文章编号：2095-2163(2024)02-0118-06 中图分类号：TP183;TN929.5 文献标志码：A

基于深度强化学习的边缘视频任务卸载策略

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

摘 要：随着物联网的发展以及智能设备的普及，视频处理技术已广泛应用于生活中。自动驾驶、产品质检等应用场景对视频处理技术的实时性需求逐步提高，移动边缘计算为计算能力不足和能源受限的设备提供计算资源以执行时延敏感性任务，为实时视频处理提供了新的计算架构。本文搭建了一个视频计算卸载场景，并以视频检测为任务，以系统时延为优化目标，建立了计算卸载模型和马尔可夫决策模型；考虑到计算卸载场景的复杂动态因素，如带宽波动、设备数量、任务大小等，以最小化系统时延为目标，提出了一种基于深度强化学习的计算卸载策略进行求解。实验表明，与其他基线方案相比，该卸载策略能够适应较复杂卸载场景，有效降低系统时延。

关键词：移动边缘计算；实时视频处理；深度强化学习

Edge video offloading strategy based on deep reinforcement learning

ZHOU Chenjing, LUO Shuyun
(School of Computer Science and Technology(School of Artificial Intelligence), Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: With the development of the Internet of Things and the proliferation of smart devices, video processing technology has been widely applied in daily life. Applications such as autonomous driving and product quality inspection increasingly demand real-time video processing. Mobile edge computing provides computing resources for devices with limited computational power and energy, supporting time-sensitive tasks and offering a new computational architecture for real-time video processing. This paper constructs a video computation offloading scenario focused on video detection tasks, with system latency as the optimization objective. It establishes a computation offloading model and a Markov decision model. Considering the complex dynamic factors of the computation offloading scenario, such as bandwidth fluctuations, number of devices, and task size, a computation offloading strategy based on deep reinforcement learning is proposed to minimize system latency. Experiments show that this offloading strategy adapts well to complex offloading scenarios and effectively reduces system latency.

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

0 引 言

视频检测与分析技术已经广泛应用于生产生活之中，如自动驾驶、监控异常检测、产品质检、虚拟现实/增强现实(Virtual Reality/Augmented Reality, VR/AR)等，产生了大量的视频数据以及视频任务，对视频任务的实时性要求也随之提高。随着移动设备、摄像头等终端设备的普及，现有的云计算逐渐无法应对数以亿计的数据任务，产生了如网络拥塞、隐私安全等问题。移动边缘计算的出现为解决这些问题提出了一个方案。

移动边缘计算(Mobile Edge Computing, MEC)是指计算能力受限的设备将任务卸载至边缘端，以加速任务处理速度的一种计算架构，具有低延迟、安全性高的特点，MEC 逐渐成为物联网应用的主要趋势[1]。视频边缘计算架构为实时视频任务提供了解决方案，有研究在边缘计算环境中部署深度学习(Deep Learning, DL)模型，用于视频检测任务。Ran 等[2]提出了分布式边缘视频分析系统 DeepDecision，将模型部署在边缘设备中，权衡模型精度、网络条件、视频分辨率等各类因素做出卸载决策，在单个 AR 应用程序中表现良好；Chen 等[3]则提出了一种基于分布式 DL 模型的智能视频监控系统，将系统部署在边缘计算环境中，在每个边缘节点上部署多个不同结构

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：2023-02-21

的 DL 子模型，并行执行不同的数据分析任务，并且提出了一种动态数据迁移方法来保证系统的工作负载平衡。但以上研究无法扩展至较大规模的视频计算卸载环境中，且其模型未考虑具体操作，如视频前期转码处理等，实时性无法保证。

现有的计算卸载解决方案主要是启发式的，难以适应复杂和动态的应用[4]。近年来，深度强化学习（Deep Reinforcement Learning，DRL）以其在环境中试错学习的特点能够较好的适应各类复杂的动态场景，许多研究尝试用 DRL 来获取卸载决策。Huang 等[5]提出了基于深度强化学习的在线卸载策略（Deep Reinforcement Learning for Online Computation Offloading，DROO）来解决计算卸载问题，该算法基于 DRL，根据时变的无线信道条件优化任务卸载决策和无线资源分配；Tang 等[6]则考虑了不可分割和延迟敏感型任务，以最小化预期的长期成本，提出了一种基于深度强化学习的分布式算法，每个设备可以在不知道其他设备的任务模型和卸载决策的情况下确定其卸载策略；Cai 等[7]提出了一种基于 DRL 的多任务多设备混合计算卸载模型，考虑了多个任务直接的交互，并引入了长短期记忆网络（Long Short-Term Memory，LSTM）来提取任务和网络状态的特征，对多个任务进行全局卸载决策。基于各类特定计算卸载场景提出的各类卸载策略，能够显著降低系统成本，但无法较好地适用于视频计算卸载场景。因此，对于本文所提的视频计算卸载场景，需要重构模型并重新训练算法。

本文针对视频计算卸载系统，考虑视频分析处理的具体过程进行系统建模。为了保证实时性，并适应高度动态的网络波动，本文将原问题转换成马尔科夫决策过程，采用了双延迟深度确定性策略梯度（Twin Delayed Deep Deterministic policy gradient，TD3）算法优化卸载策略。仿真结果显示，本文基于深度强化学习的视频计算卸载策略相较于其他卸载策略，能够有效降低系统时延。

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。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。终端设备生成视频数据并传送视频任务至 EDs，EDs 接收终端待处理视频任务并将其分割成多个时长相同的视频块；根据卸载策略，EDs 将部分视频块卸载至 ESs 进行处理，剩余视频块将在本地进行处理；最终 EDs 将结果发送至 ESs，将结果存储至服务器，方便用户调取。

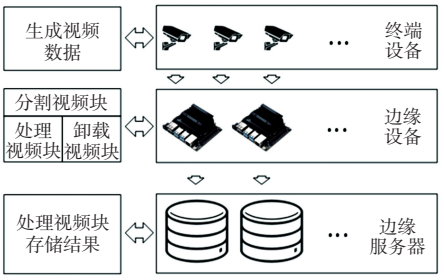

图 1　视频边缘计算系统

Fig. 1　Edge computing system for video

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，计算公式见式（1）：

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

其中，$\lceil\cdot\rceil$ 表示向上取整。

本地处理视频块数量，计算公式见式（2）：

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

以视频检测任务为例，视频在本地处理时，需要对视频进行转码操作，将其格式由 RAW 转换成可进行模型推理的 RGB 格式，再对视频进行模型推理，得到检测结果。而将视频卸载处理时，首先需将视频转换为流媒体格式，以视频流方式进行传输。当视频到达边缘服务器后，则边缘服务器需将其转换为 RGB 格式，再对视频进行模型推理。

1.1　本地处理模型

在本地进行处理的视频块，需要编解码，将格式转换为 RGB 格式，再进行视频推理。为了使视频块有序处理，本文设置队列来存放待处理的视频块。在时隙 0 时，边缘设备生成两个队列分别存放需要编解码处理的视频块和视频推理的视频块。

在时隙 t 时，编解码队列和视频推理队列长度分别为 $Q_n^{e,\mathrm{RTR}}(t)$ 和 $Q_n^{e,\mathrm{Inf}}(t)$，此时对于边缘设备 n 在本地处理的第 i 个视频块，其编解码时延，计算公式见式（3）：

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

其中，f_{cpu}^e 表示边缘设备的 CPU 频率，$C^{e,\mathrm{RTR}}$ 表示边缘设备将单个视频块转换为 RGB 格式所消耗的 CPU 资源。

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,需在编解码完成后,并且推理队列中排在其之前的任务完成推理后,才能够开始推理。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,计算公式见式(4):

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

其中,f_{gpu}^{e} 表示边缘设备视频推理处理频率,C^{Inf} 为完成视频检测任务所需的 CPU 资源。

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,计算公式见式(5):

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

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和传输队列,边缘服务器生成视频推理队列。

在时隙 t 时,编解码队列和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,计算公式见式(6):

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

其中,$C^{e,\mathrm{RTF}}$ 为单个视频块进行视频块转码操作所需的 CPU 资源。

视频块完成视频编解码后,进入至视频传输队列。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,计算公式见式(7):

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

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,计算公式见式(8):

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

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,其第 i 个视频块的传输时延,计算公式见式(9):

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

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,计算公式见式(10)

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

视频块传输至服务器后,在服务器中需进行编解码操作转换成 RGB 格式化,才可进行视频推理。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 $Q_m^{s,\mathrm{Inf}}(t)$,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):

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

其中,f_{gpu}^{s} 表示边缘服务器视频推理处理频率。

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

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

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,计算公式见式(13):

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

综上,任务 $V_n(t)$ 的总处理时延,计算公式见式(14):

$$L_n(t)=\max\left(L_n^l(t),L_m^s(t)\right) \quad (14)$$

整个系统的处理时延可表示为式(15):

$$L(t)=\max\left(L_1(t),L_2(t),\cdots,L_N(t)\right) \quad (15)$$

1.3 问题模型

根据时延模型,整个系统成本可表示为式(16):

$$\text{System cost}=\max(L(t)) \quad (16)$$

为了保证实时性,应使系统时延越小。因此该系统优化问题可表述为 $\arg\min_{A(t)}\max(L(t)),t\in$

$\{1,\cdots,T\}$，$0 \leqslant \alpha(t) \leqslant 1$，$m \in \{1,2,\cdots,M\}$。

其中，$A(t)$ 表示各终端设备做出的卸载决策的集合。

2　基于 TD3 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。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) | S(t)，A(t))。

2.1.1　状态空间

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，可以被定义为式(17)：

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

其中，$D(t)$ 表示在时隙 t 时，到达边缘设备的视频块数量集合；$B(t)$ 为当前时隙的网络带宽；$Q^{e,\mathrm{RTR}}(t)$，$Q^{e,\mathrm{RTH}}(t)$，$Q^{e,\mathrm{Inf}}(t)$，$Q^{\mathrm{trans}}(t)$，$Q^{s,\mathrm{Inf}}(t)$ 分别代表边缘设备的 RAW 到 RGB 编解码队列、RAW 到 H.264 的编解码队列、模型推理队列、传输队列和边缘服务器的模型推理队列情况。

2.1.2　动作空间

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)：

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

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

2.1.3　奖励函数

奖励函数可以引导算法趋近优化目标。在该问题中，优化目标为最小化系统时延，因此奖励函数可被定义为系统时延的负相关函数，即式(19)：

$$r(t) = -\max(L(t)) \tag{19}$$

2.2　基于 TD3 的卸载策略

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[8]。但该算法会累积误差，使其偏离最优策略 π，算法最终无法收敛。

TD3 算法有效地解决了高偏置问题[9]，并在 DDPG 的基础上进行了以下改进：

（1）双重网络：为了避免高估，TD3 提出了两个 Q 函数进行学习，使用较小值来更新参数；

（2）策略更新延迟：目标网络和策略网络有延迟更新，且 Actor 网络在 Critic 网络更新多次后再更新，以避免累积误差；

（3）目标策略平滑：为了降低目标网络的误差，将噪声添加到目标动作中，利用目标动作周围的区域来计算目标值，防止过拟合，提高目标策略的准确性以及稳定性。

因此，TD3 算法中含有 6 个网络，即 Critic1 网络、Critic2 网络、Actor 网络、Critic1 目标网络、Critic2 目标网络、Actor 目标网络。为优化边缘设备的卸载策略，智能体将当前时隙下的状态信息 $s(t)$ 输入至 Actor 网络，Actor 网络根据状态信息来输出动作 $A(t)$。智能体与环境交互获得环境奖励以及下一时隙的状态信息 $s(t+1)$，并将状态转移信息 $(S(t), A(t), R(t), S(t+1))$ 存放至经验回放池。在更新 Critic 网络时，从经验回放池中随机抽样一定数量的历史信息，计算目标 Q 值，以梯度下降方式更新 Critic 网络。在 Critic 网络更新多次后，再更新 Actor 网络以及各自的目标网络，避免累计误差。基于 TD3 的计算卸载算法过程如算法 1 所示。

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

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

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 $\omega' = \omega$

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

For each episode do：

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

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

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

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

给目标动作加入噪声 $\tilde{A}_j(t) = \pi_{\omega'}(S(t)) + \epsilon$，

$\epsilon \sim clip(N(0,\sigma), -c, c)$

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：

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

通过均方差损失函数更新 Critic 网络权重：

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

If t% 延迟更新步长=1：

通过损失函数更新 Actor 网络权重 ω：

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

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

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

If $S(t+1)$ 为终止状态：

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，边缘服务器配备了 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。

本文选取了疵点检测作为视频任务，为了方便训练，省去了视频块分割的步骤，并假设视频任务以视频块的形式到达边缘设备。因此，本文设置每个视频块数据量为 2.4 MB，假设每个时隙时到达的视频块数量为 1-6 个，增加实验的随机性。同时，设置系统时间步长 T 为 50，即系统时间包含 50 个时隙，每个时隙间隔 Δ 为 20，即每隔 20 s，到达一批视频块。实验相关参数见表 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/ s	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 $D_n(t)$ /s，$\forall n \in N, \forall t \in T$	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^e_{cpu} /GHz	1.09
边缘设备 GPU 处理频率 f^e_{gpu} /MHz	900
MEC 服务器 GPU 处理频率 f^s_{gpu} /MHz	13 000

本文将 Actor-Critic 网络学习率（Learning Rate，LR）设置为 10^{-3}，均含有一层 256 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 64。经验回放池大小为 1 000，衰减因子 γ = 0.99，探索噪声采用 Ornstein-Uhlenbeck 噪声，策略噪声 ϵ = 0.1，目标网络软更新程度 τ = 0.005。

3.2 结果分析

首先验证算法的收敛能力。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。不同收敛率下算法收敛情况如图 2 所示。

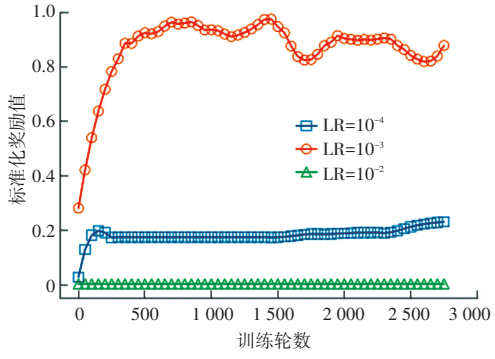

图 2 不同学习率下算法收敛图

Fig. 2 Convergence of the algorithm with different learning rates

从图 2 可见，当网络的学习率为 10^{-2} 时，由于学习率太大，算法不收敛；当网络的学习率为 10^{-4} 时，算法在 1 000 训练轮数之后的学习速度较为缓慢；当算法的学习率为 10^{-3} 时，其收敛效果较好，经过大约 1 000 轮次的训练，即获得了较高的奖励值。

为了评估算法的有效性，本文设置了 4 种基线方案：

（1）全本地执行（All Local）：所有任务在边缘设备本地执行；

（2）全卸载执行（All Offload）：所有任务全部卸载至边缘服务器执行；

（3）随机卸载策略（Random）：卸载随机数量的任务至边缘服务器执行；

（4）基于 DDPG 的计算卸载策略（DDPG）：使用 DDPG 算法来优化计算卸载决策。

不同卸载策略下的系统时延成本对比如图 3~图 6 所示，其中，N 代表边缘设备数量，S 代表边缘服务器数量。可以看出，当边缘设备数量增多时，由于任务数增多，系统时延成本也随之增加。此时若边缘服务器数量不变，全卸载执行会使得任务的传输以及等待时延大大提高。如图 5~图 6 所示，当边缘服务器数量为 1 时，全卸载执行系统时延成本最大。随机卸载策略随机选择卸载率以及服务器进行卸载，相较于全本地执行和全卸载执行，系统成本有

所减少。但由于其未考虑各类环境因素，优化效果不明显。

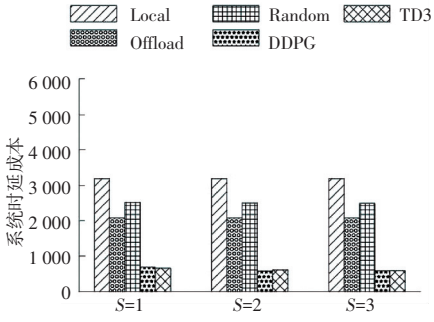

图 3　不同卸载策略下的系统时延成本($N=10$)

Fig. 3　System cost with different offloading strategies ($N=10$)

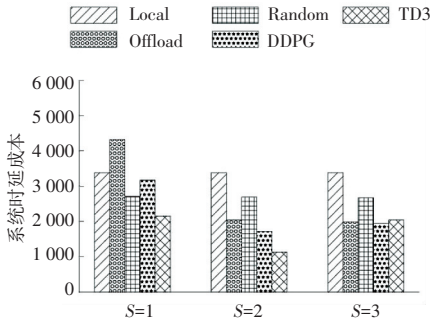

图 4　不同卸载策略下的系统时延成本($N=20$)

Fig. 4　System cost with different offloading strategies ($N=20$)

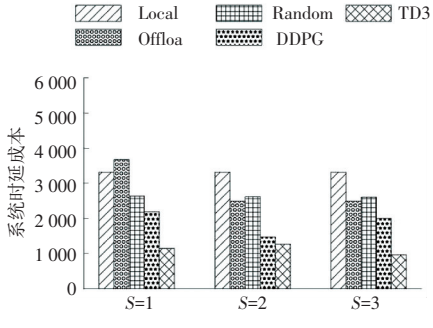

图 5　不同卸载策略下的系统时延成本($N=25$)

Fig. 5　System cost with different offloading strategies ($N=25$)

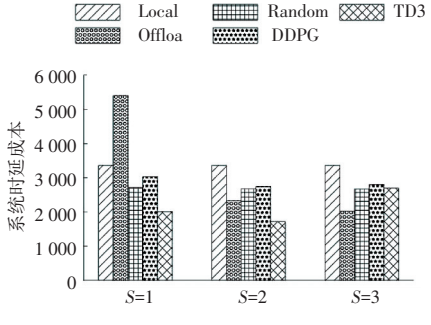

图 6　不同卸载策略下的系统时延成本($N=30$)

Fig. 6　System cost with different offloading strategies ($N=30$)

基于深度强化学习的计算卸载策略从环境中学习，能够适应各类复杂情况。但基于 DDPG 的计算卸载策略在边缘设备和边缘服务器数量增多时，由于本身算法存在局限性，其效果不佳。而基于 TD3 的计算卸载策略相比较于其他卸载策略，其优化效果在大多数情况下都更好，能够适应不同卸载环境，有效减少系统时延成本，加快任务的处理效率。

4　结束语

本文针对视频计算卸载系统，以系统时延为优化目标，建立了计算卸载模型。为了使其适应复杂的计算卸载环境，如服务器数量增多、网络动态波动等情况，将系统模型转换成 MDP 模型，并采用深度强化学习算法进行求解。实验表明，该方法相较于其他基准方法，能够适应带宽波动并有效降低系统时延，同时也能够适应不同计算卸载环境，降低视频处理时延。下一步将考虑能耗以及其他因素对系统的影响，联合优化时延与能耗目标。

参考文献

[1] MALTEZOS E, LIOUPIS P, DADOUKIS A, et al. A video analytics system for person detection combined with edge computing[J]. Computation, 2022, 10(3): 35.

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

[3] 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/T11.2019.2909473.

[4] WANG J, HU J, MIN G, et al. Computation offloading in multi-access edge computing using a deep sequentialmodel based on reinforcement learning [J]. IEEE Communications Magazine, 2019, 57(5): 64-69.

[5] 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.

[6] TANG M, WONG V W S. Deep reinforcement learning for task offloading in mobile edge computing systems [J]. IEEE Transactions on Mobile Computing, 2020, 21(6): 1985-1997.

[7] CAI J, FU H, LIU Y. Deep reinforcement learning - based multitask hybrid computing offloading for multiaccess edge computing[J]. International Journal of Intelligent Systems, 2022, 37(9): 6221-6243.

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

[9] 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.