基于多重门限机制的异步深度强化学习

徐进[1)] 刘全[1),2),3)+] 章宗长[1),2)] 梁斌[1)] 周倩[1)]

[1)](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2)](软件新技术与产业化协同创新中心 南京 210000)

[3)](吉林大学符号计算与知识工程教育部重点实验室 长春 130012)

摘 要 近年来，深度强化学习已经成为人工智能领域一个新的研究热点。深度强化学习在如 Atari 2600 游戏等高维度大状态空间任务中取得了令人瞩目的成功，但仍存在训练时间太长等问题。虽然异步深度强化学习通过利用多线程技术大幅度减少了深度强化学习模型所需的训练时间，但是，基于循环神经网络的异步深度强化学习算法依然需要大量训练时间，原因在于具有记忆能力的循环神经网络无法利用并行化计算加速模型训练过程。为了加速异步深度强化学习模型的训练过程，并且使得网络模型具有记忆能力，本文提出了一种基于多重门限机制的异步优势行动者-评论家算法。该模型主要有三个特点：一是通过使用多重门限机制使前馈神经网络具有记忆能力，使 Agent 能够通过记忆不同时间步的状态信息做出更优的决策；二是通过利用并行计算进一步加速 Agent 的训练过程，减少模型所需的训练时间；三是通过采用一种新的跳跃连接方式实现数据向更深的网络层传递，增强模型识别状态特征的能力，从而提升深度强化学习算法的稳定性和学习效果。本文通过 Atari 2600 游戏平台上的部分战略型游戏以及稀疏奖赏环境型游戏来评估新模型的性能。实验结果表明，与传统的异步深度强化学习算法相比，新模型能够以较少的时间代价来获得更优的学习效果。

关键词 深度学习；强化学习；异步深度强化学习；循环神经网络；多重门限机制；跳跃连接
中图法分类号 TP18

Asynchronous Deep Reinforcement Learning with Multiple Gating Mechanisms

XU Jin[1)] LIU Quan[1),2),3)] ZHANG Zong-Zhang[1),2)] LIANG Bin[1)] ZHOU Qian[1)]

[1)](School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, 215006)

[2)](Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, 210000)

[3)](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, 130012)

Abstract In recent years, deep reinforcement learning (RL), a combination of reinforcement learning and deep learning, has become a new research hotspot in the artificial intelligence community. Deep RL algorithms have achieved unprecedented success in high-dimensional and large-scale space tasks such as Atari 2600, but there are still issues such as the duration of training model parameters too long. Even in the case of the graphics processing unit (GPU) acceleration, deep Q-network, double deep Q-network, deep recurrent Q-network and other deep RL algorithms based on experience replay still need to train ten days or even more. DeepMind's researchers use

本课题得到国家自然科学基金项目(61272055, 61303108, 61373094, 61472262, 61502323, 61502329, 61772355)、江苏省自然科学基金（BK2012616）、江苏省高校自然科学研究项目(13KJB520020, 16KJB520041)、吉林大学符号计算与知识工程教育部重点实验室基金项目（93K172014K04, 93K172017K18）、苏州市应用基础研究计划工业部分（SYG201422, SYG201308）资助. 徐进，男，1992 年生，硕士研究生，主要研究方向为深度强化学习. E-mail：20154227016@stu.suda.edu.cn. 刘全（通讯作者），男，1969 年生，博士，教授，博士生导师，中国计算机学会高级会员，主要研究领域为智能信息助理、自动推理和机器学习. E-mail: quanliu@suda.edu.cn. 章宗长，男, 1985 年生，博士，副教授，中国计算机学会会员，主要研究领域为部分可观测的马尔科夫决策过程、强化学习和多智能体系统. E-mail: zzzhang@suda.edu.cn. 梁斌，男，1993 年生，硕士研究生，主要研究方向为情感分析、自然语言处理和深度学习. E-mail: bliang@stu.suda.edu.cn. 周倩，女，1992 年生，硕士研究生，主要研究方向为强化学习. E-mail：20154227029@stu.suda.edu.cn.

asynchronous methods to replace deep RL's experience replay, accelerating deep RL models training greatly. Asynchronous deep reinforcement learning (ADRL) algorithms can take a few hours or a few days to achieve better learning effect on a single machine with a standard multi-core central processing unit by using multi-threading techniques and are not necessary for the GPU. Although ADRL significantly reduces the training time required for deep RL models by using multi-threading techniques, the ADRL algorithms based on recurrent neural network (RNN) require lots of training time yet. It is due to RNN is unable to accelerate model training process with parallel computation. The calculation of the current time step of RNN depends on the output of the previous time step, so given a set of sequentially generated input sequences, the RNN can only compute each input sequentially, and is incapable to be calculated by parallel computation. In order to accelerate the training process of ADRL models and make deep neural networks with the ability of memory, we propose the asynchronous advantage actor-critic algorithm with multiple gating mechanisms (A3C-MGM). There are three main characteristics in the algorithm: First, similar to long short-term memory, the gates of the multiple gating mechanisms control the information passed in the hierarchy; feedforward neural networks have the ability of memory by using multiple gating mechanisms, which allows the agent to make better decision by memorizing the state information of different time steps. Second, the calculation of the current time step of A3C-MGM does not depend on the output of the previous time step, and A3C-MGM is able to accelerate the training process of the agent via parallel computation, further reducing the training time required for the model. Third, a new skip connection is utilized in our work in order to improve the stability and performance of the deep RL algorithm by passing data to deeper network layers. We evaluate the performance of the new algorithm through five challenging strategic games with sparse reward from the set of classic Atari 2600 games, i.e., Battle Zone, Tutankhum, Time Pilot, Space Invaders, and Pong. In order to further verify the performance of the deep RL algorithm, we also compare the other four Atari 2600 games. Experimental results show that the new algorithm can learn faster than traditional asynchronous deep reinforcement learning algorithms, and achieve better performance in terms of the average reward per episode especially on the Battle Zone, Tutankhum, Asterix, Chopper Command, Solaris, and Time Pilot games.

Keywords deep learning; reinforcement learning; asynchronous deep reinforcement learning; recurrent neural network; multiple gating mechanisms; skip connection

1 引言

深度强化学习（Deep Reinforcement Learning, DRL）结合了深度学习[1]（Deep Learning）和强化学习[2]（Reinforcement Learning, RL），是人工智能领域的一个新的研究热点。通过利用深度神经网络对于高维状态空间的有效识别能力，能使得强化学习算法在复杂状态任务中更加有效。目前强化学习在仿真模拟、游戏博弈和工业控制等领域[3-5]取得了显著的成果，但是在复杂的高维状态空间任务中，传统的强化学习算法的表现并不好且数据需要经过复杂的人工预处理。为了解决强化学习在高维度状态空间任务中需要复杂的人工预处理问题，Mnih 等人[6-7]首次把深度神经网络中的卷积神经网络（Convolutional Neural Network, CNN）与强化学习中的 Q 学习算法[8]相结合，提出了一种深度 Q 网络模型（Deep Q-Network, DQN）。DQN 模型在 Atari 2600 平台的大部分游戏中均表现出超越了人类玩家的水平，然而其在学习过程中容易出现过度乐观估计动作值的问题。针对此问题，Van Hasselt 等人[9]结合了双重 Q 学习（Double Q-Learning）算法，提出了一种双重深度 Q 网络模型（Double Deep Q-Network, DDQN）。DDQN 模型在计算回报值时，其动作值计算和动作选择使用的网络模型参数是两套不同的网络参数。通过这种方式计算损失，DDQN 成功地缓解了 DQN 容易出现过度乐观估计动作值的问题。DQN 和 DDQN 模型都引入了经验重放机制，通过把历史样本存放到经验重放缓冲池，在模型训练过程中以等概率采样方式从经验重

放缓冲池中选择一组小批量（mini-batch）的样本来计算损失并利用梯度下降方法更新模型参数。由于不同训练样本的重要程度是不一样的，而 DQN 和 DDQN 采用的是等概率方式采样，所以 DQN 和 DDQN 无法有效利用对于模型训练更有用的样本。针对此问题，Schaul 等人[10]提出了一种基于优先级重放采样的深度强化学习（Deep Reinforcement Learning with Prioritized Experience Replay）算法。其思想是利用立即奖赏、时间差分（Temporal Difference, TD）等信息作为经验重放缓冲池中的样本优先级评判标准，为不同的样本赋予相应的优先级，优先级高的样本对应高的采样概率，以非等概率方式随机采样经验重放缓冲池中的样本。DQN、DDQN 等 DRL 算法均可通过优先级采样方式提升模型训练效果。

虽然上述基于 CNN 的深度强化学习算法在 Atari 2600 平台的大部分游戏中都取得了超越人类玩家水平的成绩，但是由于任务状态信息在 CNN 中仅仅是简单的前向流通，CNN 无法记忆不同状态之间的依赖性信息，所以这类算法在战略性任务中（任务状态在不同的时间尺度存在依赖性）的表现都比较差。Hochreiter 和 Schmidhuber[11]提出的长短期记忆（Long Short-Term Memory, LSTM）网络和 cho 等人[12]提出的门限循环神经网络（Gated Recurrent Unit, GRU），能成功地解决战略性任务中任务状态在不同时间尺度的依赖性问题。LSTM 和 GRU 网络都是通过不同的门来控制信息的流通，从而达到记忆不同时间尺度信息的效果。Narasimhan 等人[13]基于 LSTM 提出的深度循环 Q 网络（Deep Recurrent Q-Network, DRQN）在一类文本游戏上表现良好，Hausknecht 等人[14]针对部分可观测马尔科夫决策过程（Partially Observable MDPs, POMDPs）问题，提出了一种深度循环 Q 学习算法，并且在 Atari 2600 游戏中取得了良好的效果。由于 LSTM 网络的计算与上一个时间步记忆细胞的输出信息和上一个时间步输出门的输出信息均有关，GRU 网络的计算与上一个时间步输出门的输出信息有关；因此只有获得这些输出信息，才能通过网络模型计算当前状态。因此 DRQN 无法通过并行化计算加速模型训练，这使得 DRQN 模型的训练时间相当长。

虽然 DQN、DDQN、DRQN 等基于经验重放机制的 DRL 算法在一些高维度状态空间任务中取得了巨大的成功，如 Atari 2600 游戏等。但是经验重放机制存在一些固有的局限性：

（1）使用经验重放机制的前提是需要大量的存储空间来存储训练样本，使得这类 DRL 算法对于存储空间的需求显著增加。

（2）经验重放机制要求智能体（Agent）在与环境的每一次交互后都需要通过重放一定批量的训练样本来计算损失并且更新网络模型参数，这导致计算量非常大，对于硬件要求很高，需要利用专门的硬件加速，如图形处理器（Graphics Processing Unit, GPU）。

（3）经验重放机制还必须使用异策略（Off-Policy）的 RL 算法，如 Q 学习算法；且不能使用同策略（On-Policy）的 RL 算法。因此，同策略算法无法利用经验重放机制的优势来提升 Agent 的学习效果，如 Sarsa 算法[15-16]。

由于这些问题是经验重放机制固有的问题，所以无法通过改进经验重放机制来解决。针对此问题，Mnih 等人[17]把异步方法与深度强化学习相结合，提出了异步深度强化学习（Asynchronous Deep Reinforcement Learning, ADRL）。ADRL 通过使用异步方法来替代传统的经验重放机制，使得 DRL 算法不再需要存储大量的训练样本，节省了存储资源的开销。Agent 并不是在每一次与环境交互后都更新网络模型的参数，而是在与环境多次交互之后计算累积损失，并利用梯度下降方法更新参数，降低了 DRL 算法的计算代价。传统的 DRL 算法由于其计算量巨大，严重依赖于 GPU 硬件。而 ADRL 算法对于 GPU 不是必须的，可以在一个多核心的 CPU（Central Processing Unit）设备上利用多线程技术加速 DRL 模型的训练。与传统的 DRL 算法相比，ADRL 算法可以在一个较短的时间内获得更好的训练效果。

基于卷积神经网络的异步深度强化学习（Asynchronous Deep Reinforcement Learning with Convolutional Neural Network, ADRL-CNN）算法可以通过利用多线程技术加速模型训练，大幅度减少模型的训练时间。ADRL-CNN 的相关实验表明，其在 Atari 2600 的部分游戏中有不错的性能表现，但是和 DQN、DDQN 等前馈 DRL 算法一样，ADRL-CNN 在战略性任务上的表现并不好。针对此问题，Mnih 等人[17]结合 LSTM 网络和 GRU 网络，提出了一种异步深度循环强化学习（Asynchronous Deep Recurrent Reinforcement Learning, ADRRL）算法，ADRRL 算法通过利用 LSTM 网络或者 GRU 网络的各种门单元来记忆战略性任务中不同时间尺

度的状态信息之间的依赖性，成功地解决了ADRL-CNN 算法在战略性任务中由于无法记忆不同时间尺度的状态信息之间的依赖性而导致的学习效果差的问题，进一步提高了 Agent 在战略性任务上的性能表现。虽然利用循环神经网络可以提升Agent在战略性任务中的性能表现，但是由于LSTM网络和 GRU 网络当前时间步的计算依赖于上一个时间步所获得的计算结果，ADRRL 算法无法通过并行化计算来进一步加速模型的训练过程。因此，ADRRL 算法是以更多的时间代价来获得更好的训练效果。

虽然使用异步方法的 DRL 算法可以利用多线程技术大大加速模型的训练过程，但是对于像使用了循环神经网络的 ADRL 算法，依然需要大量的训练时间。现代计算设备大多支持并行化计算，为了充分利用这一硬件优势，使用并行化计算进一步加速模型的训练过程，本文提出了一种基于多重门限机制的异步优势行动者-评论家算法（Asynchronous Advantage Actor-Critic with Multiple Gating Mechanisms, A3C-MGM）。该算法的主要改进有：1、在 ADRL 中使用多重门限机制替代循环神经网络，多重门限机制不同时间步之间的计算不存在依赖关系，可以使得模型在训练过程中利用现代计算设备能够并行化计算这一优势，进一步加速模型的训练过程；2、多重门限机制通过门单元来记忆不同时间尺度的状态信息之间的依赖性，使得模型的训练效果比使用卷积神经网络和循环神经网络的 ADRL 算法更好。此外，本文还在 A3C-MGM 算法模型的基础上提出了一种新的跳跃连接方式，通过跳跃连接使得数据向更深的网络层传递，从而让网络模型更加有效地识别状态的特征，进一步提升算法的稳定性。实验结果表明，在 Atari 2600 游戏中，由于 A3C-MGM 算法比基于循环神经网络的 ADRL 算法模型结构更加简单，训练速度得到了大幅度提升；A3C-MGM 算法有效提升了 Agent 的性能表现；并且在采用跳跃连接后，进一步提升了基于多重门限机制算法模型的训练效果和稳定性。

2 相关工作

2.1 强化学习

强化学习是一种交互式学习，Agent 与环境直接进行交互，利用环境的反馈信息进行自我学习。强化学习问题可以利用马尔科夫决策过程（Markov Decision Process, MDP）来进行建模。MDP 问题可以定义为一个四元组$\langle S, A, \rho, f\rangle$。

（1）S 表示环境的状态空间的集合，$s_t \in S$ 表示在时间步 t，Agent 所处的状态。

（2）A 表示动作空间集合，$a_t \in A$ 表示在时间步 t，Agent 所选择的动作。

（3）$\rho: S \times A \to \mathbb{R}$ 表示奖赏函数，$r(s_t, a_t)$ 表示在时间步 t，Agent 在状态 s_t 下根据给定的策略采取动作 a_t 所获得的立即奖赏。

（4）$f: S \times A \times S \to [0,1]$ 表示状态转移函数，$f(s_t, a_t, s_{t+1})$ 表示 Agent 在状态 s_t 根据给定的策略采取动作 a_t 时转移到状态 s_{t+1} 的概率。

标准的强化学习设定是：通过给定离散时间步 t，Agent 在状态 s_t 根据一个给定的策略 π 选择动作 a_t，获得从时间步 t 开始的累积折扣的未来奖赏，即在状态 s_t，Agent 所获得的期望回报 $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$，其中 $\gamma \in [0,1]$ 表示折扣因子。Agent 的最终目标是最大化在状态 s_t 所获得的期望回报 R_t，即获取一个最优的策略。定义状态-动作对的值 $Q^\pi(s_t, a_t)$ 为 Agent 在状态 s_t 根据给定的策略 π 选择动作 a_t 所获得的期望回报。相应的计算公式如式（1）所示：

$$Q^\pi(s,a) = \mathrm{E}[R_t \mid s_t = s, a_t = a] \tag{1}$$

值 $V^\pi(s_t)$ 是 Agent 在状态 s_t 根据给定的策略 π 所获得的期望回报，其计算公式如式（2）所示：

$$V^\pi(s) = \mathrm{E}[R_t \mid s_t = s] \tag{2}$$

最优的 Q 值是对于给定的状态 s 和动作 a，策略 π 可以获得最大的 Q 值，如式（3）所示：

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{3}$$

对于大规模状态空间，通过学习一个参数化的状态-动作对的值函数 $Q(s,a;\boldsymbol{\theta})$ 来训练 Agent，其中 $\boldsymbol{\theta}$ 为状态-动作对的值函数的参数。

强化学习方法可以分为基于模型的强化学习方法和模型无关的强化学习方法。模型无关的强化学习方法又可以分为动作值拟合方法和策略梯度方法。动作值拟合方法有异策略的 Q 学习算法、同策略的 Sarsa 算法等；策略梯度方法直接通过在策略梯度的方向上更新参数来改进策略[18-20]，主要有行动者-评论家[21]（Actor-Critic, AC）算法等。

AC 算法通过一个独立的存储结构来表明策略

与值函数是独立的，算法结构图如图 1 所示。AC 算法分为两个部分：

（1）行动者部分，即策略，用来在当前状态根据策略 π 采取一个动作，使环境迁移到下一个状态；

（2）评论家部分，即值函数，评论家部分采用 TD 误差来评论当前行动者所采取的动作的好坏。

TD 误差表示为当前状态的 1 步回报或者 n 步回报与当前状态的值函数之间的差值，TD 误差如式（4）所示：

$$\delta_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}) - V(s_t) \tag{4}$$

其中 δ_t 表示 TD 误差，$n=1$时表示的是 1 步回报，$n=k$ 时表示的是 k 步回报，r_{t+i} 表示在状态 s_{t+i} 根据策略 π 采取动作 a_{t+i} 所获得的立即奖赏，$V(S_t)$ 表示在状态 s_t 的期望回报值。通过 TD 误差来评估当前正被选择的动作 a_t 的好与坏。如果 TD 误差为正，则表明未来选择动作 a_t 的趋势应该加强；如果 TD 误差为负，则表明未来选择动作 a_t 的趋势应该减弱。

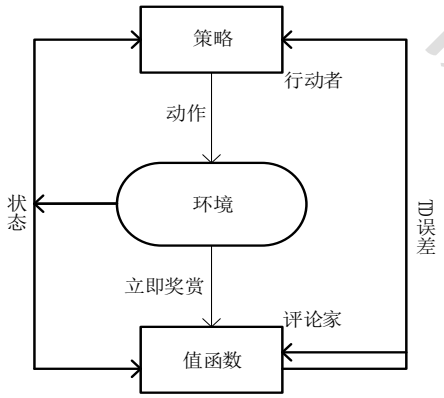


图 1　AC 算法结构图

AC 算法有两个优点：1、它是一种策略梯度算法，策略是明确的，当动作空间是连续的情况时，在选择动作时不需要为每一次的动作选择在无穷的动作空间中做大量的计算；2、它通过对策略的直接更新，使 Agent 能够学习到一个确定的随机策略。

2.2 优势行动者-评论家网络

行动者-评论家网络通过把深度神经网络与 AC 算法相结合，使得传统的 AC 算法能够在大规模任务状态空间和高维度任务状态空间中更加有效。结合了深度神经网络的 AC 算法不需要对原始数据进行复杂的预处理，能够使得算法实现端到端[22]（end-to-end）的学习。行动者-评论家网络主要有两个部分：一是值网络部分，即 $V(s_t;\boldsymbol{\theta}_v)$，其中 $\boldsymbol{\theta}_v$ 表示值网络的参数。二是策略网络部分，即 $\pi(a_t|s_t,\boldsymbol{\theta})$，其中 $\boldsymbol{\theta}$ 表示策略网络的参数。策略网络用来计算在给定状态 s_t 所采取的动作 a_t，值网络用来评价在给定状态 s_t 时策略网络所采取的动作 a_t 的好与坏。深度 AC 算法在更新策略网络参数时对于每一个状态-动作对都采用相同的权重 1，即平等地对待每一个状态动作对。然而状态-动作对之间的重要性是不同的，有的状态-动作对可以获得高的回报值，而有的状态-动作对获得的回报值相对较低，故而平等地对待这些状态-动作对忽略了他们之间所获得的回报不同这一事实。为了充分利用这一有效信息，在深度 AC 算法中引入一个优势函数，用来评价当前状态-动作对的优势，表示为 $A(s_t,a_t;\boldsymbol{\theta},\boldsymbol{\theta}_v)$，称为优势行动者-评论家算法。优势函数的计算公式如式（5）所示：

$$A(s_t,a_t;\boldsymbol{\theta},\boldsymbol{\theta}_v) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n};\boldsymbol{\theta}_v) - V(s_t;\boldsymbol{\theta}_v) \tag{5}$$

其中当 $n=1$时表示的是 1 步回报优势函数，当 $n=k$ 时表示的是 k 步回报优势函数，$\gamma \in [0,1]$ 表示折扣因子，r 表示立即奖赏。优势行动者-评论家算法的策略函数和值函数的梯度计算如式（6）和式（7）所示：

$$d\boldsymbol{\theta} = \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t;\boldsymbol{\theta}) A(s_t,a_t;\boldsymbol{\theta},\boldsymbol{\theta}_v) \tag{6}$$

$$d\boldsymbol{\theta}_v = \partial (R - V(s_t;\boldsymbol{\theta}_v))^2 / \partial \boldsymbol{\theta}_v \tag{7}$$

其中 R 表示在状态 s_t 根据策略 π 采取动作 a_t 所获得的回报值，$\boldsymbol{\theta}$、$\boldsymbol{\theta}_v$ 分别表示策略函数和值函数的参数。策略函数和值函数参数通过随机梯度下降方法来进行更新，如式（8）和式（9）所示：

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta d\boldsymbol{\theta} \tag{8}$$

$$\boldsymbol{\theta}_v = \boldsymbol{\theta}_v - \eta d\boldsymbol{\theta}_v \tag{9}$$

其中 η 表示学习率，$d\boldsymbol{\theta}$、$d\boldsymbol{\theta}_v$ 计算方式如式（6）和式（7）所示。

2.3 门限机制

循环神经网络相比于卷积神经网络，能够处理序列状态之间在不同时间尺度存在依赖关系的任

务，是由于循环神经网络中的循环连接允许网络每一层访问先前的状态信息。LSTM 网络是应用门限机制最为成功的神经网络之一[23-24]。LSTM 网络通过输入门、输出门、遗忘门以及记忆单元来控制信息在网络中的流通以达到记忆以前任务状态信息的目的。LSTM 单元的遗忘门的计算方式如式（10）所示：

$$\boldsymbol{f}_t^j = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{V}_f \boldsymbol{c}_{t-1})^j \quad (10)$$

其中 j 表示第 j 个 LSTM 单元，σ 表示 sigmoid 非线性函数，$\boldsymbol{f}_t^j$ 表示 t 时间步第 j 个 LSTM 单元的遗忘门的输出，$\boldsymbol{W}_f$、$\boldsymbol{U}_f$、$\boldsymbol{V}_f$ 表示的是 LSTM 单元遗忘门的参数，$\boldsymbol{x}_t$ 表示 t 时间步的输入数据，$\boldsymbol{h}_{t-1}$ 表示 $t-1$ 时间步 LSTM 单元的输出值，$\boldsymbol{c}_{t-1}$ 表示 $t-1$ 时间步 LSTM 单元的记忆单元的输出值。输入门的计算公式如式（11）所示：

$$\boldsymbol{i}_t^j = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{V}_i \boldsymbol{c}_{t-1})^j \quad (11)$$

其中 $\boldsymbol{i}_t^j$ 表示 t 时间步第 j 个 LSTM 单元的输入门的输出，$\boldsymbol{W}_i$、$\boldsymbol{U}_i$、$\boldsymbol{V}_i$ 表示 LSTM 单元输入门的参数。新的记忆上下文计算公式如式（12）所示：

$$\tilde{\boldsymbol{c}}_t^j = \tanh(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1})^j \quad (12)$$

其中 $\tilde{\boldsymbol{c}}_t^j$ 表示新的记忆上下文，$\boldsymbol{W}_c$、$\boldsymbol{U}_c$ 表示记忆上下文的参数，tanh 表示双曲正切非线性函数。根据输入门的输出 $\boldsymbol{i}_t^j$、遗忘门的输出 $\boldsymbol{f}_t^j$、记忆上下文 $\tilde{\boldsymbol{c}}_t^j$ 以及上一个时间步的记忆单元的内容 $\boldsymbol{c}_{t-1}^j$ 可以计算更新记忆单元的内容 $\boldsymbol{c}_t^j$，计算公式如式（13）所示：

$$\boldsymbol{c}_t^j = \boldsymbol{f}_t^j \boldsymbol{c}_{t-1}^j + \boldsymbol{i}_t^j \tilde{\boldsymbol{c}}_t^j \quad (13)$$

由 $\boldsymbol{x}_t^j$、$\boldsymbol{h}_{t-1}^j$ 和 $\boldsymbol{c}_t^j$ 可以得到 LSTM 单元的输出门的计算公式如式（14）所示：

$$\boldsymbol{o}_t^j = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{V}_o \boldsymbol{c}_t)^j \quad (14)$$

由式（13）和式（14）可以得出 LSTM 单元最终的输出 $\boldsymbol{h}_t^j$，计算公式如式（15）所示：

$$\boldsymbol{h}_t^j = \boldsymbol{o}_t^j \tanh(\boldsymbol{c}_t^j) \quad (15)$$

完整的 LSTM 单元计算过程图如图 2 所示。

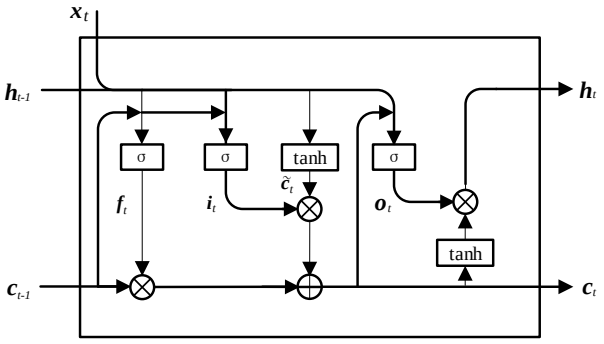


图 2 LSTM 单元计算过程

3 基于多重门限机制的异步优势行动者-评论家算法

本节将主要阐述多重门限机制和跳跃连接以及 A3C-MGM 模型的具体构架和模型参数的训练方法。其中，3.1 节介绍多重门限机制，3.2 节介绍跳跃连接方式，3.3 节介绍 A3C-MGM 各网络部分的构成以及处理数据的过程，3.4 节介绍 A3C-MGM 模型的参数更新方式。

3.1 多重门限机制

LSTM 网络通过复杂的门限机制来控制信息在网络中的流通，并且当前时刻的 LSTM 单元的信息会作为下一个时刻的 LSTM 单元的输入，允许访问以前的状态信息，达到记忆状态信息的效果。虽然 LSTM 网络通过这样的门处理机制在状态之间存在依赖关系的任务中可以获得比普通前馈网络更好的效果，但其计算依赖于上一个时间步的输出，故而其训练过程无法像前馈网络一样利用并行化计算来加速训练。

现代计算设备大多支持并行化计算，为了充分利用并行化计算技术加速深度神经网络的训练，Oord 等人[25]依据 LSTM 网络的门限控制思想提出了一种 LSTM 风格的前馈门限机制的神经网络，称为门控 tanh 单元（Gated Tanh Unit, GTU）。计算公式如式（16）所示：

$$h_l(\boldsymbol{X}) = \tanh(\boldsymbol{X} * \boldsymbol{W} + \boldsymbol{b}) \otimes \sigma(\boldsymbol{X} * \boldsymbol{V} + \boldsymbol{c}) \quad (16)$$

其中 $\boldsymbol{W} \in \mathbb{R}^{m \times n}$、$\boldsymbol{b} \in \mathbb{R}^n$、$\boldsymbol{V} \in \mathbb{R}^{m \times n}$、$\boldsymbol{c} \in \mathbb{R}^n$ 是网络参数；$\boldsymbol{X} \in \mathbb{R}^{N \times m}$ 表示 h_l 的输入，是上一层的输出；σ 表示 sigmoid 非线性函数，tanh 表示双曲正切非线性函数，$\otimes$ 表示矩阵间对应元素相乘。GTU 的

计算不需要依赖于上一时间步网络的输出，可以通过并行化计算加速网络模型的训练过程。Dauphin等人[26]提出了一种称为门控线性单元（Gated Linear Unit, GLU）的门控机制，计算如式（17）所示：

$$h_l(\boldsymbol{X})=(\boldsymbol{X}*\boldsymbol{W}+\boldsymbol{b})\otimes\sigma(\boldsymbol{X}*\boldsymbol{V}+\boldsymbol{c}) \quad (17)$$

其中各字符含义与式（16）中的字符含义相同。通过这种门限线性单元，Facebook 的研究人员 Dauphin 等人[26]首次在神经机器翻译任务中证明了非循环神经网络的性能胜过循环神经网络。

GLU 门控单元中使用了 sigmoid 函数作为门激活函数，但是 sigmoid 函数把所有数据映射到一个 0 到 1 之间的数值，使得激活后的数据都在 0 到 1 之间，sigmoid 函数映射区域比较窄，函数曲线变化平缓，映射后的数据比较集中，使得数据差异变小了。tanh 函数可以把输入数据映射到-1 到 1 之间，数据映射到了一个更宽的数据范围，使得数据差异更加明显；但是 tanh 函数的梯度容易达到饱和，即梯度为 0。sigmoid 函数的梯度的非饱和区域要比 tanh 函数的梯度的非饱和区域宽很多，如图 3 所示。为了利用并行化技术加速模型训练，并把 GTU 单元和 GLU 单元的门限控制思想的优点相结合。本文提出一种新的门控机制单元，利用多重门限机制来控制信息在网络中的流通，达到记忆信息的效果，称为多重门限机制（Multiple Gating Mechanisms, MGM）。计算公式如式（18）所示：

$$h_l(\boldsymbol{X})=(\boldsymbol{X}*\boldsymbol{W}+\boldsymbol{b})\otimes (\tanh(\boldsymbol{X}*\boldsymbol{V}+\boldsymbol{c})+\sigma(\boldsymbol{X}*\boldsymbol{V}+\boldsymbol{c})) \quad (18)$$

其中各字符含义与式（16）中的字符含义相同。

上述门控机制的优缺点可通过对其计算公式进行求梯度来进行分析，分别对式（16）、（17）和（18）求导得到式（19）、（20）和（21），如下所示：

$$\nabla[\tanh(\boldsymbol{X})\otimes\sigma(\boldsymbol{X})]=\tanh'(\boldsymbol{X})\nabla\boldsymbol{X}\otimes\sigma(\boldsymbol{X})+\sigma'(\boldsymbol{X})\nabla\boldsymbol{X}\otimes\tanh(\boldsymbol{X}) \quad (19)$$

$$\nabla[\boldsymbol{X}\otimes\sigma(\boldsymbol{X})]=\boldsymbol{X}\otimes\sigma'(\boldsymbol{X})\nabla\boldsymbol{X}+\nabla\boldsymbol{X}\otimes\sigma(\boldsymbol{X}) \quad (20)$$

$$\nabla[\boldsymbol{X}\otimes(\sigma(\boldsymbol{X})+\tanh(\boldsymbol{X}))]=\nabla\boldsymbol{X}\otimes(\sigma(\boldsymbol{X})+\tanh(\boldsymbol{X}))+\boldsymbol{X}\otimes(\sigma'(\boldsymbol{X})+\tanh'(\boldsymbol{X}))\nabla\boldsymbol{X} \quad (21)$$

其中 $\boldsymbol{X}$ 表示的是经过网络层处理但未被激活函数激活的数据，含义与式（16）、（17）和（18）中的含义不同。式（19）是 GTU 的梯度计算公式，由于对于激活函数 $\sigma(\boldsymbol{X})$ 和 $\tanh(\boldsymbol{X})$ 的梯度存在缩放因子 $\sigma'(\boldsymbol{X})$ 和 $\tanh'(\boldsymbol{X})$，随着网络层数的堆叠，缩放因子的影响将进一步放大，从而造成严重的梯度消失问题。式（20）是 GLU 的梯度计算公式，第二项对于梯度不存在缩放因子，故而梯度可以沿着这条路径传播的更远，有效缓解了随着网络层数的堆叠造成的梯度消失问题。式（21）是 MGM 的梯度计算公式，第一项对于梯度不存在缩放因子，梯度可以沿着这条路径传播的更远；并且 $(\sigma'(\boldsymbol{X})+\tanh'(\boldsymbol{X}))$ 大于 1 的部分对于梯度具有放大作用，可以缓解梯度在反向传播过程中被缩放的问题；$(\sigma(\boldsymbol{X})+\tanh(\boldsymbol{X}))$ 函数梯度的非饱和区域比 tanh 函数梯度的非饱和区域要宽很多；$\sigma(\boldsymbol{X})$ 把数据映射到一个很窄的范围，并且 sigmoid 函数变化平缓，数据间的差异容易被深度神经网络缩小，从而导致对于数据的有效识别能力变差，而 $(\sigma(\boldsymbol{X})+\tanh(\boldsymbol{X}))$ 函数通过把数据映射到-1 到 2 之间，映射的数据范围更宽，且函数曲线更陡峭，比仅用 sigmoid 函数的门限机制更能获取正确的数据。图 3 展示了各函数的数据和梯度变化曲线，从中可以看出 MGM 函数保留了 sigmoid 函数和 tanh 函数的平滑性，并且增大了数据映射和梯度的范围。

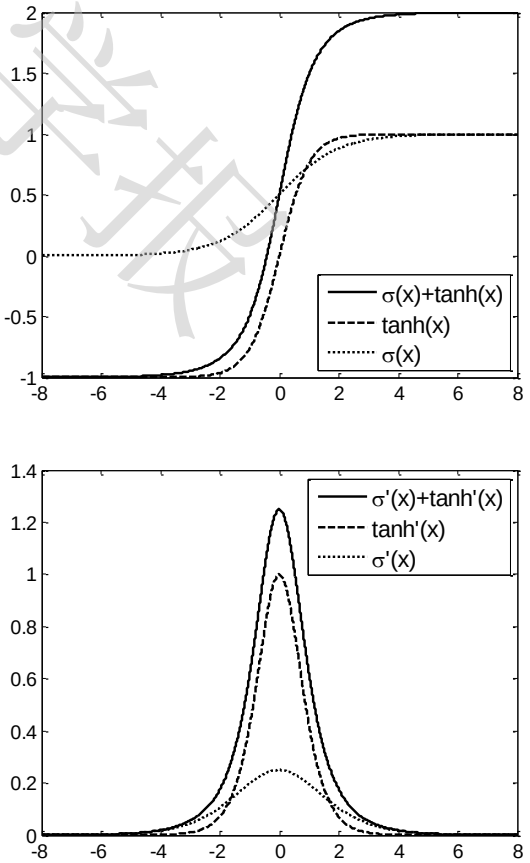


图 3 门限机制的函数曲线与梯度曲线

3.2 跳跃连接

残差网络[27]和高速通道网络[28]通过在不同的网络层之间建立捷径连接，成功地扩展了网络的深度。残差网络通过残差连接构建层与层之间的捷径连接，使得深层神经网络（几十层，甚至几百层的神经网络）可以被有效地训练。此外，文献[29-32]表明，对于大量的视觉任务而言，在 CNN 中通过跳跃连接来利用多级特征是有效的。虽然基于多重门限机制的异步优势行动者-评论家算法能够以相对少的时间获得更好的性能，但是其稳定性表现得并不好，本文第四部分展示了各模型实验结果。为此，本文提出了一种新的跳跃连接方式。残差网络主要通过残差连接使得网络的层数能够被设计的更深（残差连接主要用于几十层甚至几百层的神经网络），与残差网络不同的是，该跳跃连接方式主要是通过特征重用来提升深度神经网络模型的稳定性和训练效果。通过把状态数据向更深的网络层传递，使得网络能够进一步探索状态的新特征，达到更准确地识别当前状态信息的效果，做出更优的决策，从而提升模型的训练效果和稳定性。令 X_l 表示第 l 层卷积神经网络的输入，Y_l 表示第 l 层的输出，则该跳跃连接方式如式（22）和（23）所示：

$$X_l = concat(Y_{l-3}, Y_{l-1}) \tag{22}$$

$$Y_l = function(X_l) \tag{23}$$

其中 $concat(A, B)$ 表示将 A 和 B 的数据沿着最后一个维度进行连接，组成一个新的数据；$function(X_l)$ 表示输入数据 X_l 通过非线性函数变换得到输出 Y_l。跳跃连接方式的示意图如图 4 所示：

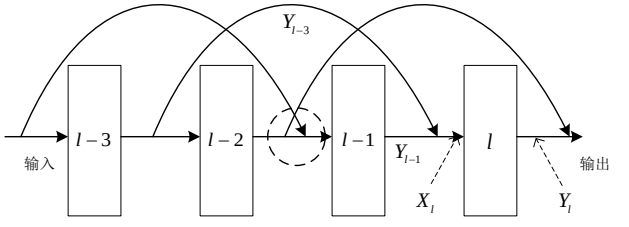


图 4 跳跃连接。其中矩形框表示的是不同的网络层，曲线箭头表示的是跳跃连接；图中的虚线圆圈区域中，跳跃连接的输入和输出是分离的，即下一个跳跃连接所传递的数据仅仅是上一个网络层的输出，并不传递前面的网络层所传递的跳越连接数据。X_l, Y_l 分别表示第 l 层的输入和输出，$l-3$，$l-2$，$l-1$ 和 l 表示的是不同的网络层

在普通的多重门限机制模型中，每一层的输入仅来自于上一层的输出。如果网络的当前层无法有效地识别输入数据中的有效特征信息，这些特征信息就会被该网络层过滤掉，整个网络模型都将无法识别这一部分的有效特征，导致 Agent 的错误识别，做出错误的决策，使得 Agent 的性能表现波动较大。而引入跳跃连接之后，可以通过把输入数据向前传递的更远，使网络模型能够更准确地提取当前状态的特征信息，从而让 Agent 做出更准确的决策，有效缓解了 Agent 性能不稳定的问题。

3.3 神经网络模型构建

本文提出的基于多重门限机制的异步优势行动者-评论家算法（A3C-MGM）具有以下特点：1、网络结构简单，能够处理状态之间存在依赖关系的任务；2、全连接层和卷积层均采用 MGM 机制；3、可以利用并行化计算加速模型训练。A3C-MGM 模型由 CNN、MGM、全连接层组成，如图 5 所示。接下来以 Atari 2600 游戏为例，具体分析 A3C-MGM 模型各模块的作用以及处理数据的过程。

Atari 2600 游戏原始视频帧大小为 $210\times160\times3$，210 表示视频帧的像素高度，160 表示视频帧的像素宽度，3 表示 3 通道的彩色（RGB）图像。由于在视频图像中并不是所有的像素点都是有价值的，例如游戏视频图像中显示得分的区域和一些无价值的边界区域。若原始游戏视频图像不经过预处理就直接输入到图 5 所示的神经网络中，这些无价值的像素区域将会耗费额外的计算资源和存储资源，并且还有可能会使模型的训练效果变差。因此有必要对原始游戏视频帧做一些简单的预处理。首先将原始的大小为 210×160 像素的 RGB 游戏图像通过一次降采样方法转换成大小为 160×160 像素的 RGB 游戏图像。然后，再次通过降采样方法把大小为 160×160 像素的 RGB 游戏图像转换成大小为 84×84 像素的 RGB 游戏图像。使用两次降采样操作主要原因是：对图像进行降采样操作，当操作后的图像相对于原来图像太小时容易丢失原来图像中的部分有价值的信息。因此，本文采用两次降采样操作来把原始游戏视频图像裁减成大小为 84×84 像素的 RGB 游戏图像。由于 84×84 像素的 RGB 图像可以表示成 $84\times84\times3$ 的形式，84×84 像素的灰度图像可以表示成 $84\times84\times1$ 的形式，RGB 图像的数据量是灰度图像的 3 倍，所以 RGB 图像所耗费的存储资源和计算资源都将会比

灰度图像多。由于计算机对于识别灰度图像和 RGB 图像并无差别，不像人类眼睛能够有效识别 RGB 图像中的信息却无法有效识别灰度图像中的信息；并且当 RGB 图像转换成灰度图像时并不会丢失图像中有价值的信息。所以将经过两次降采样操作后的游戏视频图像转换为灰度图像，可以进一步减少训练模型过程中所消耗的存储资源与计算资源。

A3C-MGM 将 1 幅大小为 210×160×3 的原始游戏视频图像经过预处理之后输入到网络模型中，决策阶段的网络模型输入规模可以表示成 1×84×84×1，评估阶段的网络模型的输入规模可以表示成 20×84×84×1（A3C-MGM 采用的最大时间步为 20，由于会遇到情节结束状态，所以最大时间步在这种情况下将会小于 20）。相比于原始的大小为 1×210×160×3 的输入规模和大小为 20×210×160×3 的输入规模，大幅度减少了计算策略函数和值函数以及更新网络参数所需的存储资源和计算资源，进一步加速了 Agent 的训练。对于 20×84×84×1 大小的输入序列，由于 LSTM 网络的计算依赖于序列中上一个数据的计算结果，故而基于 LSTM 网络的 ADRL 算法无法利用并行化计算一次性计算这 20 个数据。而 A3C-MGM 的计算并不依赖于上一个时间步，其通过 MGM 来控制信息在网络中的流通，达到记忆信息的效果，因此可以通过并行化计算，一次计算 20 个数据，大幅度减少了 ADRL 模型的训练时间。

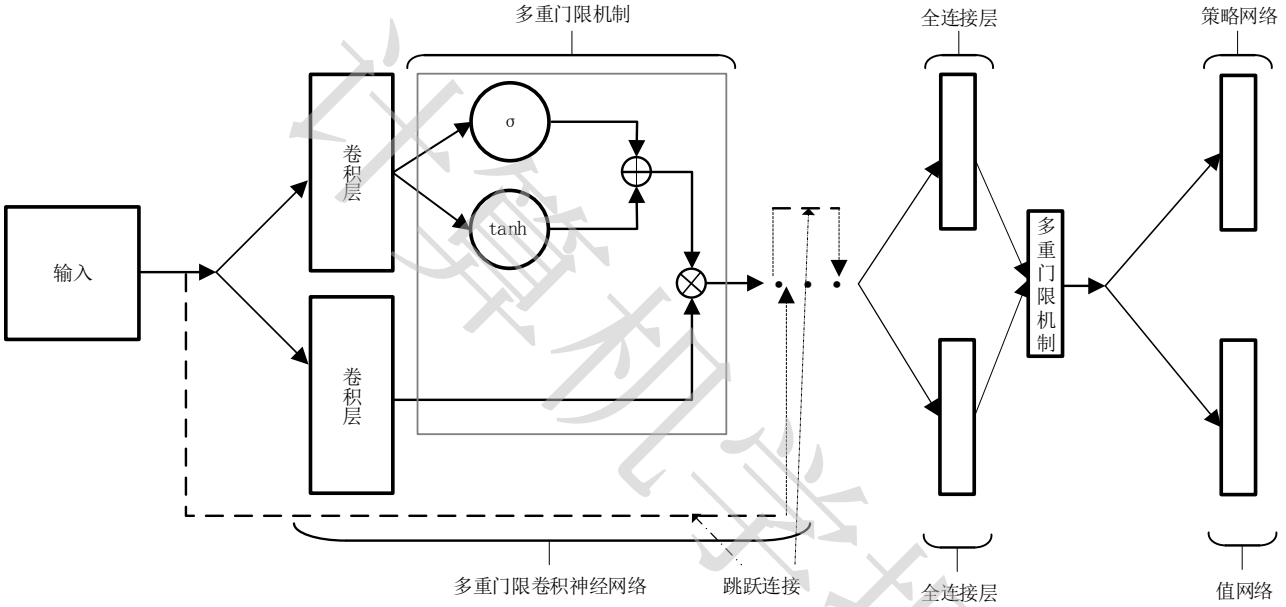


图 5 A3C-MGM 网络模型。模型采用三层多重门限卷积神经网络、一层全连接网络、一层全连接策略网络和一层全连接值网络；且多重门限卷积层之后的全连接层也采用多重门限机制。⊕ 表示矩阵之间对应元素相加，⊗ 表示矩阵之间对应元素相乘，σ 表示 sigmoid 非线性函数，tanh 表示双曲正切非线性函数。虚线部分展示了跳跃连接在门限机制网络模型中的连接方式。输入为一幅或者一组经过预处理的游戏视频图像

本文的 A3C-MGM 网络模型采用 3 层多重门限卷积层、1 层全连接层、1 层策略网络和 1 层值网络。通过上述预处理操作后的游戏视频数据作为第一层多重门限卷积层的两个卷积神经网络的输入，其中两个卷积神经网络均采用 32 个大小为 8×8 的卷积核，步长大小为 4。输入数据经过第一个卷积神经网络处理后产生 32 幅大小为 20×20 的特征图，通过使用 sigmoid 函数和 tanh 函数进行非线性变换，再把非线性变换后的数值对应元素相加；第二个卷积神经网络同样产生 32 幅大小为 20×20 的特征图，且不使用激活函数来进行非线性变换。两个卷积神经网络所产生的输出对应元素之间相乘，作为下一层的输入。第二层多重门限卷积神经网络和第三层多重门限卷积神经网络与第一层多重门限卷积神经网络具有相同的结构。第二层使用 64 个大小为 4×4 的卷积核，步长大小为 2，输出 64 幅大小为 9×9 的特征图，作为第三层的输入。第三层使用 64 个大小为 3×3 的卷积核，步长大小为 1，输出 64 幅大小为 7×7 的特征图。

第三层多重门限卷积神经网络层输出的特征图，经过扁平化操作处理后，作为一个包含 256 个神经元的全连接层的输入。该全连接层采用和卷积层同样的多重门限机制，输入数据分别作为两层同样的全连接层输入，其中一层的输出经过多重门限

机制操作后，与另一层的输出对应元素相乘，作为策略网络层和值网络层的输入。策略网络是一个全连接层，其神经元数量与所处理的游戏的动作空间大小相同；值网络是一个仅包含一个神经元的全连接层。策略网络输出是动作对应的动作值，经过 Softmax 回归操作处理后，用来选择游戏动作；值网络的输出是状态值，用来计算优势。

3.4 模型参数更新方法

基于多重门限机制的模型主要包含一个策略函数网络 $\pi(a_t|s_t\ \boldsymbol{\theta}$ 和一个值函数估计网络 $V(s_t;\boldsymbol{\theta}_v)$，是一个平滑的网络模型，模型中的各部分都是可微分的，即模型中的可训练参数都存在关于自身的梯度。本文采用一种自适应矩估计[33]（Adaptive Moment Estimation, Adam）的随机梯度下降方法来更新网络模型的参数。

策略函数 $\pi(a_t|s_t;\boldsymbol{\theta})$ 用来在状态 s_t，根据策略 π 选择一个动作 a_t；值函数 $V(s_t;\boldsymbol{\theta}_v)$ 计算状态 s_t 的状态值，状态值用来计算状态 s_t 的优势。A3C-MGM 采用了异步方法来更新参数，并且当 Agent 与环境交互后并不立即利用梯度下降方法更新模型参数，而是在 Agent 与环境交互 $t_{\max}$ 时间步后或者当一个情节结束时更新一次模型参数。策略函数网络和值函数网络的参数均使用梯度下降方法进行更新，梯度计算公式如式（24）和式（25）所示：

$$d\boldsymbol{\theta}=\sum_{i=t}^{t_{\max}}\nabla_{\boldsymbol{\theta}'}\log\pi(a_i|s_i;\boldsymbol{\theta}')(R_i-V(s_i;\boldsymbol{\theta}'_v)) \tag{24}$$

$$d\boldsymbol{\theta}_v=\sum_{i=t}^{t_{\max}}\frac{\partial(R_i-V(s_i;\boldsymbol{\theta}'_v))^2}{\partial\boldsymbol{\theta}'_v} \tag{25}$$

其中 $t_{\max}$ 表示固定的时间步间隔或者遇到情节结束情况时所经历的时间步步数；R_i 表示在状态 s_i 采取动作 a_i 获得的期望回报；$\boldsymbol{\theta}$ 和 $\boldsymbol{\theta}_v$ 分别表示共享的策略函数网络和共享的值函数网络的参数；$\boldsymbol{\theta}'$ 和 $\boldsymbol{\theta}'_v$ 分别表示当前策略函数网络和值函数网络的参数。

式（24）和（25）中利用参数 $\boldsymbol{\theta}'$ 和 $\boldsymbol{\theta}'_v$ 的梯度来更新参数 $\boldsymbol{\theta}$ 和 $\boldsymbol{\theta}_v$ 的原因是 A3C-MGM 利用多线程技术加速模型训练过程中，每个线程拥有自己的网络模型，参数为 $\boldsymbol{\theta}'$ 和 $\boldsymbol{\theta}'_v$，此外还有一个所有线程共享的网络模型，参数为 $\boldsymbol{\theta}$ 和 $\boldsymbol{\theta}_v$。每个线程网络模型的当前参数 $\boldsymbol{\theta}'$ 和 $\boldsymbol{\theta}'_v$ 都是从共享网络模型中获取，即 $\boldsymbol{\theta}'\leftarrow\boldsymbol{\theta}$ 和 $\boldsymbol{\theta}'_v\leftarrow\boldsymbol{\theta}_v$。每个线程并不更新自己的网络模型参数，而是更新所有线程共享的网络模型的参数。利用 Adam 梯度下降方法更新整个网络模型参数的详细过程如下：

$$\boldsymbol{g}_t=\nabla_{\bar{\theta}'_t}(\sum_{i=t}^{t_{\max}}\log\pi(a_i|s_i;\boldsymbol{\theta}'_t)(R_i-V(s_i;\boldsymbol{\theta}'_{v,t}))+\sum_{i=t}^{t_{\max}}(R_i-V(s_i;\boldsymbol{\theta}'_{v,t}))^2+H(\pi(s_t;\boldsymbol{\theta}'_t))) \tag{26}$$

其中 H 表示熵，$\bar{\boldsymbol{\theta}}'_t$ 表示当前线程整个网络模型的参数，$\boldsymbol{g}_t$ 表示总损失函数对于整个网络模型参数的梯度。

$$\boldsymbol{m}_t=\beta_1\boldsymbol{m}_{t-1}+(1-\beta_1)\boldsymbol{g}_t \tag{27}$$

$$\boldsymbol{v}_t=\beta_2\boldsymbol{v}_{t-1}+(1-\beta_2)\boldsymbol{g}_t^2 \tag{28}$$

其中 $\boldsymbol{m}_t$ 表示关于梯度的一次矩估计，$\boldsymbol{v}_t$ 表示关于梯度的二次矩估计，初始值都为 $\mathbf{0}$；$\beta_1,\beta_2\in[0,1)$，其值初始化为一个接近于 1 的值。

$$\hat{\boldsymbol{m}}_t=\frac{\boldsymbol{m}_t}{1-\beta_1^t} \tag{29}$$

$$\hat{\boldsymbol{v}}_t=\frac{\boldsymbol{v}_t}{1-\beta_2^t} \tag{30}$$

其中 $\hat{\boldsymbol{m}}_t$ 和 $\hat{\boldsymbol{v}}_t$ 分别表示 $\boldsymbol{m}_t$ 和 $\boldsymbol{v}_t$ 的偏差校正计算值。

$$\bar{\boldsymbol{\theta}}_{t+1}=\bar{\boldsymbol{\theta}}_t-\frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_t}+\varepsilon}\hat{\boldsymbol{m}}_t \tag{31}$$

其中 $\bar{\boldsymbol{\theta}}_t$ 表示整个共享网络模型的参数，ε 表示一个小的正常数，用来防止分母为 0 情况出现，η 表示学习率。

4 实验

本节先介绍实验所使用的平台和实验过程中所使用的参数设置，随后在部分战略性 Atari 2600 游戏中评估了 A3C-FF、A3C-LSTM、A3C-GLU、A3C-Tanh、A3C-MGM 和 MGM-skip 模型的训练时间和训练效果。A3C-FF 表示仅使用前馈神经网络（Feedforward, FF），A3C-LSTM 表示使用 LSTM 网络，A3C-GLU 表示使用了 GLU 门单元的前馈神经网络，A3C-Tanh 表示使用了 tanh 函数替代 sigmoid 函数的 GLU 门控机制。MGM-skip 表示使用了跳跃连接的 MGM 前馈神经网络模型，为了保证跨层传递的数据维度一致，本文在 MGM-skip 模型的不同网络层中采用相同的特征图数量。同时为了使跳跃连接得到充分利用，本文采用了 4 层卷积层来构建网络模型，其它设置与 A3C-MGM 相同。最后结合实验结果分析各模型在训练时间和训练

效果方面的优势与不足。

4.1 实验环境与参数设置

本文使用 OpenAI Gym[1]开源平台中的 Atari 2600 游戏环境作为实验环境。OpenAI Gym 是一个开源的工具包，其提供了各式各样的 Atari 2600 游戏接口，游戏类型涵盖了策略类、体育竞技类和桌游类游戏等。Mnih 等人[17]的研究表明，在 Atari 2600 大部分游戏中 A3C-FF 和 A3C-LSTM 在训练时间以及训练效果上都明显优于基于经验重放机制的 DRL 算法，如 DQN、DDQN 等。Dauphin 等人[26]的研究表明 GLU 门限机制要优于 GTU 门限机制。因此本文实验的重点是比较 A3C 算法的不同深度神经网络模型与本文提出的 A3C-MGM 之间的性能差异。本文选取了五个单 Agent 的 Atari 2600 游戏：Battle Zone、Tutankham、Space Invaders、Time Pilot 和 Pong，分别测试 A3C-FF、A3C-LSTM、A3C-GLU、A3C-Tanh、A3C-MGM 和 MGM-skip 模型在上述游戏环境中的表现。根据实验结果，对上述不同模型进行分析比较。为了进一步验证模型的有效性，本文额外选取了四个 Atari 2600 游戏来测试各模型的性能。

为了更好地对比不同模型的性能，本文所有模型均使用相同的参数集合和 Adam 梯度下降方法。Adam 梯度下降方法中的参数设置如下：$\eta = 0.001$、$\beta_1 = 0.9$、$\beta_2 = 0.99$、$\varepsilon = 10^{-3}$。上述所有算法的折扣因子均采用 $\gamma = 0.99$。网络参数的异步更新方式如下：每 20 步（帧）或者情节结束时更新一次共享网络模型的参数。决策阶段，每次输入 1 帧经过预处理的大小为 84×84 的灰度图像，作为网络模型的输入；评估阶段，每次输入一定批量经过预处理的大小为 84×84 的灰度图像。由于模型每 20 帧或者情节结束时更新一次网络参数，所以评估阶段的输入数据批量不是固定的，会出现小于 20 的情况。实验采用 1,000 个训练阶段（Epoch）作为训练周期，其中每个阶段 80,000 步，共训练 80,000,000 步。本文所有实验均使用 8 线程来加速模型训练。

4.2 实验评估与结果分析

文献[34]表明：当强化学习算法与非线性函数逼近器相结合时，会导致算法的收敛性难以得到保证；深度强化学习算法正是采用大型深度神经网络作为非线性函数逼近器来学习值函数或者动作值函数，因而算法的收敛性无法从理论上得到保证。然而，Mnih 等人[6-7]提出了 DQN，引入了经验重放缓冲区和单独的目标网络，使得算法的稳定性得到显著的改善。本文使用异步方法实现与 DQN 算法中的经验重放机制相同的作用，提高算法稳定性并加快模型的学习。一般的深度强化学习模型的训练通常都需要很长的时间周期，在深度强化学习中引入异步方法替代经验重放机制，可以减少存储资源和计算资源的消耗。A3C-MGM 通过使用 MGM，增加非饱和区域的宽度，并且把数据映射到一个更宽的数值范围，能更真实地映射数据间的关系，函数曲线如图 3 所示。A3C-MGM 通过 MGM 来控制信息在网络中的流通，达到记忆任务状态之间的依赖性信息的效果，并且可以使用并行计算加速模型的训练过程，以更少的时间代价来提升模型训练效果。MGM-skip 通过跳跃连接使数据可以向更深的网络层传递，有效提高了网络模型识别数据的能力，从而改善了模型训练效果和增强了算法稳定性。

本文首先比较了 A3C-FF、A3C-LSTM、A3C-GLU、A3C-Tanh、A3C-MGM 和 MGM-skip 模型在训练 Agent 玩 Battle Zone、Tutankham、Space Invaders、Time Pilot、Pong 等游戏中的训练时间。在 Intel Core i7-6800k CPU 上各模型的训练时间如表 1 所示。在 NVIDIA TITAN X（PASCAL） GPU 上各模型的训练时间如表 2 所示。

表 1 在单个 Intel Core i7-6800k CPU 上各模型的每步训练时间和训练总时间

模型	每步训练时间（s）	训练总时间（h）
MGM-skip	0.0030	66.7
A3C-MGM	0.0016	35.6
A3C-LSTM	0.0050	111.1
A3C-Tanh	0.0016	35.6
A3C-GLU	0.0016	35.6
A3C-FF	0.0010	22.2

表 2 在 NVIDIA TITAN X（PASCAL）GPU 上各模型的每步训练时间和训练总时间

模型	每步训练时间（s）	训练总时间（h）
MGM-skip	0.0012	26.7
A3C-MGM	0.0008	17.8
A3C-LSTM	0.0013	28.9
A3C-Tanh	0.0008	17.8

[1] https://github.com/openai/gym

A3C-GLU	0.0008	17.8
A3C-FF	0.0006	13.3

表 1 所示数据展示了各模型在 CPU 上的训练时间。其中 A3C-FF 的每步训练时间以及训练总时间最少，A3C-LSTM 的每步训练时间以及训练总时间最多，A3C-GLU、A3C-Tanh、A3C-MGM 的每步训练时间以及训练总时间相同。A3C-FF 的每步训练时间比 A3C-LSTM 少 0.0040 秒，比 A3C-GLU、A3C-Tanh、A3C-MGM 少 0.0006 秒。虽然 A3C-MGM 的每步训练时间比 A3C-FF 多 0.0006 秒，但比 A3C-LSTM 少 0.0034 秒；训练总时间比 A3C-FF 多 13.4 小时，但比 A3C-LSTM 少 75.5 小时。表 2 数据展示了各算法模型在 GPU 上的训练时间。GPU 虽然大幅度减少了各模型的训练时间，但是 A3C-LSTM 的训练总时间依然需要 28.9 小时，是 A3C-FF 的 2.17 倍，是 A3C-MGM、A3C-GLU 和 A3C-Tanh 的 1.62 倍。A3C-FF 训练时间最短是由于其卷积层仅有一个通道，而 A3C-MGM、A3C-GLU 和 A3C-Tanh 均有两个卷积通道。A3C-FF 训练时间最短，但其训练效果很差。虽然 MGM-skip 所需的训练时间比 A3C-MGM 多，但是训练效果更好更稳定。

虽然 LSTM、GLU 和 MGM 都是利用门控机制来处理状态之间存在依赖关系的任务，但是 LSTM 由于其复杂的门控单元连接以及循环结构，基于 LSTM 的异步深度强化学习算法的训练过程依赖于上一个时间步的计算结果，无法使用并行化计算进一步加速训练，导致模型训练时间特别长。而 MGM 和 GLU 通过简单的门单元来控制信息的流通，达到记忆信息的效果，且不存在循环连接结构。因此基于 MGM 和 GLU 的异步深度强化学习算法可以利用并行化计算处理数据，进一步加速模型训练过程。表 1、表 2 的数据充分展示了新模型在训练时间上的优势。

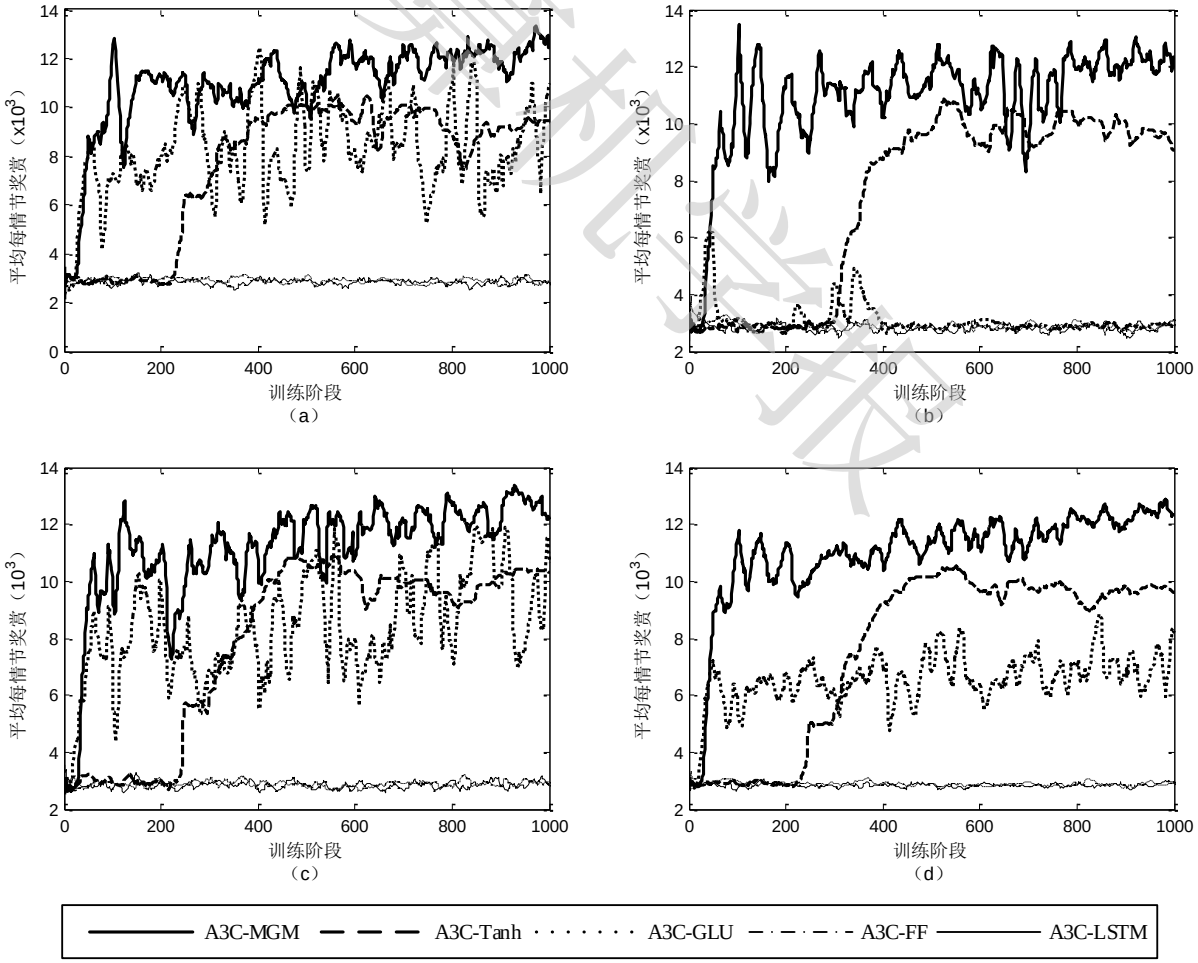
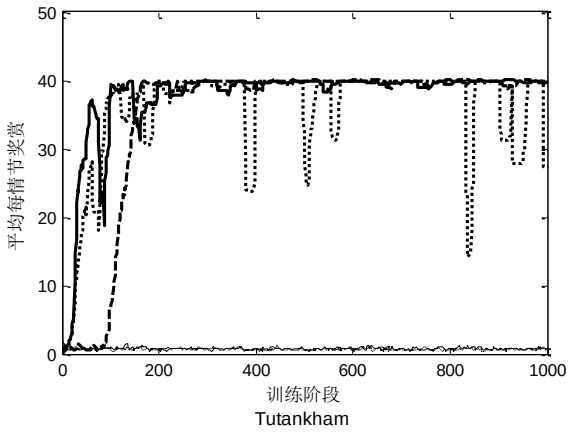


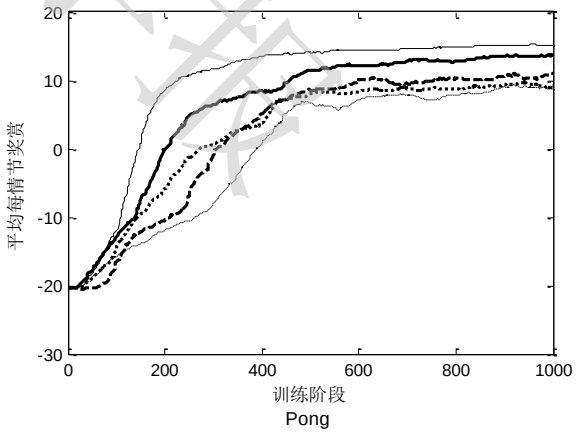
图 6 各模型在 Battle Zone 游戏中的训练效果，（a）、（b）和（c）表示各模型在 Battle Zone 游戏中三次独立的训练结果。（d）表示的是各模型三次训练结果的平均值。平均每情节奖赏表示的是训练过程中每 100 个情节的平均奖赏

本文评估了 A3C-MGM 在训练 Agent 玩 Battle Zone 游戏过程时各阶段平均每情节所获得的奖赏值，如图 6 所示。因强化学习是一种交互式学习，通过与环境交互获得一个反馈信号来训练 Agent。故而训练数据没有人工标记的标签，不需要单独的测试数据集来测试模型的最终性能。由于各个模型的参数初始值都是随机初始化的，不同的参数可能会影响模型的训练效果。本文在 Battle Zone 游戏环境中独立进行了三次实验，每次的模型参数都是随机初始化的，图 6 中的（a）、（b）和（c）分别是第一次、第二次和第三次实验结果。图 4 中的纵坐标平均每情节奖赏表示的是训练过程中每 100 个情节所获得的奖赏的平均值。从图 6 中的（a）、（b）和（c）可以看出，各个模型训练效果的曲线图虽然出现了不同程度的波动，但是整体趋势一样。因此不同的参数初始值并不能决定各模型性能的好坏，仅能在一定程度上影响着模型的性能。图 6 中的(d）表示的是三次独立实验各模型训练效果的平均曲线图。下面结合 Battle Zone 游戏，对不同的模型在训练时存在性能差异较大的原因进行具体分析。

在 Battle Zone 游戏过程中，由于 Battle Zone 游戏环境的奖赏是稀疏的，需要很多个时间步才能获得一次奖赏。例如，Battle Zone 中的坦克在寻找目标过程中，由于并未击毁目标故而无法获得奖赏，这一过程有时会持续非常多的时间步。而当前状态的输入信息仅仅是由当前游戏画面的一帧（A3C-FF）或者单层 LSTM 单元记忆的有限时间步长内的历史信息（A3C-LSTM）构成时，Agent 很大程度上无法及时地获得这个稀疏的奖赏，从而无法有效的学习。本文提出的 A3C-MGM 模型通过三层 MGM 卷积机制和一层 MGM 全连接机制来记忆更长时间步内的信息，有效缓解了当环境奖赏是稀疏情况时动作回报延时高的问题。并且相比于 LSTM，MGM 可以利用并行化计算加速训练过程，使得 Agent 能够很快学习到一个更优的策略。

图 6 展示了各算法在训练过程中的平均每情节奖赏演化曲线，可以看出，A3C-MGM 模型的训练效果最优，游戏得分可以达到 12,000 分左右。A3C-Tanh 的稳定性要比 A3C-GLU 好很多，但是其游戏得分最终不到 10,000 分，要比 A3C-MGM 差 2,000 分以上。而 A3C-FF 和 A3C-LSTM 的表现非常差，虽然模型表现最稳定，但是在 Battle Zone 游戏中的得分仅在 3,000 分左右，比 A3C-MGM 差了 9,000 分以上。MGM 把 GLU 门限机制宽的非饱和梯度区域优势和 tanh 函数宽的数据映射范围优势相结合，使得 A3C-MGM 的学习效果要比 A3C-GLU 和 A3C-Tanh 要好。在有限的 1,000 个训练阶段中，由于 A3C-FF 和 A3C-LSTM 均无法及时反馈环境的稀疏奖赏，其训练效果无法得到明显提升。





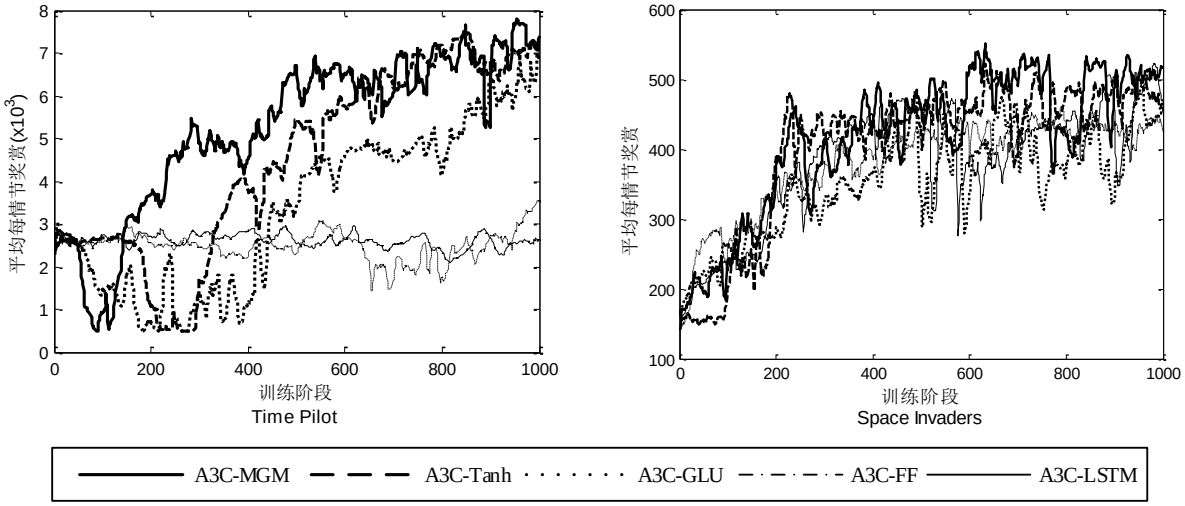


图 7 各模型在 Tutankham、Pong、Time Pilot 和 Space Invaders 游戏中的训练效果

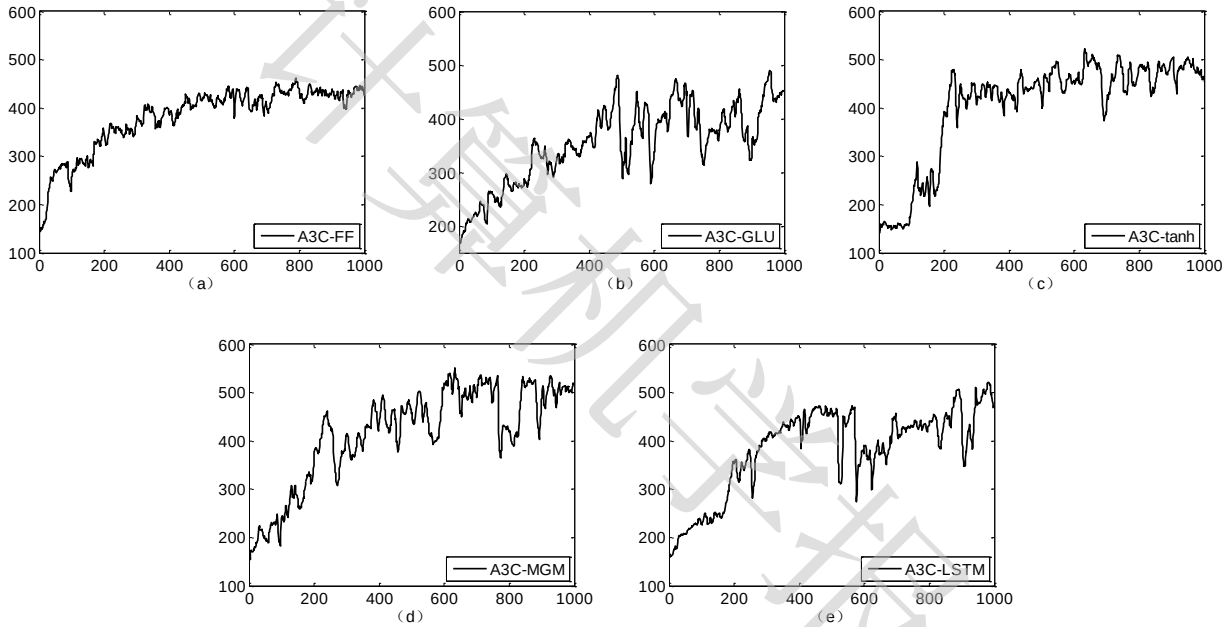


图 8 各模型在 Space Invaders 游戏环境中的性能表现。纵坐标表示的是平均每情节奖赏值，横坐标表示训练阶段

此外，本文还评估了 A3C-MGM 在训练 Agent 玩 Tutankham、Time Pilot、Space Invaders 和 Pong 这四种游戏时的性能表现，如图 7 所示，纵坐标含义与图 6 相同。各模型在 Space Invaders 环境中的表现差异性很小，所以在图 8 中单独展示该环境下各模型的训练效果。

上述各个模型在 Battle Zone、Tutankham、Space Invaders、Time Pilot 和 Pong 游戏中的表现都存在不稳定问题。这种现象是由于模型训练过程中的网络参数通过随机梯度下降方法不断地更新，即使参数发生很小的变动，也会导致模型输出的动作值发生改变，从而可能使得下一阶段策略选择动作时发生很大变化[6]。从图 6 和图 7 可以看出，在 Battle Zone 和 Tutankham 游戏中，A3C-GLU 的不稳定情况最为严重。这是由于 sigmoid 函数把数据映射到一个窄的、变化更加平缓的区域，数据间的差异性会被缩小。当 Agent 在当前游戏视频帧采取一个动作环境转换到下一个游戏视频帧或者采取较少的几个动作之后到达了后续的视频帧时，由于时间步比较少视频帧之间差异性比较小，这个小的差异被 sigmoid 函数进一步缩小，从而使得 Agent 无法做出对于当前视频帧更加正确的动作选择，导致了低的游戏得分。多个时间步之后，视频帧之间的差异性逐渐变大，此时即使 sigmoid 函数对数据差异性有缩小效果也无法严重影响这个差异，因而 Agent 能够根据不同的状态信息做出更加正确的动作选择，

从而可以获得高的游戏得分。这两个过程交替进行导致了 A3C-GLU 的严重不稳定问题。

其中 Tutankham 和 Time Pilot 游戏环境的奖赏相比于 Space Invaders 和 Pong 更加稀疏。前文实验表明不同的网络模型参数初始值并不能决定模型性能的好坏，故而在这四种游戏中不再取三次独立实验的实验结果的平均值。从图 7 可以看出，在 Tutankham 游戏中，A3C-MGM 和 A3C-Tanh 的平均每情节奖赏明显优于另外三种模型，在第 200 个训练阶段时开始收敛，在整个训练周期中最终收敛于 40（游戏得分）；在 Time Pilot 游戏中，A3C-MGM 和 A3C-Tanh 的性能表现也比另外三种模型要好。在 Space Invaders 游戏中，环境奖赏稀疏性比较弱，各个模型性能表现差异很小。由于 Pong 游戏环境的奖赏稀疏性比较弱，而且环境的状态是部分可观测的（即 Agent 无法观测到乒乓球的速度、加速度等信息，仅能观测到球的位置等信息），A3C-MGM 的性能比 A3C-LSTM 略差。因此 A3C-MGM 模型在环境奖赏的稀疏性比较弱的情况下，模型表现相比于另外几种模型，并不能取得明显的优势。所以 A3C-MGM 模型更加适用于环境奖赏是稀疏情况的一类战略性任务。

然而从上述实验结果中可以看出，A3C-MGM 算法的稳定性比较差。为了使 Agent 在这一类游戏中的表现更稳定，本文进一步提出了 MGM-skip 模型。该模型通过跳跃连接方式将数据传递的更远，使得不同的网络层能够学习到更多的状态特征，从而有效提升了 Agent 的稳定性。该模型在部分 Atari 2600 游戏环境中的实验结果如图 9 所示。

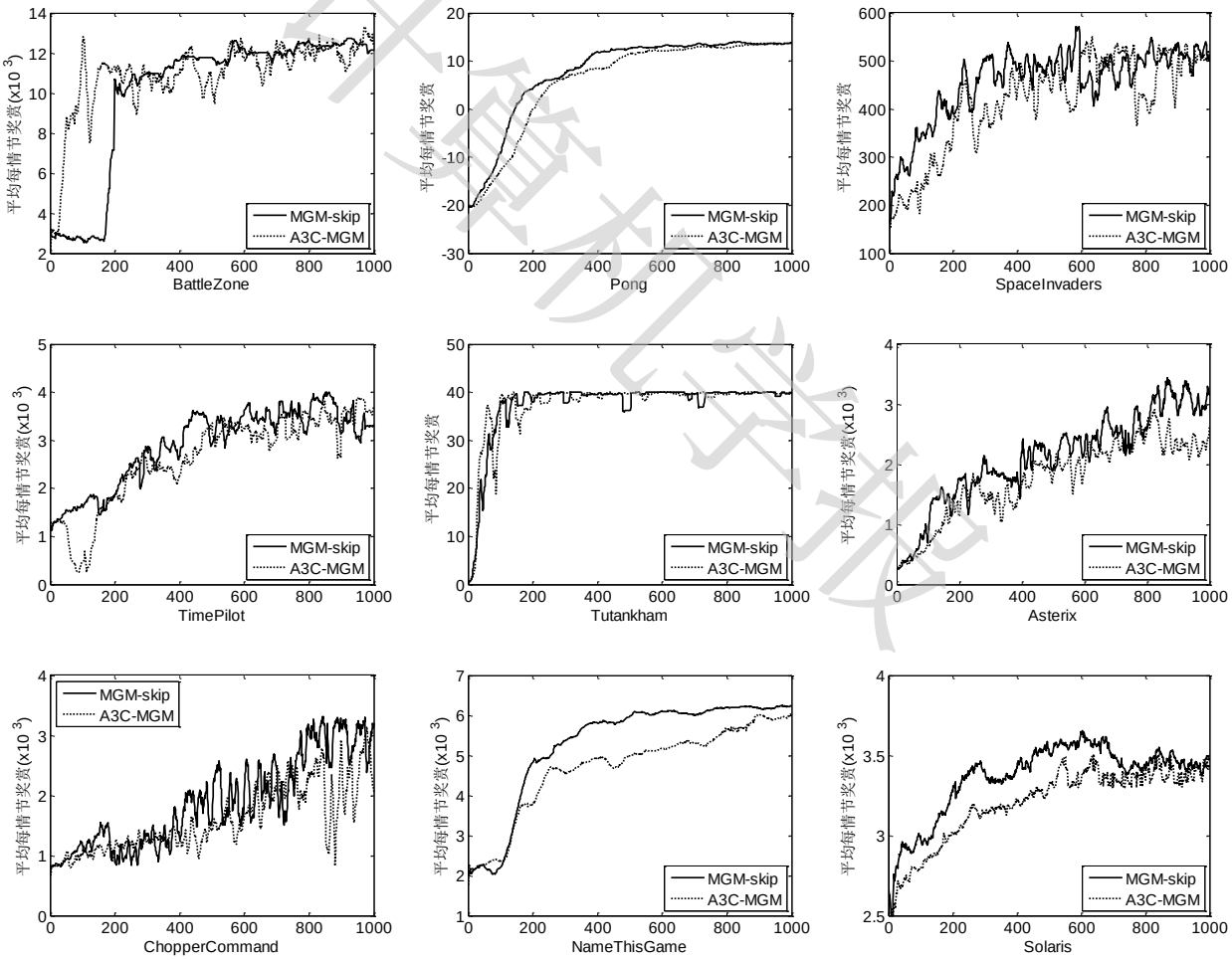


图 9 MGM-skip 模型与 A3C-MGM 模型的实验结果，横坐标表示训练阶段

从图 9 中可以看出，在 Battle Zone 游戏中 MGM-skip 比 A3C-MGM 更加稳定，在 Pong 游戏中 MGM-skip 获得了更好的学习效果，在 Space Invaders 和 Time Pilot 游戏中 MGM-skip 的学习效果要比 A3C-MGM 更稳定。在 Space Invaders 游戏中，在 780 个训练阶段之后，MGM-skip 波动幅度比 A3C-MGM 要小很多。在 Tutankham 游戏中，两者表现几乎一样。在 Asterix、Chopper Command 和

Solaris 游戏中，分别在 800、800 和 700 个训练阶段后，MGM-skip 均比 A3C-MGM 学习效果要好，并且更加稳定。在 Asterix 游戏中，A3C-MGM 在前 800 个训练阶段中学习效果逐步上升，在 800 个训练阶段之后，学习效果开始下降；而 MGM-skip 在整个学习过程中学习效果逐步上升，并且在一个较小的范围内波动。在 Chopper Command 游戏中，A3C-MGM 随着训练阶段的增加，不稳定性尤为明显，在 800 个训练阶段后，A3C-MGM 的波动范围特别大，而 MGM-skip 的表现更为稳定。在 Solaris 游戏中，虽然 MGM-skip 在整个学习过程中出现学习效果下降的表现，但是依然要好于 A3C-MGM 的学习效果。在 Name This Game 游戏中，MGM-skip 的学习效果在 200 个训练阶段后就明显优于 A3C-MGM，并且整个训练过程中都很稳定。因为游戏环境的本身固有的差异性（奖赏反馈机制以及反馈的奖赏数值均不同），导致了算法模型在有些环境中表现的很稳定，而在有些环境中表现的不稳定。但是在上述游戏环境中，MGM-skip 表现的均比 A3C-MGM 稳定。

为了进一步验证上述各个模型在 Atari 2600 游戏中的性能表现，本文额外选取了四个 Atari 2600 游戏来测试各模型的性能表现，实验结果如表 3 所示。表 3 中的数据展示了 MGM-skip 在 Atari 2600 游戏中的学习效果，在保证训练效果的前提下，MGM-skip 相比于 A3C-MGM 模型可以更稳定地获得更高的游戏得分。

表 3 各模型在部分 Atari 2600 游戏中的平均每情节奖赏（平均每情节奖赏值取整数）

游戏环境 / 模型	Asterix	Chopper Command	Name This Game	Solaris
MGM-skip	**3046±240**	**2852±391**	**6200±37**	**3438±72**
A3C-MGM	**2435±322**	**2442±698**	**5998±48**	**3396±72**
A3C-GLU	1949±252	2029±623	5487±296	3191±94
A3C-Tanh	1413±759	2421±528	5995±46	3290±286
A3C-FF	2021±259	914±72	5940±40	3066±108
A3C-LSTM	1826±142	938±39	2034±106	2657±72

从图 7 中训练效果曲线图可以看出，仅有 Pong 游戏在训练阶段相同时 A3C-MGM 的性能没有 A3C-LSTM 的性能好，但是 A3C-LSTM 耗费的训练时间更多，且 Pong 游戏环境奖赏的稀疏性比其它几个环境的奖赏稀疏性弱。为了比较在相同时间内 A3C-MGM 和 A3C-LSTM 在 Pong 游戏环境中的性能表现，各个模型在 Intel Core i7-6800k CPU 上和 NVIDIA TITAN X（PASCAL） GPU 上分别训练 20 小时和 12 小时，训练效果如图 10 所示。从图 10 中的（a）可以看出，在 CPU 上训练 20 小时 A3C-MGM 可以获得最优的训练效果，平均每情节奖赏为 12。图 10 中的（b）表明，在 GPU 上训练 12 小时 A3C-LSTM 是最优的，平均每情节奖赏为 14；A3C-MGM 是次优的，平均每情节奖赏是 12.5。在环境稀疏性比较弱和环境状态部分可观测的 Pong 游戏中，A3C-MGM 相比于 A3C-LSTM 无法获得最好的训练效果。但是在 CPU 上，A3C-MGM 相比于 A3C-LSTM 依然有时间优势。在 Pong 游戏中，虽然 A3C-MGM 在 GPU 上表现比 A3C-LSTM 差，但在 CPU 和 GPU 上 A3C-MGM 的表现要比 A3C-FF、A3C-GLU 和 A3C-Tanh 更优。而 MGM-skip 可以获得比 A3C-MGM 更好的学习效果，如图 11 所示。

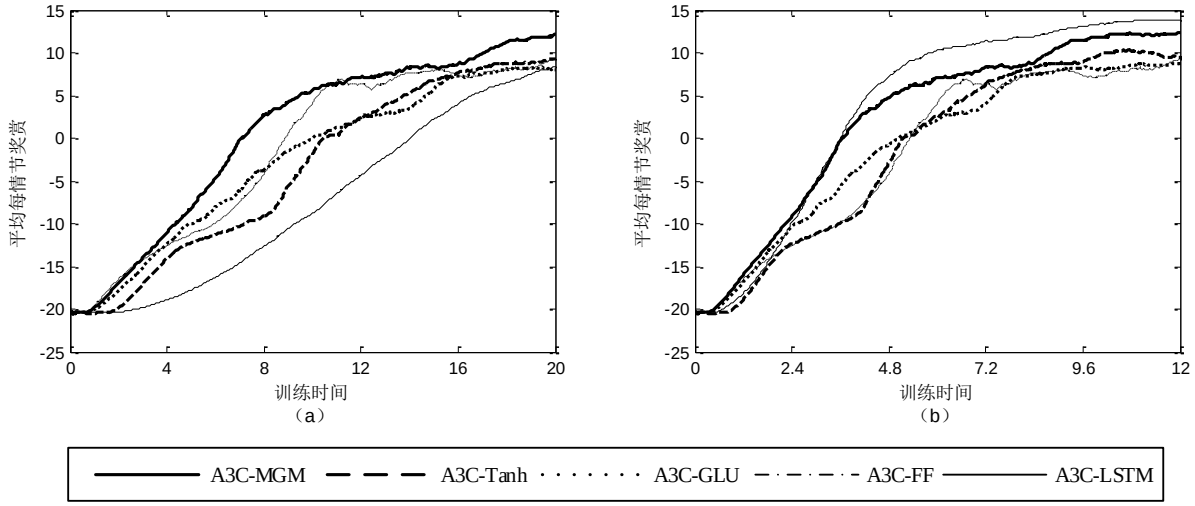


图 10　在 Pong 游戏中，各模型在 CPU 和 GPU 上分别训练 20 小时和 12 小时的训练效果。图中横坐标单位是小时（h）。（a）：在 Intel Core i7-6800k CPU 上的训练效果；（b）：在 NVIDIA TITAN X (PASCAL) GPU 上的训练效果

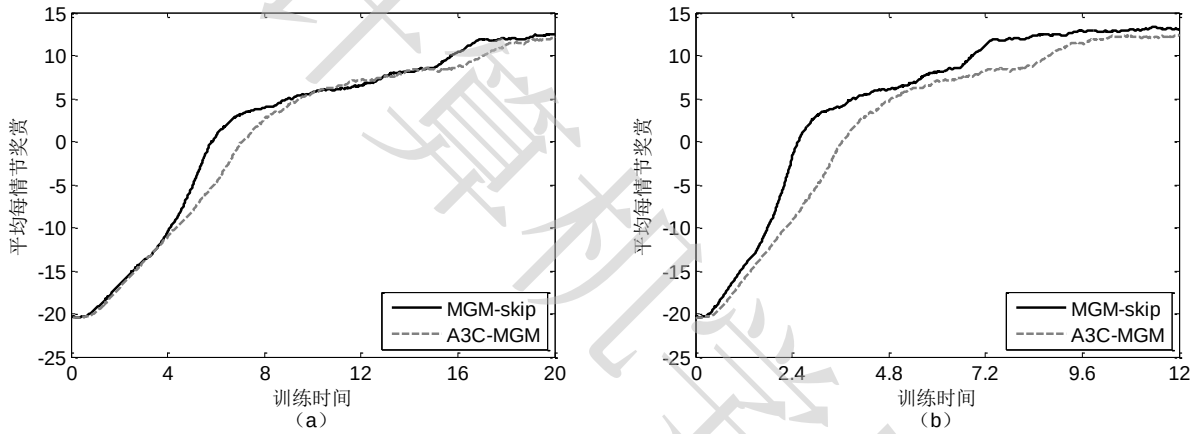


图 11　在 Pong 游戏中，MGM-skip 与 A3C-MGM 在 CPU 和 GPU 上分别训练 20 小时和 12 小时的训练效果。横坐标单位是小时（h）。（a）：在 Intel Core i7-6800k CPU 上的训练效果；（b）：在 NVIDIA TITAN X (PASCAL) GPU 上的训练效果

5 结束语

现代计算设备大多都具有并行计算的能力。虽然基于前馈网络的异步深度强化学习可以利用多线程技术以及并行计算加速模型训练，但是由于其网络结构自身固有的原因，无法记忆不同时间尺度状态之间的依赖信息。而循环神经网络，如 LSTM，可以有效的解决这一问题，但是由于其计算依赖于上一个时间步的输出信息，所以无法利用并行化计算加速模型训练。为了能够利用并行化计算处理状态之间存在依赖关系的任务，本文提出了一种基于多重门限机制的异步优势行动者-评论家算法（A3C-MGM）和基于跳跃连接的多重门限机制算法（MGM-skip）。通过在前馈神经网络中引入多重门限机制来记忆不同时间尺度的任务状态之间的依赖信息，但是不存在循环结构，使得模型能够利用并行计算加速训练过程。本文通过五个 Atari 2600 游戏，其中包含三个稀疏环境奖赏的游戏和两个环境奖赏稀疏性较弱的游戏，验证了 A3C-MGM 和 MGM-skip 在稀疏环境奖赏游戏中的有效性。并且额外选取了四个 Atari 2600 游戏，进一步测试了新模型的性能。实验结果表明新模型在训练过程中所获得的平均每情节奖赏总体上优于另外几种模型，且该优势在 Battle Zone 游戏和 Tutankham 游戏中表现得尤为明显。本文所用的方法在不同的游戏环境中采用的是同一套参数设定，同一个模型和同一种算法，因此本文提出的基于多重门限机制的异步优势行动者-评论家算法具有较强的泛化能力。

然而，基于多重门限机制的模型在上述类型游戏上的表现稳定性并不是特别好，并且距离专业人类玩家水平还有很大的差距。未来的研究重点是提高基于多重门限机制的模型的稳定性，以及利用人类玩家已有数据提高 Agent 在这类游戏上的性能表现。深度强化学习算法通过结合人类玩家的数据来协助指导 Agent 的训练，使得 Agent 能够利用这些数据获得更好的学习效果，以及更加真实的学习效果，以便未来深度强化学习算法能够更好地应用到真实场景中。

参 考 文 献

[1] Yu Kai, Jia Lei, Chen Yu-Qiang, Xu Wei. Deep learning: yesterday, today, and tomorrow. Journal of Computer Research and Development, 2013, 50(9): 1799-1804 (in Chinese)
(余凯, 贾磊, 陈雨强, 徐伟. 深度学习的昨天、今天和明天. 计算机研究与发展, 2013, 50(9): 1799-1804)

[2] Sutton R S, Barto A G. Reinforcement learning: an introduction. Cambridge, USA: MIT press, 1998.

[3] Gao Yang, Zhou Ru-Yi, Wang Hao, Cao Zhi-Xin. Study on an average reward reinforcement learning algorithm. Chinese Journal of Computers, 2007, 30(8): 1372-1378 (in Chinese)
(高阳, 周如益, 王皓, 曹志新. 平均奖赏强化学习算法研究. 计算机学报, 2007, 30(8): 1372-1378)

[4] Fu Qi-Ming, Liu Quan, Wang Hui, Xiao Fei, YU Jun, Li Jiao. A novel off-policy Q(λ) algorithm based on linear function approximation. Chinese Journal of Computers, 2014, 37(3): 677-686 (in Chinese)
(傅启明, 刘全, 王辉, 肖飞, 于俊, 李娇. 一种基于线性函数逼近的离策略 Q(λ)算法. 计算机学报, 2014, 37(3): 677-686)

[5] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge. Nature, 2017, 550(7676): 354-359.

[6] Mnih V, Kavokcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing atari with deep reinforcement learning//Proceedings of the Workshops at the 26th Neural Information Processing Systems 2013. Lake Tahoe, USA, 2013.

[7] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529-533

[8] Watkins C J C H. Learning from delayed rewards [Ph. D. Thesis]. King's College, Cambridge, England, 1989.

[9] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning//Proceedings of the Association for the Advance of Artificial Intelligence. Phoenix, USA, 2016: 2094-2100.

[10] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay//Proceedings of the International Conference on Learning Representations. San Juan, Puerto Rico, 2016.

[11] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735-1780.

[12] Cho K, van Merri ënboer B, Bahdanau D, et al. On the properties of neural machine translation: encoder–decoder approaches//Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. Doha, Qatar, 2014: 103-111.

[13] Narasimhan K, Kulkarni T D, Barzilay R. Language understanding for text-based games using deep reinforcement learning//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Lisbon, Portugal, 2015: 1-11.

[14] Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs//Proceedings of the Association for the Advance of Artificial Intelligence Fall Symposium Series. Arlington, USA, 2015: 29-37.

[15] Rummery G A, Niranjan M. On-line Q-learning using connectionist systems. Cambridge, UK: Cambridge University Engineering Department, Technical Report: CUED/F-INFENG/TR 166, 1994.

[16] Sutton R S. Generalization in reinforcement learning: successful examples using sparse coarse coding//Proceedings of the Advances in Neural Information Processing Systems. Denver, USA, 1995: 1038-1044.

[17] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning//Proceedings of the International Conference on Machine Learning. New York City, USA, 2016: 1928-1937.

[18] Kakade S. A natural policy gradient//Proceedings of the Advances in Neural Information Processing Systems. Vancouver, Canada, 2001: 1531-1538.

[19] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms//Proceedings of the International Conference on Machine Learning. Beijing, China, 2014: 387-395.

[20] Sutton R S, Mcallester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation//Proceedings of the Advances in Neural Information Processing Systems. Denver, USA, 1999: 1057-1063.

[21] Konda V R, Tsitsiklis J N. Actor-critic algorithms//Proceedings of the Advances in Neural Information Processing Systems. Denver, USA, 1999: 1008-1014.

[22] Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 2016,

17(39): 1-40.

[23] Chung J, Gulcehre C, Cho K H, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling//Proceedings of the Workshops at the 27th Neural Information Processing Systems 2014. Montreal, Canada, 2014.

[24] Xu K, Ba J, Kiros R, et al. Show, attend and tell: neural image caption generation with visual attention//Proceedings of the International Conference on Machine Learning. Lille, France, 2015: 2048-2057.

[25] Van Oord A, Kalchbrenner N, Kavukcuoglu K. Pixel recurrent neural networks//Proceedings of the International Conference on Machine Learning. New York City, USA, 2016: 1747-1756.

[26] Dauphin Y N, Fan A, Auli M, et al. Language modeling with gated convolutional networks//Proceedings of the International Conference on Machine Learning. Sydney, Australia, 2017: 933-941.

[27] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, USA, 2016: 770-778.

[28] Srivastava R K, Greff K, Schmidhuber J. Training very deep networks//Proceedings of the Advances in Neural Information Processing Systems. Montreal, Canada, 2015: 2377-2385.

[29] Hariharan B, Arbeláez P, Girshick R, et al. Hypercolumns for object segmentation and fine-grained localization//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 447-456.

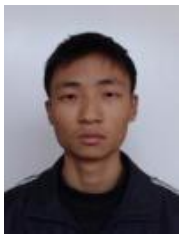
[30] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, USA, 2015: 3431-3440.

[31] Sermanet P, Kavukcuoglu K, Chintala S, et al. Pedestrian detection with unsupervised multi-stage feature learning//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Portland, USA, 2013: 3626-3633.

[32] Yang S, Ramanan D. Multi-scale recognition with DAG-CNNs//Proceedings of the IEEE International Conference on Computer Vision. Santiago, Chile, 2015: 1215-1223.

[33] Kingma D, Ba J. Adam: a method for stochastic optimization//Proceedings of the International Conference on Learning Representations. San Diego, USA, 2015.

[34] Busoniu L, Babuska R, De Schutter B, et al. Reinforcement learning and dynamic programming using function approximators. State of Florida, USA: CRC press, 2010.



XU Jin, born in 1992, Master student. His main research interests include deep reinforcement learning.

LIU Quan, born in 1969, Ph.D. , professor, Ph.D. supervisor. His main research interests include intelligence information processing, automated reasoning and machine learning.

ZHANG Zong-Zhang, born in 1985, Ph.D. , associate professor. His research interests include POMDPs, reinforcement learning and multi-agent systems.

LIANG Bin, born in 1993; Master student. His main research interests include sentiment analysis, natural language processing, and deep learning.

ZHOU Qian, born in 1992, Master student. Her main research interest is reinforcement learning.

Background

Deep reinforcement learning based on experience replay includes deep Q-network, double deep Q-network, deep recurrent Q-network, etc. These methods have achieved unprecedented progress in challenging domains, such as Atari 2600 games, robot control and text recognition. However, experience replay has the following drawbacks: it uses more computation and memory per real interaction; and it requires off-policy learning algorithms such as Q-learning that can update from data generated by an old policy. In order to overcome these drawbacks, asynchronous deep reinforcement learning (ADRL) uses asynchronous methods instead of experience replay. ADRL surpasses the state-of-the-art algorithms, e.g., A3C-FF and A3C-LSTM, on the Atari 2600 games, with only half of computation time on a single multi-core CPU. However, the ADRL based on recurrent neural network (RNN) still needs a lot of training time, because the RNN is unable to be parallelized. To make deep neural networks have the memory and parallelization ability, we propose a novel asynchronous deep reinforcement learning model based on multiple gating mechanisms. Experimentally, our results demonstrate that our new model surpasses the performance of A3C-FF, A3C-LSTM and A3C-GLU on five strategic tasks with sparse rewards among Atari 2600 games.

This paper is partially supported by National Natural

Science Foundation of China (61272055, 61303108, 61373094, 61472262, 61502323, 61502329), High School Natural Foundation of Jiangsu (13KJB520020, 16KJB520041), Natural Science Foundation of Jiangsu (BK2012616), Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (93K172014K04), Suzhou Industrial application of basic research program part (SYG201422, SYG201308). These projects aim to enrich the reinforcement-learning theory and develop efficient approximate algorithms to expand the power and applicability of reinforcement learning on large scale problems.