.547	0.547	0.546	0.547	0.547
All Local	0.489	0.489	0.489	0.489	0.489	0.489
All Offload	0.605	0.605	0.605	0.605	0.605	0.605

不同规模的卸载系统下，不同卸载策略下系统时延大小如图 4～图 7 所示。可见，随着边缘设备的增多，系统时延也随之增大。如图 5～图 7 所示，随机策略下的系统时延相对更短，但其无法保证系统准确度的要求。而在 DRL 算法中，为了保证系统准确度，智能体会选择牺牲部分系统时延去达到系统准确度的要求。相较于 DDPG、TD3 策略，本文所提的卸载策略能够明显降低系统时延，尤其是在服务器数量增多，环境更复杂的情况下。当服务器数量大于 1 时，所提卸载策略下的系统时延明显降低。在图 5 和图 6 中，所提策略相较于 DDPG 和 TD3，系统时延几乎都有所下降，当 S = 2 和 S = 3 时，所提策略下的系统时延下降了约 10%～40%左右，说明所

提卸载策略能够有效感知周围复杂环境,做出较优决策,更适用于复杂的计算卸载环境。

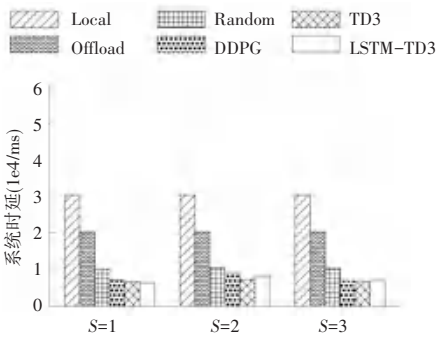

图 4 不同卸载策略下的系统时延成本($N=5$)

Fig. 4 System cost with different offloading strategies ($N=5$)

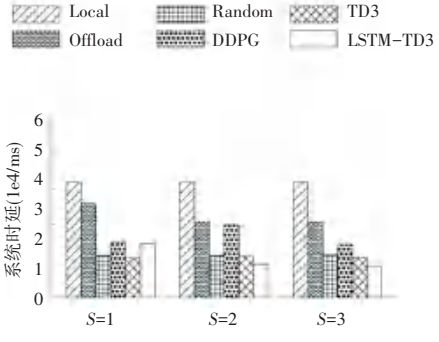

图 5 不同卸载策略下的系统时延成本($N=10$)

Fig. 5 System cost with different offloading strategies ($N=10$)

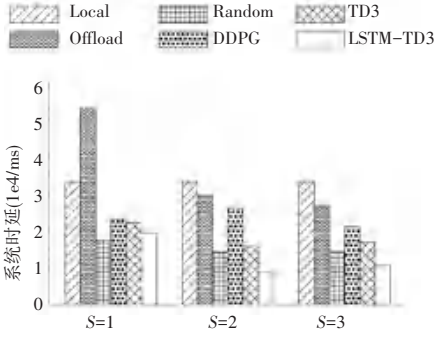

图 6 不同卸载策略下的系统时延成本($N=20$)

Fig. 6 System cost with different offloading strategies ($N=20$)

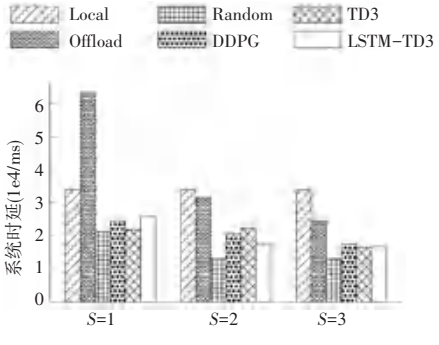

图 7 不同卸载策略下的系统时延成本($N=25$)

Fig. 7 System cost with different offloading strategies ($N=25$)

综上,所提的 LSTM-TD3 卸载策略在较小规模($N < 10$)的视频计算卸载系统下,其系统时延明显小于其他基本策略,与随机策略相比,系统时延约减小了 26%~32%左右。但在较大规模的视频计算卸载系统下,为满足系统准确率要求,系统时延会有一定程度地增加。与其他基于 DRL 的卸载策略相比,所提算法的系统时延约减少了 14%~34%左右。在某些情况下,如 $N = 20, S = 2$ 时,其时延约减少了 47%。说明本文所提出的卸载算法能够在满足系统准确率的要求下,节省系统时延和成本,以加速系统处理。

4 结束语

本文考虑动态网络环境以及视频卸载场景中准确度要求,提出了一个基于深度强化学习的视频计算卸载策略方法。基于真实的视频计算卸载场景进行建模,并将其转换成马尔可夫决策过程模型。考虑系统准确度要求,设置准确度奖励以引导智能体探索较优卸载策略,同时提出了一个基于深度强化学习的卸载方案,使 MEC 系统在满足系统准确度的要求下最小化系统成本。在 TD3 算法基础上,加入了 LSTM 网络预测服务器处理队列的未来拥塞状况,以避免智能体不断向服务器卸载任务造成系统拥堵。仿真实验表明,所提算法能有效优化卸载策略,节省系统时延,与其他卸载策略相比,能够在保证准确率的前提下最大程度地节省系统时延。

参考文献

[1] JIANG F, DONG L, WANG K, et al. Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration[J]. IEEE Internet of Things Journal, 2021, 9(9): 6597-6610.

[2] LU H, GU C, LUO F, et al. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning[J]. Future Generation Computer Systems, 2020, 102: 847-861.

[3] 黄凯奇, 陈晓棠, 康运锋, 等. 智能视频监控技术综述[J]. 计算机学报, 2015, 38(6):1093-1118.

[4] ANJUM A, ABDULLAH T, TARIQ M F, et al. Video stream analysis in clouds: An object detection and classification framework for high performance video analytics[J]. IEEE Transactions on Cloud Computing, 2016, 7(4): 1152-1167.

[5] 段续庭, 周宇康, 田大新, 等. 深度学习在自动驾驶领域应用综述[J]. 无人系统技术, 2021, 4(6): 1-27.

[6] 张开元, 桂小林, 任德旺, 等. 移动边缘网络中计算迁移与内容缓存研究综述[J]. 软件学报, 2019, 30(8): 2491-2516.

[7] 胡恒, 金凤林, 郎思琪. 移动边缘计算环境中的计算卸载技术研究综述[J]. 计算机工程与应用, 2021, 57(14): 60-74.

[8] PHAM Q V, FANG F, HA V N, et al. A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology

integration，and state-of-the-art[J]. IEEE Access，2020，8：116974-117017.

[9] RAN X，CHEN H，ZHU X，et al. Deepdecision：A mobile deep learning framework for edge video analytics[C]//Proceedings of 2018 IEEE Conference on Computer Communications. IEEE，2018：1421-1429.

[10]CHEN J，LI K，DENG Q，et al. Distributed deep learning model for intelligent video surveillance systems with edge computing[J]. IEEE Transactions on Industrial Informatics，2019：1-1. DOI：10.1109/TII.2019.2909473.

[11] TRAN T X，POMPILI D. Adaptive bitrate video caching and processing in mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing，2018，18(9)：1965-1978.

[12] WANG D，PENG Y，MA X，et al. Adaptive wireless video streaming based on edge computing：Opportunities and approaches[J]. IEEE Transactions on Services Computing，2018，12(5)：685-697.

[13]LIU M，YU F R，TENG Y，et al. Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing[J]. IEEE Transactions on Wireless Communications，2018，18(1)：695-708.

[14]ZHOU P，XIE Y，NIU B，et al. QoE-aware 3D video streaming via deep reinforcement learning in software defined networking enabled mobile edge computing[J]. IEEE Transactions on Network Science and Engineering，2020，8(1)：419-433.

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

[16]邵延华，张铎，楚红雨，等. 基于深度学习的 YOLO 目标检测综述[J]. 电子与信息学报，2022，44(10)：3697-3708.

[17]KAELBLING L P，LITTMAN M L，MOORE A W. Reinforcement learning：A survey[J]. Journal of Artificial Intelligence Research，1996，4：237-285.

[18]HOCHREITER S，SCHMIDHUBER J. Long short-term memory[J]. Neural Computation，1997，9(8)：1735-1780.

[19]LILLICRAP T P，HUNT J J，PRITZEL A，et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv：1509.02971，2015.

[20] FUJIMOTO S，HOOF H，MEGER D. Addressing function approximation error in actor-critic methods[C]// Proceedings of International Conference on Machine Learning. PMLR，2018：1587-1596.