南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：面向博弈类游戏的深度强化学习算法框架

姓　　名：李亦琛

学　　号：11912433

系　　别：计算机科学与工程系

专　　业：计算机科学与技术

指导教师：程然 副教授

2023 年 6 月 2 日

诚信承诺书

1.本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2.除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3.本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4.在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：

_____年____月____日

面向博弈类游戏的深度强化学习算法框架

李亦琛

（计算机科学与技术系　指导教师：程然）

[摘要] 本文主要介绍了一个基于深度强化学习的面向博弈类游戏的 AI 算法应用框架。深度强化学习是人工智能领域的研究热点，它将深度学习的感知能力和强化学习的决策能力相结合，并在各种游戏任务中取得了实质性的突破。本项目首先实现了经典的深度强化学习算法，包括基于值函数、基于策略梯度以及两者相结合的基于行动者评论家（Actor-Critic，AC）的深度强化学习算法；其次对一些经典的博弈类游戏环境进行了搭建；最后在各种游戏环境中对算法进行了实验测试。本项目设计了一套基于此类算法的面向博弈类游戏的 AI 算法框架，使用者可通过这套框架灵活方便地对一些流行的博弈类游戏进行 AI 算法训练，并得到性能较好的游戏 AI。该框架具有易用、灵活的特点，提供给使用者方便易懂的接口，简化算法部署流程。

[关键词] 人工智能；强化学习；深度强化学习；游戏 AI

[ABSTRACT] This paper concludes a framework for the application of AI algorithms based on deep reinforcement learning for stategy games. Deep reinforcement learning is a hot research topic in the field of artificial intelligence, as it combines the perceptual capabilities of deep learning with the decision-making capabilities of reinforcement learning and has led to substantial breakthroughs in various gaming tasks. The framework developed in this project implements several classical deep reinforcement learning algorithms, such as value-based, policy-based and Actor-Critic(AC)-based deep reinforcement learning algorithms. Moreover, several standard strategy games have been built to be used as testing environments for evaluating the performance of these deep reinforcement learning models. Additionally, experiments have been conducted to show the effectiveness of the algorithms in various game environments. The proposed framework is user-friendly and easy to use. It simplifies the deployment process, making it easier for users to apply deep reinforcement learning algorithms to their games. Furthermore, the framework provides a friendly interface, which allows users to adjust the parameters of the algorithms to better suit their specific gaming needs.

[Keywords] Artificial Intelligence; Reinforcement Learning; Deep Reinforcement Learning; Gaming AI

目录

1. 绪论

1.1 研究背景

博弈类游戏是一种重要的人工智能应用领域，具有许多挑战性的问题。在博弈类游戏中，双方根据游戏规则和环境信息，选择各自的策略以实现最大化的回报。棋类游戏是比较经典的博弈游戏，其规则相对简单，但仍具备很高的复杂度。在行动前，棋手需要对当前状态进行评估并选择最优策略。在处理这种复杂的博弈类游戏时，传统的人工智能算法面临着很多挑战。由于这类游戏中需要处理的信息量非常大，同时游戏状态和策略也因局面而异，因此传统的人工智能算法受到很多限制。

近年来，深度强化学习（Deep Reinforcement Learning，DRL）被大量应用在博弈类游戏中，为游戏智能化提供了强大的支持。由谷歌的 DeepMind 人工智能团队将深度学习（Deep Learning，DL）的感知能力和强化学习（Reinforcement Learning，RL）的决策能力相结合，形成了深度强化学习，并逐渐成为人工智能的热点领域[1]。当前的 DRL 算法主要分为三类，包括基于值函数（value-based）、基于策略梯度（policy-based）以及两者相结合的基于行动者评论家（Actor-Critic，AC）的深度强化学习算法。这些算法在游戏领域中取得了显著的成果，已有许多比较成功的游戏 AI 案例。例如，AlphaGo[2]是基于围棋实现的深度强化学习算法，在 2016 年成功地战胜了人类顶尖的围棋选手。此外，斗地主 AI DouZero[3]在斗地主这种非完全信息博弈游戏中通过深度蒙特卡罗方法进行自博弈，在棋牌类游戏领域取得了显著的进展。

DRL 算法在游戏智能化及人工智能研究领域具有广泛的应用前景和研究价值。DRL 算法不仅可以优化博弈类游戏的体验和玩法，还可以通过研究人类和 AI 在博弈类游戏中的不同表现，更好地理解人类以及 AI 的行为和决策机制。

1.2 研究意义

深度强化学习算法是当前人工智能领域的热点技术之一，其通过从环境中不断获取反馈，达到不断优化自身决策策略的目的。然而，在实际应用过程中，由于数据量大、计算复杂等原因，深度强化学习算法的开发和应用更具挑战性。

为了帮助用户更好地理解和应用深度强化学习算法，本项目设计了一套面向

博弈类游戏的 AI 算法框架。该框架封装了多种深度强化算法，如深度 Q 网络（Deep Q-Network，DQN）、近端策略优化（Proximal policy optimization，PPO）PPO、异步优势行动者评论家算法（Asynchronous Advantage Actor-Critic，A3C）等，可以方便地应用于不同的博弈类游戏环境中。此外，该框架还提供了多种经典的博弈类游戏环境，如五子棋、黑白棋等，供用户使用。

用户可以自定义游戏、算法、模型的组合，根据自己的需求和实际情况，进行训练和测试，并通过设置不同的超参数来进一步提高算法的效果和应用。此框架对使用者上手友好，提供了方便易懂的接口，让用户可以轻松地进行操作。使用者可以使用不同的算法在不同的游戏中进行训练和测试，充分发掘算法的优缺点，提高算法的效果和应用。此外，该框架兼具良好的灵活性和可扩展性，在训练和测试过程中也提供了多项性能指标，如训练进度、所获奖励、损失误差、测试胜率等，以方便用户进行系统性能分析和评估。全面的设计和完备的功能使得该项目具有较好的应用前景，在游戏领域中具有重要的实际意义。

1.3 研究方法

基于深度学习框架 PyTorch，使用 Python 语言进行开发。对前沿的深度强化学习算法进行学习并复现，设计框架结构以及统一的函数接口，并进行相关实验测试。

本文对 DRL 算法、设计的算法框架做了详细的阐述，并对该框架中各个 DRL 算法在游戏环境中进行了测试。本文整体结构如图 1 所示：

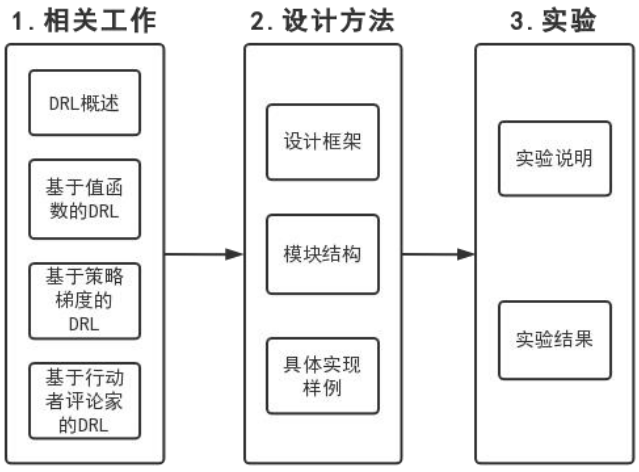

图 1 本文的整体架构

2. 相关工作

2.1 深度强化学习概述

强化学习 在强化学习中，智能体（agent）与环境（environment）不断交互，并通过最大化所获得的奖励来学习行为方式。RL 的过程如图 2 所示，智能体与环境一直处于交互过程中。智能体在每个时刻 t 获取环境的状态S_t，然后基于其当前的状态信息做出一个决策并输出一个动作A_t。环境会根据智能体的动作反馈相应的奖励R_{t+1}以及下一个状态S_{t+1}，这个反馈会被智能体用来优化其行为，从而更好地适应环境的变化。由此形成了一个不断循环的交互过程。

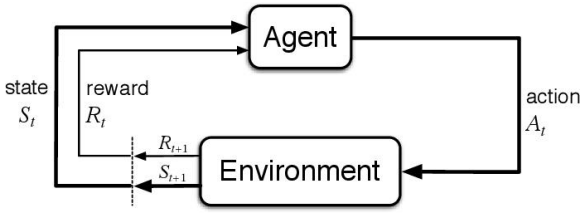

图 2 强化学习过程图[4]

这个过程可被称为马尔可夫决策过程（Markov Decision Process，MDP）。MDP 过程中可以得到一个序列：

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots, s_{n-1}, a_{n-1}, r_n. \tag{1}$$

智能体的目标是最大化长期累积回报，在时刻 t 时，带有折扣因子$\gamma \in [0,1]$的长期累计回报为[5]：

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\,. \tag{2}$$

智能体通过不断学习，找到最优策略π^*后，状态价值函数$V^{\pi}(s)$和动作价值函数$Q^{\pi}(s,a)$分别表示为：

$$V^*(s) = max_{\pi} E[R_t | s_t = s], \tag{3}$$

$$Q^*(s,a) = max_{\pi} E[R_t | s_t = s, a_t = a]. \tag{4}$$

式（3）被称为最优状态价值函数，式（4）被称为最优动作价值函数，均遵循贝尔曼最优方程（Bellman optimality equation），即：

$$V^*(s) = max_a \sum_{s',r} P(s', r | s, a)[r + \gamma V^*(s')]\,, \tag{5}$$

$$Q^*(s,a) = \sum_{s',r} P(s', r | s, a)[r + \gamma max_{a'} Q^*(s', a')]\,. \tag{6}$$

理论上可以通过迭代贝尔曼方程来求解价值函数，通过不断迭代使函数收敛，

最终获得最优策略π^*。但是在实际问题中，采用策略迭代方法需要耗费大量时间和资源，因此通常采用线性函数或深度神经网络等非线性函数来近似估计价值函数。

深度强化学习 深度强化学习结合了深度学习在视觉等方面的感知能力以及强化学习的决策能力，实现了端到端学习。DRL 的出现改善了 RL 的缺陷，使 RL 真正得以广泛应用于实际复杂的问题。从 2013 年 Mnih 等人提出的深度 Q 网络（Deep Q-Network，DQN）算法开创了深度强化学习领域，DRL 领域中出现了大量不断改进的算法。

DRL 有许多分类方式。根据行为策略和需要优化的策略是否相同，分为同步策略（on-policy）算法和异步策略（off-policy）算法；根据是否模拟和学习环境，分为基于模型（model-based）算法和无模型（model-free）算法；根据智能体选择动作的依据，分为基于值函数（value-based)、基于策略（policy-based）以及价值与策略相结合的基于行动者评论家（actor-critic）算法，这是当前主流的分类方式，下文将对这三类算法逐一进行介绍。

2.2 基于值函数的深度强化学习

深度 Q 网络 Q-Learning[6]作为传统 RL 的经典算法，用于状态和动作空间都是离散且维数较小的情况。该算法使用 Q-table 存储每个状态和动作的价值，并通过更新参数来逼近最优 Q 值。如果状态和动作空间是高维的，则可以使用价值函数的近似表示。通过更新参数θ使 Q 函数逼近最优 Q 值，即$Q(s,a;\theta) \approx Q'(s,a)$。

Mnih 等人[7]将强化学习与卷积神经网络相结合，提出了深度 Q 网络（Deep Q-Network，DQN）算法（NIPS 2013）。随后 Mnih 等人[8]对 DQN（NIPS 2013）算法进行了改进，提出了 Nature DQN（NIPS 2015），分别用两个神经网络来近似表示值函数和产生目标 Q 值。DQN 主要有以下两处改进：

DQN 算法在训练过程中通过使用经验池（experience replay）来解决样本间具有关联性的问题。在每一步获得的转移样本$e_t = (s_t, a_t, r_t, s_{t+1})$存入经验池$D_t = \{e_1, \cdots, e_t\}$中，并在训练时随机采样小批量转移样本，采用随机梯度下降（Stochastic Gradient Descent，SGD）方式更新神经网络参数θ。

DQN 算法使用$Q(s,a;\theta_i)$来表示当前 Q 值网络的输出，用来生成动作并且需要更新。除此之外，还使用了另一个网络$Q(s,a;\theta_i^-)$来表示目标 Q 网络的输出，

则目标 Q 值即值函数的优化目标为 $Y_i = r + \gamma max_{a'} Q(s', a'; \theta_i^-)$。通过最小化当前 Q 值与目标 Q 值间的均方误差来更新网络参数θ。误差函数为：

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}[(Y_i - Q(s, a; \theta_i))^2], \tag{7}$$

其中γ是折扣因子，θ_i和θ_i^-分别是当前 Q 网络和目标 Q 网络在第 i 轮更新时的参数。目标 Q 网络参数θ_i^-每隔一段时间更新为θ_i，可以减少两者的相关性。

深度双 Q 网络　Hasselt 等人[9]提出的深度双 Q 网络（Deep Double Q-Network，DDQN）算法对 DQN 进行了改进。由于 DQN 算法中目标 Q 值$Y_i = r + \gamma max_{a'} Q(s', a'; \theta_i^-)$采用贪婪法，选择使 Q 值最大所对应的动作，虽然可以让 Q 值快速向优化目标靠拢，但会产生过度估计问题，导致结果有较大的偏差。DDQN 算法通过解耦目标 Q 值动作选择和目标 Q 值策略评估的方法来消除过度估计的问题。

在 DDQN 中有两套不同的参数，分别是θ_i和θ_i^-。其中θ_i用来选择使得 Q 值最大的动作，θ_i^-用来估计该动作的 Q 值。DDQN 算法的其他流程与 DQN 一样，只在计算目标 Q 值不同。DDQN 算法的目标 Q 值计算公式为：

$$Y_i^{DDQN} = r + \gamma Q(s', argmax_a Q(s', a; \theta_i); \theta_i^-). \tag{8}$$

基于优先级采样的深度双 Q 网络　DQN 通过使用经验池（experience replay）存储转移样本$e_t = (s_t, a_t, r_t, s_{t+1})$来消除样本间的相关性。在每一步训练时，从经验池中以等概率随机采样小批量转移样本用来训练。然而，这种算法并未考虑样本的重要性。Schaul 等人[10]在此基础上提出基于优先级采样的深度双 Q 网络（Double DQN with proportional prioritization），对样本采样方法进行改进。

该算法将样本的时间差分（Temporal Difference, TD）误差作为优先级标准进行采样。TD 误差为 $r + \gamma max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)$。TD 误差绝对值越大，代表网络的预测进度还有更多上升空间，反向传播作用也越大，优先级也应越大。在采样过程采用随机比例化（stochastic prioritization）来确保采样的概率在样本优先级中是单调的，同时保证最低优先级的样本概率是非零的。定义了样本的采样概率为$P(j) = p_j^\alpha / \sum_k p_k^\alpha$。$p_j$表示样本的优先级，指数$\alpha$决定使用多少优先级。当$\alpha = 0$ 时即为均匀分布。

另外，由于优先级采样方式引入了偏差，改变了分布，从而影响了最终收敛

的结果。该算法使用重要性采样权重（importance-sampling weights）$w_j = \frac{(N*P(j))^{\beta}}{max_i(w_i)}$来修正偏差。

基于竞争架构的深度 Q 网络　Wang 等人[11]对 DQN 网络结构进行了优化，提出了一种竞争网络结构（dueling network）。该网络结构将 Q 网络分为两个部分，即将深度卷积神经网络提取出的特征分流为两个支路，一路是价值函数部分，仅与状态 S 有关，与采取的动作 A 无关，记为$V(s;\theta,\beta)$；另一路是优势函数，同时与状态 S 和动作 A 有关，记为$A(s,a;\theta,\alpha)$。则最终价值函数可以表示为

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha), \tag{9}$$

其中θ是网络结构公共部分的参数，α是价值函数独有部分的网络参数，β是优势函数独有部分的网络参数。然而，由于上述公式无法辨识最终输出中$V(s;\theta,\beta)$和$A(s,a;\theta,\alpha)$的作用，为了体现可辨识性，最终公式为：

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{|\mathrm{A}|}\sum_{a'} A(s,a';\theta,\alpha)). \tag{10}$$

2.3 基于策略梯度的深度强化学习

蒙特卡洛策略梯度　策略梯度算法通过最大化累积回报不断计算策略参数的梯度并更新策略参数，从而获得最优策略[12]。策略π被描述为一个包含参数θ的深度神经网络。

在完整的一幕中，将所有的状态、动作和奖励组合起来，即为一个轨迹$\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, \cdots, s_{T-1}, a_{T-1}, r_{T-1}, s_T\}$。策略梯度的目标即最大化期望回报为：

$$\overline{R}_\theta = \sum_\tau R(\tau)P(\tau|\theta), \tag{11}$$

则策略梯度表示为：

$$\nabla\overline{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla logP(a_t^n|s_t^n,\theta), \tag{12}$$

利用该梯度调整策略参数θ：

$$\theta \leftarrow \theta + \eta\nabla\overline{R}_\theta. \tag{13}$$

Williams 等人[13]提出蒙特卡洛策略梯度（REINFORCE）算法，使用蒙特卡洛（MC）方法估计梯度策略，但产生样本效率较低，且会引入较大的方差。REINFORCE 算法通过在回报中减去基线 b 的方式来降低方差。故最终公式为：

$$\nabla\overline{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} (R(\tau^n) - b)\nabla logP(a_t^n|s_t^n,\theta). \tag{14}$$

近端策略优化 Schulman 等人[14]提出信任区域策略优化（trust region policy optimization，TRPO）算法，通过加入 KL 散度约束来限制新旧策略分布的差异，防止策略参数出现较大改变。此后，Schulman 等人[15]提出近端策略优化（Proximal policy optimization，PPO）算法，分为近端策略优化惩罚（PPO-penalty）和近端策略优化裁剪（PPO-clip）两种。其中 PPO-clip 算法没有引入 KL 散度，其需最大化的目标函数为：

$$L(\theta)=\sum_{(s_t,a_t)} min(\frac{p_\theta(a_t|s_t)}{p_{\theta_{old}}(a_t|s_t)}A(s_t,a_t), clip(\frac{p_\theta(a_t|s_t)}{p_{\theta_{old}}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon)A(s_t,a_t)). \quad (15)$$

2.4 基于行动者评论家的深度强化学习

行动者评论家（actor-critic，AC）算法结合了 value-based 和 policy-based 算法，同时学习价值函数和策略[16]。Critic 负责训练价值函数，actor 则根据 critic 输出的价值函数来训练策略。代表性的 actor-critic 算法有深度确定性策略梯度算法（Deep Deterministic Policy Gradient，DDPG）[17]、异步优势行动者评论家算法（Asynchronous Advantage Actor-Critic，A3C）[18]、双延迟确定性策略梯度算法（Twin Delayed Deep Deterministic policy gradient，TD3）等[19]，这些算法通过不同的方式对价值函数和策略进行学习和更新。其中 A3C 采用了多线程技术，可以异步并行地执行多个 agent，提升了采样效率和训练速度。DDGP 注重探索和利用之间的平衡，利用经验回放和目标网络来保证稳定性。TD3 通过使用两个评论家和延时更新方式来降低近似误差，并添加截断噪声来提高策略的鲁棒性。

3. 设计方法

3.1 设计框架

本文对设计的这套游戏 AI 算法框架进行了详细的阐述。整体架构如图 3 所示。整个框架的设计理念基于软件工程的低耦合高内聚思想。为了提高算法的可维护性和可拓展性，框架将不同部分进行了解耦，方便用户根据自己的需求自定义游戏环境、参数、网络模型和探索方法等。

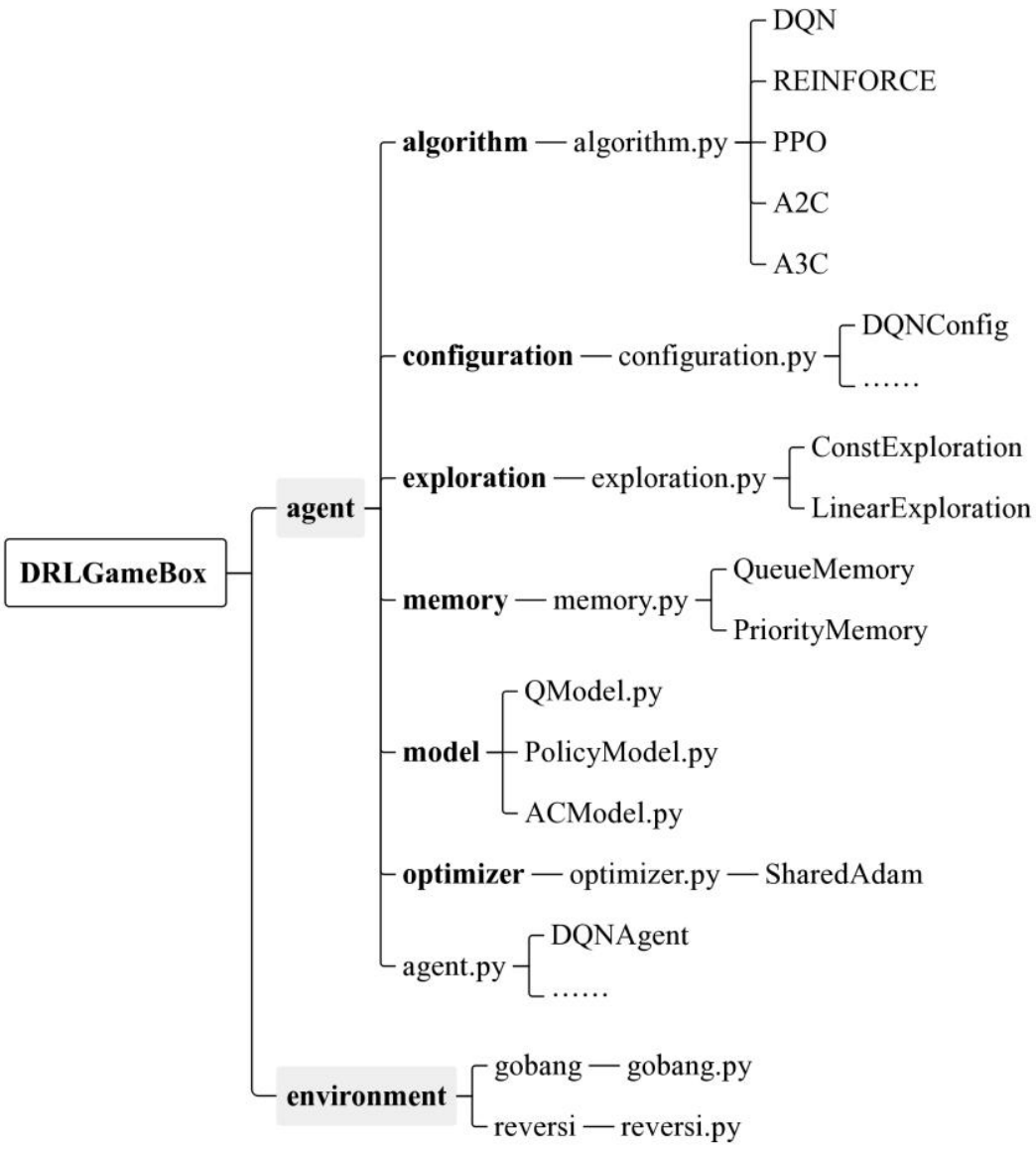

图 3 框架结构图

3.2 模块功能

在框架中，提供了多个相互独立的模块，每个模块都具备不同的内容和功能，详情如表 1 所示。

表 1 模块功能表

算法	内容	功能
Algorithm	DQN 等一系列 DRL 算法类	算法核心
Configuration	DQNConfig 等算法配置类	算法所需的参数配置
Exploration	ConstExploration（常数探索率）类、LinearExploration（线性探索率）类	探索策略
Memory	QueueMemory 类、PriorityMemory 类	经验池
Model	QModel 等一系列网络类	算法所需模型
Optimizer	SharedAdam	为 A3C 算法提供梯度同步
Agent	DQNAgent 等一系列 DRL agent 类	封装 DRL 算法
Environment	Gobang（五子棋）、Reversi（黑白棋）	游戏环境

在框架中，算法（algorithm）部分和经验池（memory）、网络（model）等部分是相互独立的。例如，由于 DDQN 和 DQN 算法仅在算法中计算目标 Q 值时存在不同，因此这两种算法可以共用同一份算法代码，用户可以通过 bool 超参数传入决定是否采用 DDQN 算法。由于 Prioritized DQN 与 DQN 最主要的区别是对经验池（memory）的处理，在算法中仅存在一些改动，因此将 memory 模块与 algorithm 模块进行了解耦。同时用户还可以根据自己的需求来实现不同的经验池。框架同时支持了 Dueling DQN 算法，这个算法与 DQN 的区别在于神经网络架构（model）部分。为了方便用户根据自己的需求来选择神经网络架构，框架将 model 模块与 algorithm 模块进行了解耦。

不仅如此，针对每种算法，框架还提供了不同的配置类（configuration）供用户自定义相关参数或使用默认参数。这些配置类可以帮助用户更加方便快捷地配置自己所需的算法、模型和环境等。

游戏环境（environment）的设计则参考了 OpenAI 的 Gym 库。Gym 库提供常用的强化学习环境，为用户开发和比较强化学习算法提供了便利，同时也提供

一个共享的接口，允许用户编写通用的算法。在 environment 模块中，五子棋（gobang）和黑白棋（reversi）均采用了 Gym 库中的统一接口，包括重置游戏（reset）、执行动作并返回环境信息（step）以及渲染（render）函数。

3.3 具体实现样例

下图 4 为用 Prioritized Dueling DDQN 算法训练一个棋盘大小为 15×15 的五子棋 AI 的代码样例（未使用默认配置参数）。

```
eval_model = LinearQModel(
    input_dim=15 * 15 * 2, hidden_dim=512, output_dim=15 * 15, dueling=True
)
target_model = LinearQModel(
    input_dim=15 * 15 * 2, hidden_dim=512, output_dim=15 * 15, dueling=True
)
exploration = LinearExploration(
    init_epsilon=1.0, min_epsilon=0.1, epsilon_decay=0.995
)
config = DQNConfig(
    eval_model=eval_model,
    target_model=target_model,
    ckpt_path="../checkpoint/ddqn_duel_prior_gobang_n15_mlp.ckpt",
    batch_size=32,
    lr=0.0001,
    gamma=0.99,
    target_replace_frequency=100,
    max_grad_norm=0.5,
    ddqn=True,
    exploration=exploration,
    memory=PriorityMemory(capacity=50000),
)
agent = DQNAgent(config)
do_exp(
    agent=agent,
    env=WrapperGoBang(15),
    train_epochs=1000,
    test_freq=50,
    test_n=50,
    data_path='../data/ddqn_duel_prior_gobang_n15_mlp'
)
```

图 4 使用 Prioritized Dueling DDQN 算法训练五子棋 AI 的具体实现样例

4. 实验

4.1 实验说明

本实验旨在利用该深度强化学习框架对五子棋智能体进行训练，并通过评估比较各个算法的胜率，来测试框架的可行性并探究不同强化学习算法的效果。在该实验中，我们选用了棋盘大小为 15×15 的五子棋环境。在训练时，将棋盘转化为黑方视角，并将黑白棋子分别编码至第一层和第二层，形成环境状态。

训练和测试 在训练过程中，每个智能体均进行了 1000 轮训练，并每隔 50 次训练进行一次评估，记录其与随机算法智能体对打 50 局的胜率。为了增加算法探索能力，在训练时，每个算法的动作选择都进行了探索。在测试时，我们只选择价值或概率最大的动作。

奖励设置 在游戏结束前每走一步给予−1 的奖励，最后一步导致平局则奖励为−0.2，赢了则奖励为 0。

参数配置 在本实验中，均采用了 Adam 优化器进行神经网络的训练，网络均使用 MLP 模型。在 DQN 系列算法中，在训练过程中使用了ε-贪婪法，探索率ε从最初的 1 经过逐轮学习线性下降到 0.1，其余参数如表 2 所示。在 REINFORCE 算法中，设置学习率 lr 为 0.001，折扣因子γ为 0.99。对于 PPO 算法，使用表 3 中的参数。对于 A2C 和 A3C 算法，设置学习率 lr 分别为 0.000005 和 0.0001，折扣因子γ均为 0.99。

表 2 DQN 系列算法参数配置

Algorithm	learning rate	gamma	batch size	capacity	target replace frequency	max grad norm
DQN	0.001	0.99	32	5000	100	1.0
DDQN	0.001	0.99	32	5000	100	1.0
Prioritized DDQN	0.0001	0.99	32	5000	100	1.0
Dueling DDQN	0.00001	0.99	32	5000	100	0.5
Prioritized Dueling DDQN	0.000005	0.99	32	50000	100	0.5

表 3 PPO 算法参数配置

Algorithm	actor learning rate	critic learning rate	gamma	clip epsilon	gae lambda	batch size	learning epochs	update steps
PPO	0.0003	0.0003	0.95	0.2	0.95	32	10	1024

4.2 实验结果

各类 DRL 算法在五子棋游戏环境中训练 1000 轮期间损失函数 loss 值变化如图 5 所示。训练 1000 轮期间每轮获得的奖励 reward 变化如图 6 所示。训练 1000 轮期间与随机算法智能体对打的胜率变化如图 7 所示。

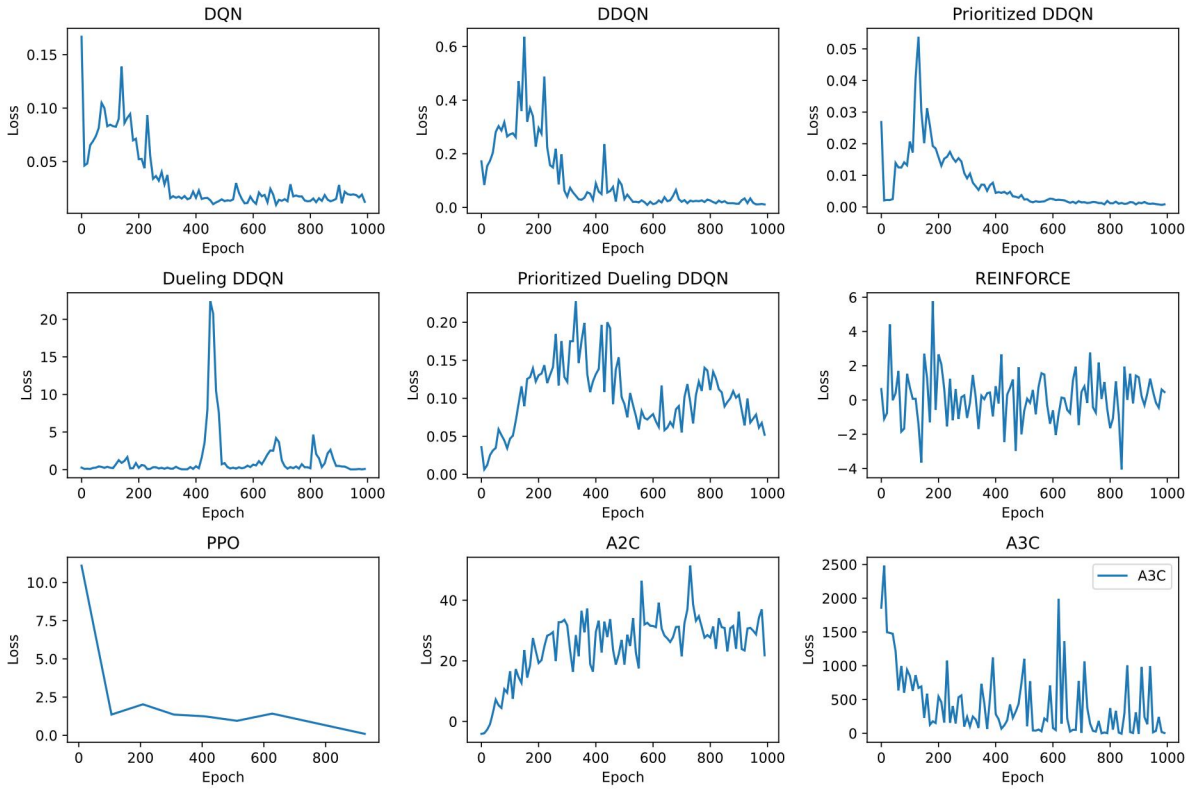

图 5 DRL 算法在五子棋环境中训练的损失

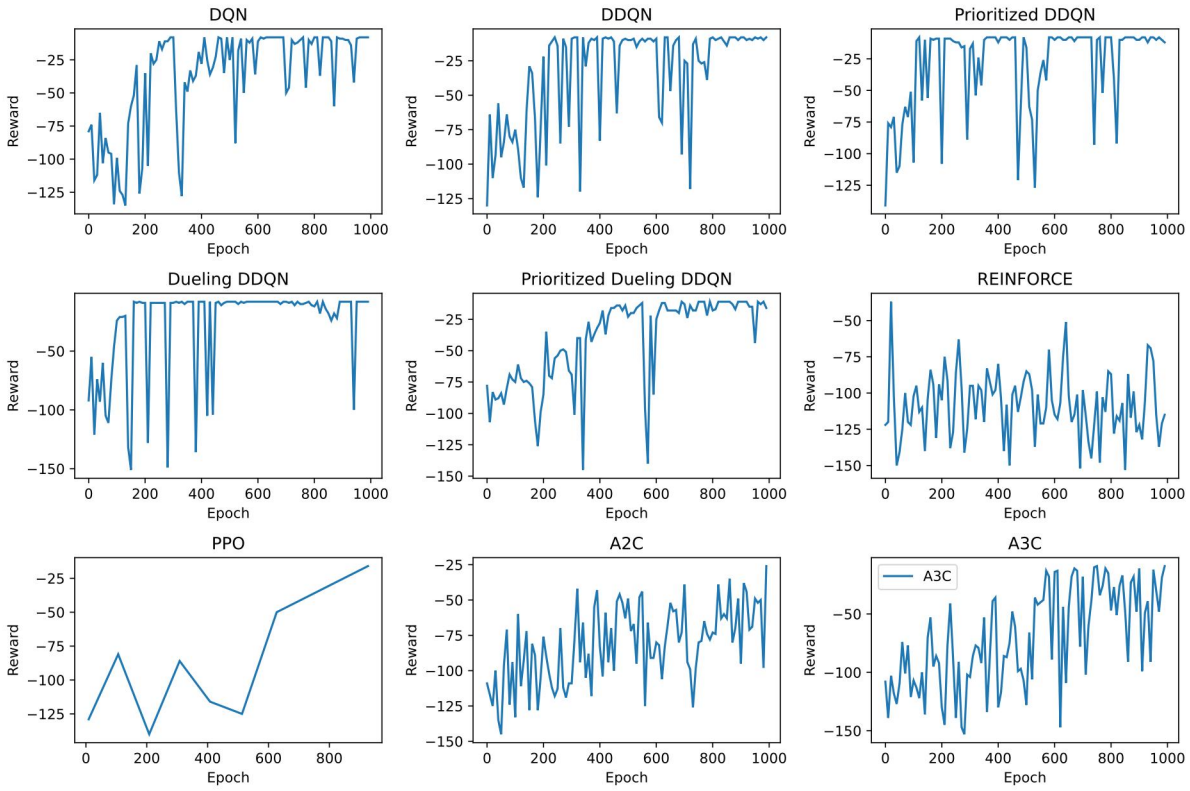

图 6 DRL 算法在五子棋环境中获得的奖励

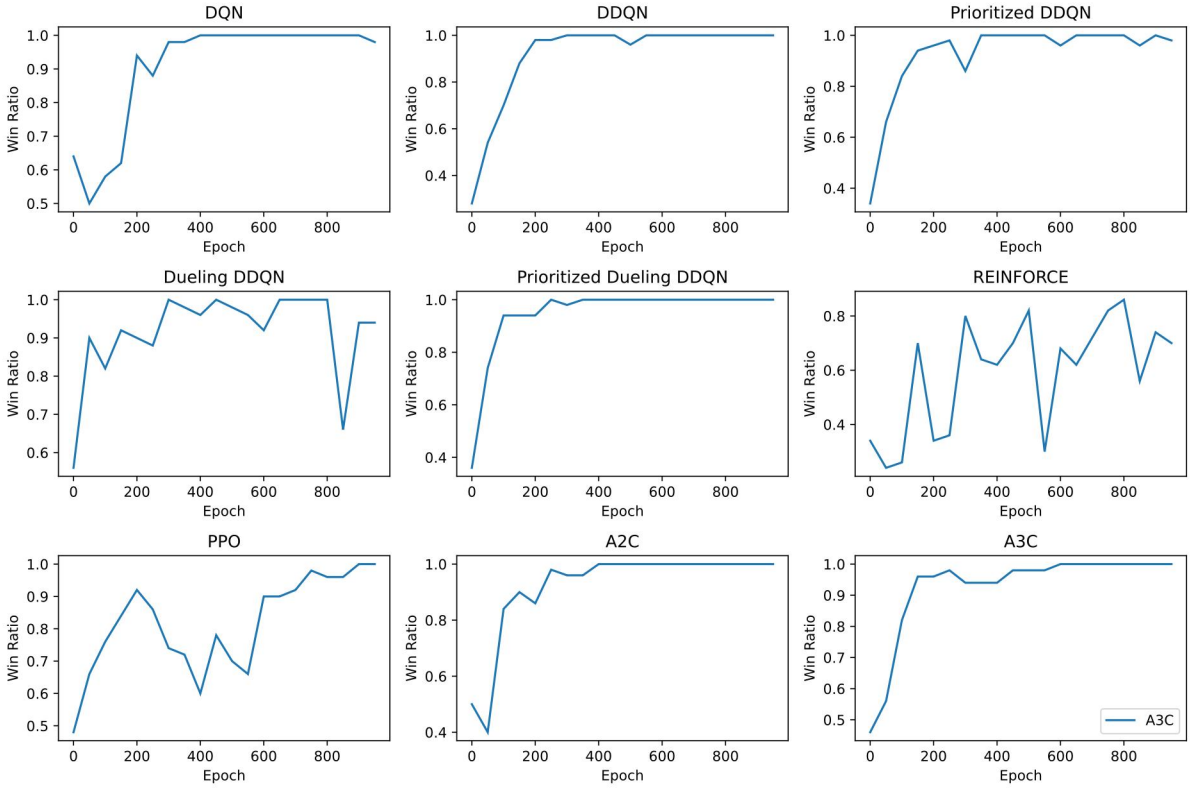

图 7 DRL 算法在五子棋环境中与随机算法智能体对打的胜率

由实验结果可知，所有 DRL 算法均能够成功学习，并且在训练 1000 轮内，能够在和随机算法智能体的 50 局对打中达到 40 局以上的胜场。除 REINFORCE 算法外，其余八种算法均能在训练中后期取得 100%胜率。

根据实验结果观察到，DQN、DDQN、Prioritized DDQN、Prioritized Dueling DDQN、A2C、A3C 算法比较稳定。在实验初期，DDQN 相比于 DQN 收敛更快，即达到 100%胜率速度更快，而 Prioritized Dueling DDQN 与 DDQN 差距不大。Dueling DDQN 训练极其困难，容易发散，最终实验结果也表现一般。最著名的 PPO 算法也在此游戏环境中表现不佳。A2C 和 A3C 均为 AC 架构，尽管 A3C 采用了异步架构，但最终结果差别不大。

在训练过程中，我发现这些 DRL 算法相对于 AlphaZero 等树搜索基础架构[20]在棋类游戏上仍有所欠缺，包括调参困难、奖励函数难以设置、训练过程波动大不稳定等问题。DQL 算法对参数配置极其敏感，例如 DQN 系列的算法在训练五子棋游戏时损失函数容易发散，必须调小学习率，设置梯度裁剪等。同时，奖励函数的设置也尤为关键。在设置奖励函数的实验过程中，我发现采用赢棋奖励 1，其余为 0 的策略会导致训练结果发散，而采用赢棋奖励 0，其余为−1 的策略时能够收敛。此外，DRL 算法不宜使用较复杂的网络结构。在最初的实验中，我使用了一层卷积层，在训练过程中始终发散。而后参考 DouZero 论文实验中 DQN 的网络架构，采用 MLP 模型，调整好参数后可以收敛并能够取得较好的结果。

5. 总结

在本项目中，我基于深度强化学习算法设计实现了一套面向博弈类游戏的 AI 算法框架。通过对前沿 DRL 算法的学习和复现，我深入掌握了经典 DRL 的核心原理，在此基础上进行框架的结构设计和函数接口开发。通过对游戏的实验测试和性能评估，我发现该框架具有良好的应用效果。

本项目的主要意义在于为博弈类游戏的智能化和人工智能研究领域提供了重要的支持和参考。该框架不仅具有较好的灵活性和可扩展性，而且在接口设计方面也提供了方便易懂的操作，使得用户可以更加便捷地应用不同的算法和策略。通过在游戏场景下的实验测试和应用分析，同时也验证了该框架在实际应用中具有较好的效果和可行性。

然而，本项目仍存在一些局限性和不足之处。例如，在算法和性能效率的优化方面仍有待进一步探索和研究。此外，在应用场景和领域方面，也需要在实践中不断完善和拓展，以更好地适应不同的需求和实际情况。

综上所述，本项目提供了一套有益的博弈类游戏 AI 算法框架，为相关研究和应用工作提供了方便和参考。未来，我将继续优化和拓展该框架，并在实际应用中进行进一步探索和验证。

参考文献

[1] 刘全，翟建伟，章宗长，等. 深度强化学习综述[J]. 计算机学报，2018，41(1)：1-27.

[2] Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484-489.

[3] Zha D, Xie J, Ma W, et al. Douzero: Mastering doudizhu with self-play deep reinforcement learning[C]//International Conference on Machine Learning. PMLR, 2021: 12333-12344.

[4] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

[5] 李茹杨，彭慧民，李仁刚，等. 强化学习算法与应用综述 ①[J]. 计算机系统应用.

[6] Watkins C J C H, Dayan P. Q-learning[J]. Machine learning, 1992, 8: 279-292.

[7] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.

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

[9] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2016, 30(1).

[10] Schaul T, Quan J, Antonoglou I, et al. Prioritized experience replay[J]. arXiv preprint arXiv:1511.05952, 2015.

[11] Wang Z, Schaul T, Hessel M, et al. Dueling network architectures for deep reinforcement learning[C]//International conference on machine learning. PMLR, 2016: 1995-2003.

[12] Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation[J]. Advances in neural information processing systems, 1999, 12.

[13] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Reinforcement learning, 1992: 5-32.

[14] Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization[C]//International conference on machine learning. PMLR, 2015: 1889-1897.

[15] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

[16] Konda V R, Tsitsiklis J N. Onactor-critic algorithms[J]. SIAM journal on Control and

Optimization, 2003, 42(4): 1143-1166.

[17] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.

[18] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//International conference on machine learning. PMLR, 2016: 1928-1937.

[19] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods[C]//International conference on machine learning. PMLR, 2018: 1587-1596.

[20] Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.

致谢

首先，我要感谢程然老师在我整个本科期间和毕业设计过程中的悉心指导。是程老师对我的严格要求和高质量标准，让我在不断的修改和完善中提高了论文写作和研究能力。是程老师在毕业设计选题和实验过程中的专业指导和思路引领，让我的毕业论文顺利完成。

同时，我也要感谢 EMI 实验室的全体教师和同学，让我在实验室里获得了宝贵的学习和科研机会。在这里，我收获了许多知识和技能，感受到了学术氛围的独特魅力。特别感谢实验室的同学们，是你们给予了我很多帮助和支持，让我感受到了团队的合作精神和力量。

此外，我也要感谢我的家人和朋友们。你们的关心和支持，一直是我学习和写作中重要的动力来源，是你们的无私支持和鼓励让我在学习和科研之路中倍感温暖。

最后，我要感谢所有为我提供过帮助的人，没有你们的支持和帮助，我的论文将无法顺利完成。在此，向你们表示最诚挚的谢意！