DDD: Double-player Dueling DQN For Fighting Game

陈诺

中国科学院自动化研究所，202118020629011

陈若愚

中国科学院信息工程研究所，202118018629015

摘要—格斗游戏要求智能体基于观测给出合适的攻击动作，以尽可能降低对方的 HP。在该游戏中，对手行为建模的效益不高，且智能体处于不同位置时的决策不一致，故难以套用二人零和博弈求解。鉴于以上特点，本文设计了 DDD 算法，利用对手行为的弱关联性引入“经验池共享机制”，并针对 s、a 在不同位置的作用，设计 Dueling DQN 网络架构，将 Q 函数拆解为状态值 $V(s)$ 和优势函数 $A(s,a)$ 两部分，与格斗游戏的机制较为契合。实验表明，在操纵 ZEN 和 GARNET 两名角色时，该算法能在较长时间内建立稳定的优势，以极高的胜率击败对手。

关键词：格斗游戏；强化学习；Dueling DQN；经验共享

I 背景概述

格斗游戏需要控制智能体完成攻击、防御、位移、大招等高难动作，并试图通过连击打赢对面获胜，凭借其较高的策略性，成为测试强化学习算法的理想环境。格斗过程中，智能体不仅需要根据双方位置确定出招动作，还需合理分配能量以打出强力大招，试图在攻击命中的同时避免受击，以最大化 HP 差值优势。

该问题的难点在于，由于对应的防御策略较少，即便完美预测出对手攻击动作也难以预防，这导致对手行为建模的意义不大，难以套用二人零和博弈的框架求解；此外，状态和决策在不同的位置分别起主导作用，这使得建模价值函数 $Q(s,a)$ 的过程中，需要同时凸显状态值函数 $V(s)$ 的重要性。

针对上述难点，本文采用双玩家经验共享的 Dueling DQN 算法。DDD 利用对手行为重要性不大这一特点，引入双玩家经验共享的机制，使算法能快速学习到有效的出招模式；同时在 DQN 的基础上引入 Dueling 机制，将 $Q(s,a)$ 分解为状态值函数 $V(s)$ 和优势函数 $A(s,a)$ 两部分，分别在远处和近处起主导作用，更符合格斗游戏本身的特性。

下文组织顺序如下：第二部分简要列举与格斗游戏相关的文献，并针对格斗游戏的特点分析其适配性；第三部分将详细阐述基于双玩家经验共享的 Dueling DQN 算法,；第四部分为实验论证，从训练和测试时的性能出发，阐明该算法的优越性。

II 相关工作

格斗游戏中，参与双方均需要击中对面并避免被对面击中，且在一次有效攻击中，攻击方的正反馈与受击方的负反馈保持数值一致。因此，该问题可自然建模为**二人零和博弈**的形式，套用随机博弈的框架求解。

此外，若将对方视为环境的一部分，即假定游戏中仅有一个智能体受到控制，则该问题也可以使用**单智能体强化学习**的思想求解。在这种思路下，智能体依据己方状态作出决策，根据攻击命中伤害和受击伤害，分别获得环境给出的正反馈和负反馈值，并使得指数衰减收益最大化。[1]

在格斗游戏中，针对对手行为采用针对性策略（Counter Strategy）的意义不大。因此本文的主要思路来源于单智能体强化学习，同时巧妙利用了“二人零和”这一特点，引入双方玩家共享经验这一机制，能让智能体快速学习到对方的有效攻击模式。与之相关的文献列举如下：

A. 二人零和博弈

二人零和博弈特指双方构成完全竞争关系的博弈，其显著特点为 $Q_1(s;a_1,a_2)=-Q_2(s;a_1,a_2)$。[2]。从博弈论的视角看，求解该问题的核心在于构建纳什均衡：针对单阶段问题，建立线性规划模型求解纳什均衡；针对扩展博弈问题，求解其子阶段完美纳什均衡。[3]

从强化学习的视角看，二人零和博弈可用随机博弈（Stochastic Game）框架 [4] 描述，即状态转移概率 $p(s'|s,a_1,a_2)$ 由博弈双方的联合决策 (a_1,a_2) 决定。在此框架下，智能体需要**针对对手行为建立模型**，通过预

估对手决策 a_2 的方式，将问题转化为一般的强化学习问题求解。

借鉴博弈论中 Nash 均衡的视角，自然可以给出最保守的对手建模方式，即 Minimax-Q 算法 [5]。其假定对手的行动方式总能让己方收益最小化，以此将 Q-Learning 中的 $\max Q$ 修改为 $\min_{a_2}\max_{a_1} Q$。此外，还可以为对手维护一个假想敌模型，并训练智能体给出针对假想敌的最优反应（Best Responce），此即虚拟博弈（Fictitious Play）框架 [6]。其中一种著名的思路是将假想敌调整为和己方模型一致，即采取自我博弈（self-play）策略，能做到不借助任何先验，训练时完全从零出发，最终得到超越人类专家的性能 [7]。

此外，针对扩展型博弈问题，著名的 CFR 算法 [8] 通过引入反事实遗憾值这一指标，在无限注德州扑克上取得了超过人类的水平；PSRO [9] 通过元策略学习的方式，在策略集合上运用 Double Oracle 算法，有效解决了真实世界游戏中决策集合过大的问题。

B. 单智能体强化学习

在强化学习中，最关键的三要素分别是**状态、决策、收益**。其中状态是智能体作决策的主要依据，其转移过程使用状态转移概率 $p(s'|s,a)$ 刻画；决策是智能体在当前状态下实施的动作，给出所有状态下模型的最优决策 $\pi(a|s)$ 正是强化学习的任务目标；收益 $Q(s,a)$ 则是强化学习需要优化的指标，它反映了智能体在状态 s 下作出决策 a 的长久效益，一般通过指数衰减累积环境反馈的形式计算。

当状态转移概率完全已知时，使用动态规划能保证收敛到全局最优策略。动态规划的核心思想是 Markov 性，即假设智能体的决策与历史信息无关，仅依赖当前状态。在 MDP 假设下，环境动态可用状态转移概率 $p(s'|s,a)$ 完全刻画，且收益函数 $V^\pi(s)$ 和 $Q^\pi(s,a)$ 可通过如下 Bellman 方程计算 [10]：

$$Q^\pi(s,a) = (1-\gamma)r(s,a) + \gamma\sum_{s'} p(s'|s,a)V^\pi(s') \quad (1)$$

当环境动态未知时，可采取蒙特卡罗算法求解。蒙特卡罗的思想是用模拟结果近似代替价值函数中的期望：

$$V^\pi(s) = \mathbb{E}\left[r_0 + \gamma r_1 + \cdots + \gamma^n r_n \middle| s_0 = s\right] \quad (2)$$

通过快速的模拟策略迅速达到游戏终止态后，将最终反馈 r_n 按照衰减指数 γ 反向传回，以更新 $V^\pi(s)$。该算法及其变种 MCTS 常适用于仅在游戏终止时确认胜负的棋牌类游戏，例如围棋、国际象棋等 [11]。

在实际应用中，最常用的是以 Q-Learning 为代表的一步时间差分。时间差分是 Monte Carlo 与动态规划的折中，综合了动态规划的转移与 Monte Carlo 的估计，采用如下方式更新 Q 函数：

$$Q(s,a) \leftarrow Q(s,a) + \alpha\cdot[r + \gamma Q(s',a') - Q(s,a)] \quad (3)$$

当 a' 采取在线学习策略时，即 a 按照当前的决策过程给出，该算法被称为 SARSA；当 a' 按照最大化 Q 函数的形式给出，即 $Q(s',a') = \max_{a'} Q(s',a')$，该算法被称为 Q 学习。

此外，当状态 s 的形式较为复杂，采用表格方式难以枚举 $Q(s,a)$ 的所有值时，常会使用函数拟合的方式近似 Q 函数。当 $Q(s,a)$ 采用深度神经网络进行拟合时，该算法被称为 Deep Q-Learning（DQN）[12]。其核心机制为经验回放（Experience Replay），即在博弈过程中记录所有的四元组 (s,a,r,s') 作为经验，并在经验池中随机采样以消除样本的时间关联性，作为样本优化下列目标函数：

$$\|y - y^*\|_2^2 = \|Q_\theta(s,a) - \left(r + \gamma\max_{a'} Q_{\theta'}(s',a')\right)\|_2^2 \quad (4)$$

其中参数 θ' 所对应的 $Q_{\theta'}$ 称为目标网络，每隔若干步后更新为当前网络的参数 Q_θ。采用 DQN 能从更复杂的状态、决策集中学习 Q 函数，从而极大扩展了强化学习的应用范围。自 DQN 起始，深度学习与强化学习的结合日渐紧密，代表算法有 Double DQN [13] 和 Dueling DQN [14] 等。

III DDD 算法详解

该部分首先对格斗游戏作出全面深入的剖析，并结合游戏本身的特点，将算法定制为双玩家经验共享的 Dueling DQN 算法。

A. 思路分析

在格斗游戏中，博弈双方需要控制相同的人物（ZEN/LUD/GARNET），根据当前的人物状态（站立/浮空/下蹲/倒地），在 40 个指令动作中作出决策。Java 底层中，游戏的攻击命中判定借助碰撞框实现，各个动作均有各自对应的碰撞距离。此外游戏设定了能量机制，通过攻击和受击积攒能量值，并在能量充足时允许释放长距离抛射物（下简称“哈道肯”）。

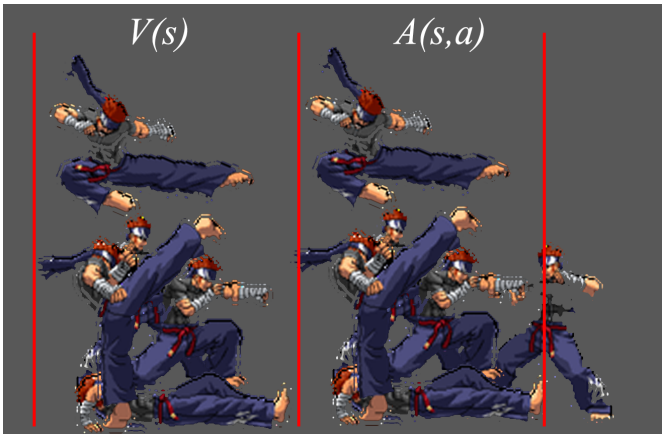

图 1: 状态与决策的影响随双方距离变化的示意图

受决策集限制，格斗游戏有两个显著特点：

- **对手行为建模的效益不明显**。在游戏允许的 40 个决策中，大部分都是攻击动作，而防御措施极少。这意味着即使能预判到对手的动作，也难以给出有效防御，因此诸如虚拟博弈、MiniMax-Q 等基于随机博弈的算法适配性不大。
- **处于不同位置时，智能体的决策重心不一致**。由图1可知，当两个智能体间隔较远时，采取不同的攻击动作没有任何差别，智能体的决策重心应当放在位移上；当距离接近到一定程度时，不同碰撞范围的动作将导出截然不同的反馈，智能体的决策重心应当放在攻击上。

因此，针对第一个特点，本文选择将问题建模为单一智能体的强化学习，并将对手视为环境的一部分，同时利用二人零和这一特性，将对手的经验镜像后共同加入经验池；针对第二个特点，本文采用 Dueling DQN 架构，将 $Q(s,a)$ 拆解为状态值函数 $V(s)$ 和优势函数 $A(s,a)$ 的叠加，不仅能反映出 Q 函数随状态 s 的敏感性，还能显式建模出不同状态下各决策的相对优势，与格斗游戏适配性高。

B. 算法流程

格斗游戏中，智能体可接收到的观测为 144 维向量，可执行的决策为 40 个离散值。该观测中不仅包含智能体状态的刻画，还包含该智能体在当前状态下的决策，故首先将其转换为 $(s_1,a_1;s_2,a_2)$ 的形式，具体方式参见附录 A。

观察到上述形式和经验回放 (s,a,r,s') 极为相似，故在构造经验池时首先保留前后两步的观测 $(s_1,a_1;s_2,a_2)$ 以及 $(s_1',a_1';s_2',a_2')$，每次将 (s_1,a_1,r_1,s_1') 计入经验池。此外，注意到格斗游戏的镜像特性（二人零和博弈、对手行为的关联度低），故引入双玩家经验共享机制，同时将对手的经验 $(s_2,a_2,-r_2,s_2')$ 计入经验池，使得算法能从对手的经验中快速学到进攻策略。

DDD 算法的流程如图2所示。在 Q 函数拟合部分，DDD 通过 Dueling DQN 架构同时拟合状态函数 $V(s)$ 和优势函数 $A(s,a)$，并通过公式(5)计算最终的 Q 值：

$$Q_\theta(s,a)=V(s)+A(s,a)-\frac{1}{|\mathcal{A}|}\sum_{a_i\in\mathcal{A}}A(s,a_i) \tag{5}$$

而在环境仿真与模型训练部分，模型首先从观测中解构出双方的状态与决策 $(s_1,a_1;s_2,a_2)$，并针对当前状态 s_1，利用 Q 函数作出决策 a_1^*：

$$a_1^*=\arg\max_a Q_\theta(s_1,a) \tag{6}$$

将 (s_1,a_1^*) 交由环境仿真模拟，并得到新的观测：$(s_1',a_1';r;s_2',a_2')$。利用双方玩家的镜像性质，不仅将经验 (s_1,a_1^*,r,s_1') 加入经验池，还同时将经验 (s_2,a_2',r,s_2') 加入经验池。

该算法中，Q 函数的学习方式与一般的 DQN 保持一致。Q 函数的拟合目标是 Target 网络的 Q 估值，即 $y=r+\gamma\cdot\max_{a'}Q_{\theta'}(s',a')$，且每隔若干步将 θ' 赋值为当前网络参数 θ。训练按照 MSE 误差进行：

$$l=\frac{1}{|\mathcal{D}|}\sum_{(s,a,r,s')\in\mathcal{D}}\left[Q_\theta(s,a)-r-\gamma\max_{a'}Q_{\theta'}(s',a')\right]^2 \tag{7}$$

算法伪代码如下：

IV 实验分析

由于时间原因，本文仅选用 ZEN 与 GARNET 两个决策进行训练与分析。实验分为三个部分进行：首先记录 DDD 算法训练过程中的性能变化，以此表明算法训练的稳定性与收敛性；之后限定游戏时间 =60s，记录 100 轮游戏中 DDD 算法的胜率、HP 差和秒均伤害，以此反映算法在较长时间段内的稳定表现；最后限定双方 HP 上限为 100，进行 1000 轮游戏并结算每轮结束后的游戏结果，以此验证算法的短期表现是否稳定。

A. 训练过程的性能变化

在训练过程中，己方 AI 与敌方的 HP 差值变化如图3所示。从图3中可看出，DDD 算法的 HP 差值随迭代过程稳定增长，并于迭代中止时稳定于 100 附近，此时 DDD 能以较大的优势战胜对手。

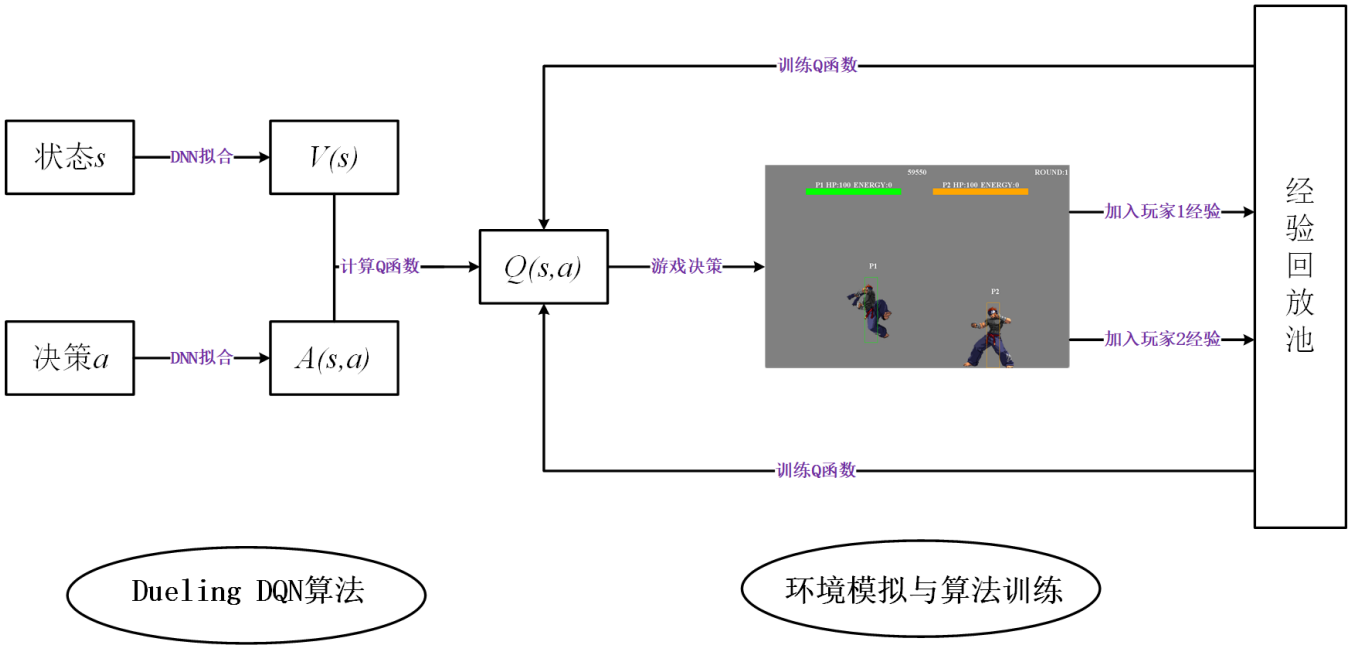

图 2: DDD 算法流程图

Algorithm 1: Double-player Dueling DQN

Initial:
s_1,$Q_{\theta'} \leftarrow Q_\theta$,$a_1 \leftarrow Q_\theta(s_1)$,Memory=[],iter=0;
HyperParams: *update*, *epochs*,γ;
for *epoch* $\leftarrow 0$ **to** *epochs* **do**
  $(s_1', a_1', r, s_2', a_2') \leftarrow$ GetObeservation(s_1, a_1);
  Memory.append$((s_1, a_1', r, s_1'))$;
  Memory.append$((s_2, a_2', r, s_2'))$;
  $(s_1, a_1, s_2, a_2) \leftarrow (s_1', a_1', s_2', a_2')$;
  iter$\leftarrow$*iter* $+1$;
  if *iter* $==$ *update* **then**
    // update target network
    iter $\leftarrow 0$;
    $Q'_\theta \leftarrow Q_\theta$;
  $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}') \leftarrow$ SampleFromMemory();
  $target \leftarrow \boldsymbol{r} + \gamma \cdot \max_{\boldsymbol{a}'} Q_{\theta'}(\boldsymbol{s}', \boldsymbol{a}')$;
  Training Q_θ with loss= $\frac{1}{|\mathcal{D}|}\left(Q_\theta(\boldsymbol{s}, \boldsymbol{a}) - target\right)^2$;

B. 限时 60s 的测试结果

为了消融验证 DDD 模型架构的可行性，另给定 SARSA 算法作为 baseline，它仅仅将 Q 函数改为一层线性映射，训练过程保持不变。限定比赛时间 60 秒，不限制双方 HP 上限，测试 100 轮后得到结果如表I所示：

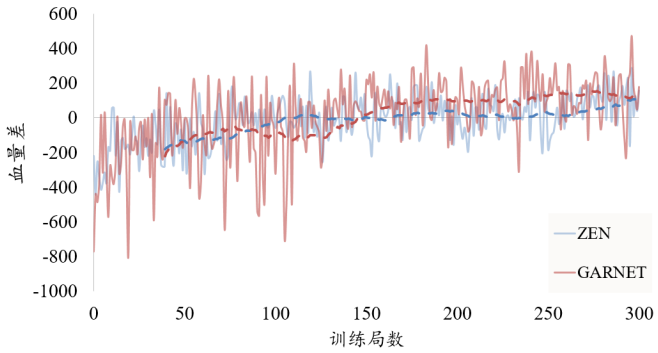

图 3: 训练过程中 AI 的性能变化

表 I: 限时 60 秒时，100 轮对战的测试结果

测试结果汇总						
角色	ZEN			GARNET		
算法	胜率	HP 差	DPS	胜率	HP 差	DPS
DDD	0.87	113.83	6.09	0.93	185.23	10.53
SARSA	0.52	0.81	3.48	0.77	115.16	9.79

其中胜率、HP 差值和 DPS 分别代表 100 轮内对战的平均胜率、与敌方的平均 HP 差距以及己方 AI 的秒均伤害。表I不仅表明 DDD 相比 SARSA 有很大的性能提升，足以验证 Dueling DQN 结构的有效性，还以极高的胜率以及可观的血量差表明，DDD 算法控制的 AI 能稳定击败敌方智能体。

C. HP 上限为 100 的测试结果

为了验证 DDD 算法在短期内的表现是否稳定，限定格斗过程中 HP 上限为 100，此时智能体只需 8 次左右的有效攻击便可击败对方。直方图4记录了 1000 轮测试过程中，游戏结束时双方的血量差，y 轴右侧代表本方获胜，y 轴左侧则代表对方获胜。

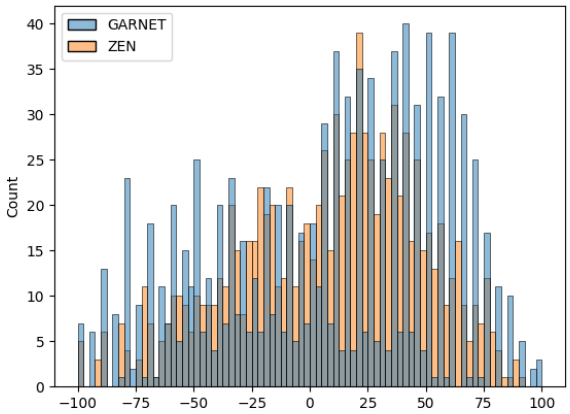

图 4: 1000 轮游戏中双方的游戏结果

从图4中看出，GARNET 和 ZEN 两个角色的重心均往 y 轴右侧偏移，证明即使是较短的时期内，DDD 算法仍能建立一定的优势。然而，y 轴左侧的数据同样较为密集，表明算法的短期性能仍有较大幅度的波动。

V 总结

本文针对格斗游戏的特色，设计了 Double-player Dueling DQN 算法，在敌方受蒙特卡罗搜索树控制时，能稳定取得最终胜利。该算法的核心点在于，虽然鉴于对手建模收益较低的特色，并未使用二人零和博弈框架描述并求解问题，但仍然巧妙利用该特性设计了经验池共享机制。

这对其它问题的算法设计有一定启发意义：简单却有效的建模往往基于对问题描述的透彻理解，并能将问题本身的特性加以利用。此外，单智能体 RL 和二人零和博弈的本质区别在于**“对手建模的实际意义”**，例如围棋、象棋等游戏中，己方决策极度依赖对手历史决策，故采用二人零和博弈中的 self-play 等方式求解；然而格斗游戏中缺少针对性的防御机制，因此将对手视为环境的一部分则更为合理。

该算法仍然暴露出一些问题，例如短期内的性能波动仍然不可控，对能量的把控较为混乱，以及训练过程中的 loss 并不收敛。可行的后续改进措施有：增加深度学习模型的参数、针对能量调配设定专有机制、针对智能体的当前状态排除不可用决策等。

参考文献

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction(2nd edition)*. MIT Press, 2018.

[2] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint*, 2021.

[3] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[4] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[5] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163. Elsevier.

[6] George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[7] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[8] Noam Brown and Tuomas Sandholm. Libratus: The superhuman ai for no-limit poker. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 5226–5228, 2017.

[9] Marc Lanctot, Vincius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Prolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4190–4203. í é.

[10] Nan Jiang Alekh Agarwal and Sham M. Kakade. *Reinforcement Learning: Theory and Algorithms*. Working Draft, 2019.

[11] D Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, Gvd Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *Arxiv pre-print*, 2013.

[13] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning, 2016.

[14] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning, 2016.

附录 A：观测、状态与决策详解

经过对 Java 代码作解包分析，可得出决策名称及其对应索引如下。

索引	java 序数	动作名	指令序列	耗能
0	31	AIR_A	A	
1	32	AIR_B	B	
2	53	AIR_D_DB_BA	214A	5
3	54	AIR_D_DB_BB	214B	50
4	49	AIR_D_DF_FA	236A	
5	50	AIR_D_DF_FB	236B	
6	33	AIR_DA	2A	
7	34	AIR_DB	2B	
8	51	AIR_F_D_DFA	623A	
9	52	AIR_F_D_DFB	623B	
10	39	AIR_FA	9A	
11	40	AIR_FB	9B	
12	41	AIR_UA	8A	
13	42	AIR_UB	8B	
14	8	BACK_JUMP	7	
15	4	BACK_STEP	454	
16	29	CROUCH_A	2A	
17	30	CROUCH_B	2B	
18	37	CROUCH_FA	3A	
19	38	CROUCH_FB	3B	
20	11	CROUCH_GUARD	1	
21	3	DASH	656	
22	7	FOR_JUMP	9	
23	2	FORWARD_WALK	6	
24	6	JUMP	8	
25	0	NEUTRAL	-	
26	27	STAND_A	A	
27	28	STAND_B	B	
28	47	STAND_D_DB_BA	214A	
29	48	STAND_D_DB_BB	214B	
30	43	STAND_D_DF_FA	236A	5
31	44	STAND_D_DF_FB	236B	20
32	55	STAND_D_DF_FC	236C	150
33	45	STAND_F_D_DFA	623A	
34	46	STAND_F_D_DFB	623B	
35	35	STAND_FA	6A	
36	36	STAND_FB	6B	
37	10	STAND_GUARD	4	
38	23	THROW_A	4A	
39	24	THROW_B	4B	
/	1	STAND	/	
/	5	CROUCH	/	
/	9	AIR	/	
/	20	DOWN	/	
/	21	RISE	/	
/	22	LANDING	/	
/	25	THROW_HIT	/	
/	26	THROW_SUFFER	/	

其中部分哈道肯类动作需要耗费能量，此处为不完全统计；第一列为动作在 python 中的索引，共有 40 维，与模型输出相对应；第二列为动作在 Java 中的索引，共 56 维，与给定的观测以及 Java 内部的处理方式对应；第三列为搓招顺序，是指令序列在 Java 中的记录方式。

给定的观测则是 144 维向量，其包含的信息如下：

列表索引	观测含义	转化为绝对数值
0	自己的能量	*300
1	自己的 X 坐标	*480,+480
2	自己的 Y 坐标	*640
3	自己的 X 速度	*20
4	自己的 Y 速度	*28
5-8	自己的人物状态	one-hot
9-64	自己的动作	与 java 序数对应
65	自己的残余帧	*70
66	对手的能量	*300
67	对手的 X 坐标	*480,+480
68	对手的 Y 坐标	*640
69	对手的 X 速度	*20
70	对手的 Y 速度	*28
71-74	对手的人物状态	one-hot
75-130	对手的动作	one-hot
131	对手的残余帧	*70
132-137	己方抛射物状态	-
138-143	对手抛射物状态	-

由于格斗游戏具有 Markov 性，即决策仅依赖当前状态而与历史决策无关，因此 s_1, s_2 中不引入过往决策。从中选取与决策高度相关的 $[0 \sim 8, 65 \sim 74, 131, 138 \sim 143]$ 作为状态，且由双方玩家的镜像性质，s_1 的“己方”恰为 s_2 中的“对方”，反之亦然。

附录 B：代码目录详解

代码最核心的部分为算法代码与训练过程，此外还包含游戏本体以及辅助绘图代码，其架构如下所示：

- **格斗游戏本体**。主要由 Java 游戏代码构成，还包括格斗游戏的环境文件 fightingice_env.py 与智能体类 gym_ai.py。
- **algorithm 文件夹**。里面记录了团队两名成员所写代码，包括自己各自写的 Dueling DQN 算法，以及作为 baseline 提供的 SARSA（其实就是刚开始的版本）。
- **test_Duel.py**。这是 DDD 算法的测试代码，通过篡改环境参数，可以达成不同的测试目的。
- **train.py**。这时算法的训练代码，在标准 DQN 的训练基础上引入了 Dueling 和经验池共享两个机制。

- **checkpoint 文件夹**。主要存放训练好的模型文件，ZEN 和 GARNET 各训练了 300 轮以上。

陈诺 中国科学院自动化研究所 21 级直博生，研究方向为多智能体博弈理论。在团队作业中，主要贡献有：撰写实验报告与 PPT；解包并分析 Java 代码；参与最终算法的设计等。

陈若愚 中国科学院信息工程研究所 21 级直博生，研究方向为计算机视觉。在团队作业中，主要贡献有：全程主导算法设计；完成算法训练与测试等。