格斗游戏大作业

目录 CONTENTS

01 环境分析

环境分析

在每一帧，读取两个AI的Key，执行Key得到新的帧，为每个AI设置帧数据（FrameData），启动两个AI线程（notifyAll）并阻塞游戏主线程，AI线程在processing方法中利用帧数据计算Key，等待两个AI线程均计算完成后，恢复游戏主线程（endFrame.notifyAll()）。

站在env的角度，在processing方法中接收到帧数据，抽取obs、计算奖励返回给step方法，将step给出的动作作为决策动作传递给游戏。

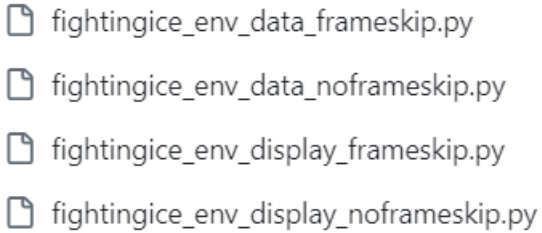
使用日志回放功能进行展示时，直接从日志文件中读取AI的动作，因此AI在做决策阶段所用时间在回放时不可见。

可以使用simulator作为前向引擎，给出帧数据、角色编号、我方动作、对手动作以及模拟的帧数，可以计算得到模拟后的帧数据。

```
public FrameData simulate(FrameData frameData, boolean playerNumber, Deque<Action> myAct, Deque<Action> oppAct,
    int simulationLimit)
```

环境分析

官方提供了四种环境：



- 带有data的obs使用的手工设计的特征，比如我方血量、位置、能量，对手血量、位置、能量等。
- 带有display的obs使用的是某一帧的画面图片。
- frameskip指的是，在env.step之后，返回下一个可以做决策的帧的状态（是否可以控制以及动作的key是否都消耗完毕）。
- noFrameskip指的是，每一帧的状态都返回。

示例代码中提供的是fightingice_env_data_frameskip的环境。

环境分析

University of Chinese Academy of Sciences

motionName	frameNumber	cancelAbleFrame	control
NEUTRAL	6	-1	TRUE
STAND	48	-1	TRUE
FORWARD_WALK	27	-1	TRUE
DASH	29	-1	TRUE
BACK_STEP	25	-1	FALSE
CROUCH	49	-1	TRUE
JUMP	4	-1	FALSE
FOR_JUMP	4	-1	FALSE
BACK_JUMP	4	-1	FALSE
AIR	21	-1	TRUE
STAND_GUARD	23	-1	TRUE
CROUCH_GUARD	23	-1	TRUE
AIR_GUARD	6	-1	TRUE
STAND_GUARD_RECOV	6	-1	FALSE
CROUCH_GUARD_RECOV	6	-1	FALSE
AIR_GUARD_RECOV	6	-1	FALSE
STAND_RECOV	28	-1	FALSE
CROUCH_RECOV	28	-1	FALSE
AIR_RECOV	28	-1	FALSE
CHANGE_DOWN	6	-1	FALSE
DOWN	21	-1	FALSE
RISE	29	-1	FALSE
LANDING	6	-1	FALSE
THROW_A	30	-1	FALSE

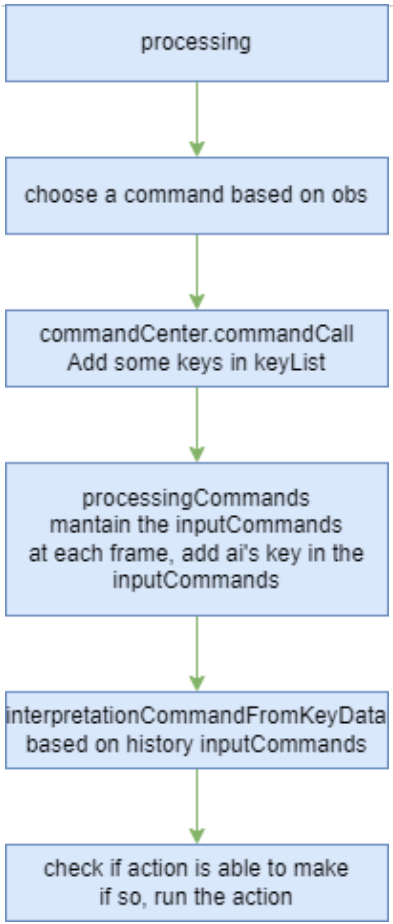
```
this.action = executeAction;
this.state = exeMotion.getState();

if (exeMotion.getSpeedX() != 0) {
    this.speedX = this.front ? exeMotion.getSpeedX() : -exeMotion.getSpeedX();
}
this.speedY += exeMotion.getSpeedY();
this.control = exeMotion.isControl();
Name = executeAction.toString();
```

```
if (this.remainingFrame <= 0) {
    if (this.action == Action.CHANGE_DOWN) {
        runAction(Action.DOWN, resetFlag:true);
    } else if (this.action == Action.DOWN) {
        runAction(Action.RISE, resetFlag:true);
    } else if (this.state == State.AIR || getHitAreaBottom() < GameSetting.STAGE_HEIGHT) {
        runAction(Action.AIR, resetFlag:true);
    } else if (this.state == State.CROUCH) {
        runAction(Action.CROUCH, resetFlag:true);
    } else {
        runAction(Action.STAND, resetFlag:true);
    }
}
```

动作具有持续性，一些动作执行之后在一定帧数内无法控制。

环境分析



为了提高效率，仅在每次可以做决策时选择action。

不考虑提前取消动作的情况，这样问题模型得到简化：在每个可以做决策的帧，根据obs选择动作，解析为一系列key，这些key在接下来的时间帧内被依次执行，形成目标动作，然后等待目标动作完全执行完毕，然后重复以上过程。

02 问题求解

问题求解

格斗游戏是一个典型的实时动作游戏，玩家在游戏中选择一定的动作，在规定的时间内击败对方角色，赢得胜利。本任务基于 FightingICE 格斗游戏平台 [1]，以已知的固定 bot “MctsAi” 作为游戏对手，利用课堂上讲述的强化学习方法设计游戏 AI，通过训练学习得到具有一定智能水平的格斗 AI。

作业要求是和一个固定的对手“MctsAi”进行对战，这就将**两人零和马尔可夫博弈问题**简化为**了单智能体的马尔可夫决策过程。**

对手作为环境的一部分，智能体做出动作与环境交互，得到奖励反馈，从而优化自己的行为。

可以使用强化学习的各种算法进行求解，比如DQN、PPO。通过不断学习，智能体可以学会对手动作规律并找到较好的应对，从而打败对手。

只要训练时间足够长，智能体应该可以overfit到固定的对手，从而找到最有效对抗这个固定对手的策略。

问题求解

我们首先尝试使用PPO训练智能体，训练一段时间后，可以很好地打败MctsAi。

```
Using cpu device
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
At the end, own_hp 108: opp_hp 0. you win.
At the end, own_hp 117: opp_hp 0. you win.
At the end, own_hp 142: opp_hp 0. you win.
At the end, own_hp 166: opp_hp 0. you win.
At the end, own_hp 63: opp_hp 0. you win.
At the end, own_hp 289: opp_hp 0. you win.
At the end, own_hp 63: opp_hp 0. you win.
At the end, own_hp 212: opp_hp 0. you win.
At the end, own_hp 232: opp_hp 0. you win.
At the end, own_hp 144: opp_hp 0. you win.
```

03 问题扩展

问题扩展

由于我们选择MctsAi作为固定对手进行训练，因此智能体overfit到了这个对手上，并没有学到更加通用的对抗策略。我们使用可以打败MctsAi的策略和其它Ai进行对抗，发现对抗一些Ai的胜率很低。

Opponent	Win Rate (%)
BCP	100
BlackMamba	0
DiceAI	96
Dora	6
FalzAI	20
HaibuAI	57
JayBot_GM	78
KotlinTestAgent	1
LGIST_Bot	27
MctsAi	86
SimpleAI	30
Thunder	4
Toothless	4
TOVOR	95
UtalFighter	90

问题扩展

我们打算做一个**面对各种对手，都能比较好对抗的通用智能体。**

我们不局限于和固定的MctsAi进行对抗，将问题恢复为**两人零和马尔可夫博弈问题。**

对于固定对手，目标是在游戏结束时和[我方血量-对手血量]尽可能大。在没有一个固定对手时，目标是什么呢？是面对任何对手时，[我方血量-对手血量]都尽可能大吗？这个目标可以实现吗？

假设存在一个策略，可以面对任何对手时，[我方血量-对手血量]都尽可能大，那这个策略和自己对抗呢？如果把这个策略作为固定对手再训练另一个策略对抗它呢？

我们的目标应该是训练一个**混合纳什均衡策略**，即无论面对什么对手，都找不到我的策略的弱点。（类似于剪刀石头布游戏中等概率出剪刀、石头、布的策略）（如果把这个策略作为固定策略去训练另一个策略进行对抗，无论训练多长时间，胜率始终在50%左右）

04 问题扩展求解

问题扩展求解

求解两人零和马尔可夫博弈问题，可以采用基于价值函数的强化学习算法。

可以使用Minmax-Q算法，根据在线观测数据，求解出纳什Q*。然后可以将两人零和马尔可夫博弈问题转换为标准形式博弈，使用线性规划在多项式时间求解出两人零和的纳什均衡动作概率分布。

$$Q_{t+1}(s_t, a_t^1, a_t^2) = Q_t(s_t, a_t^1, a_t^2) + \alpha_t \left[r_t + \gamma \max_{\pi^1} \min_{a_{t+1}^2} \sum_{a_{t+1}^1} \pi^1(a_{t+1}^1 | s_{t+1}) Q_t(s_{t+1}, a_{t+1}^1, a_{t+1}^2) - Q_t(s_t, a_t^1, a_t^2) \right]$$

每一时刻在线观测$(s_t, a_t^1, a_t^2, r_t, s_{t+1})$，更新Q表

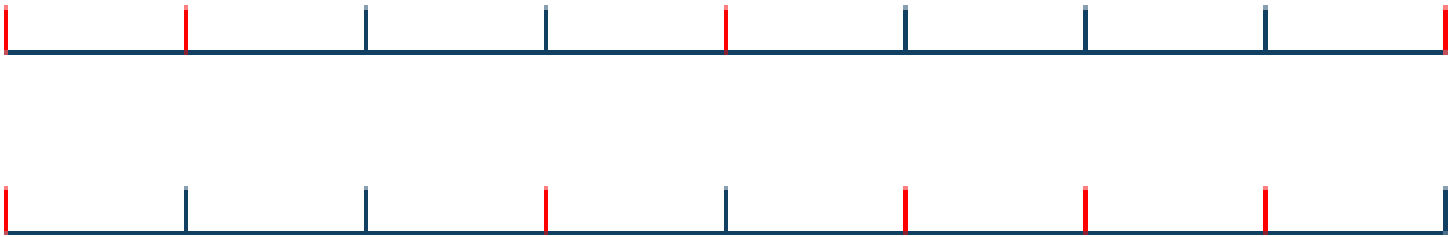
在无限探索、无穷时刻收敛为贪心策略(GLIE)条件下，Minimax-Q 算法收敛到纳什$Q*$

$$\pi_*^1(s) = \arg \max_{\pi^1} \min_{a^2} \sum_{a^1} \pi^1(a^1 | s) Q_*(s, a^1, a^2)$$

问题扩展求解

实际上格斗游戏并不完全符合两人零和马尔可夫博弈模型。

正如环境分析中介绍的，每个人执行某个动作后，在一段时间内可能无法进行控制。在两人零和马尔可夫博弈模型中，在每个时刻（对应游戏中的帧），两人同时做出决策，但在格斗游戏中，可能某一方连续做出多个决策，另一方才做出决策。



问题扩展求解

我们使用基于策略优化的强化学习对问题进行求解。

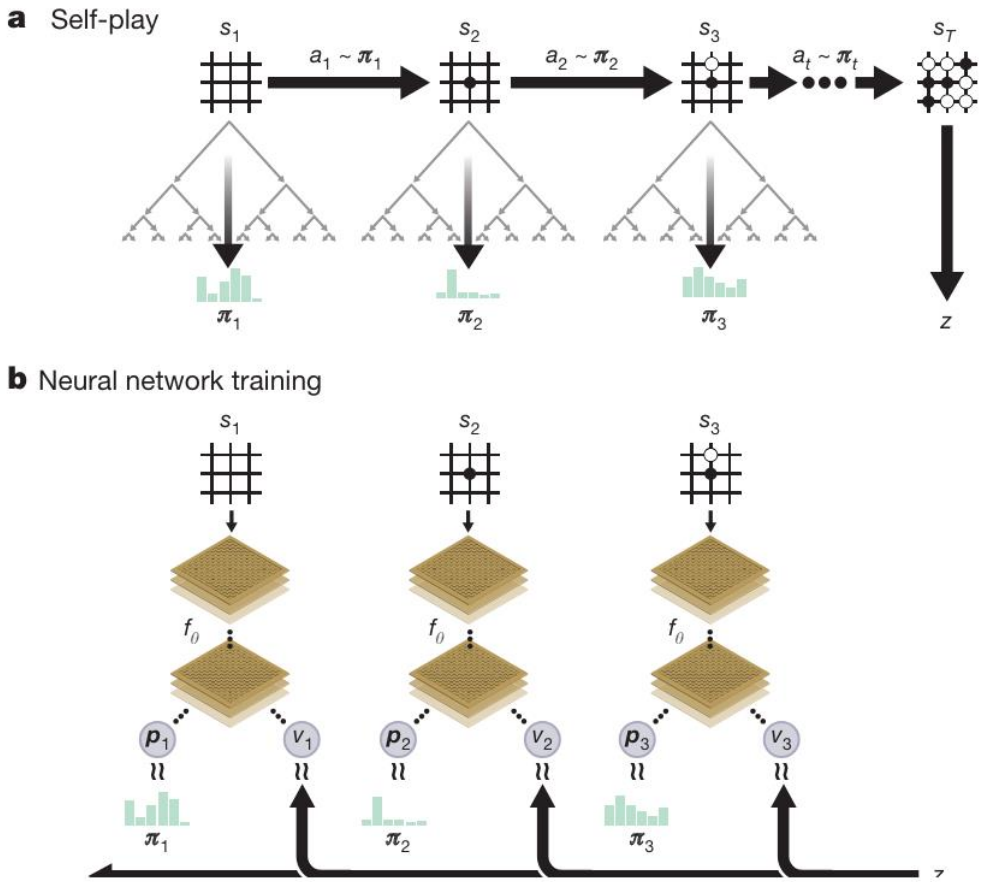
首先尝试的是使用Alpha Zero的蒙特卡洛树搜索 + 自我博弈的方式。

动机：使用策略网络只能根据当前状态得到动作，不能向后推演几步然后再选择最有效的动作。通过蒙特卡洛树搜索可以对未来的进行推演，选择出当前状态最有效的动作。

分为两个阶段：

第一阶段：自我博弈产生训练数据。

第二阶段：使用产生的训练数据对网络进行训练。

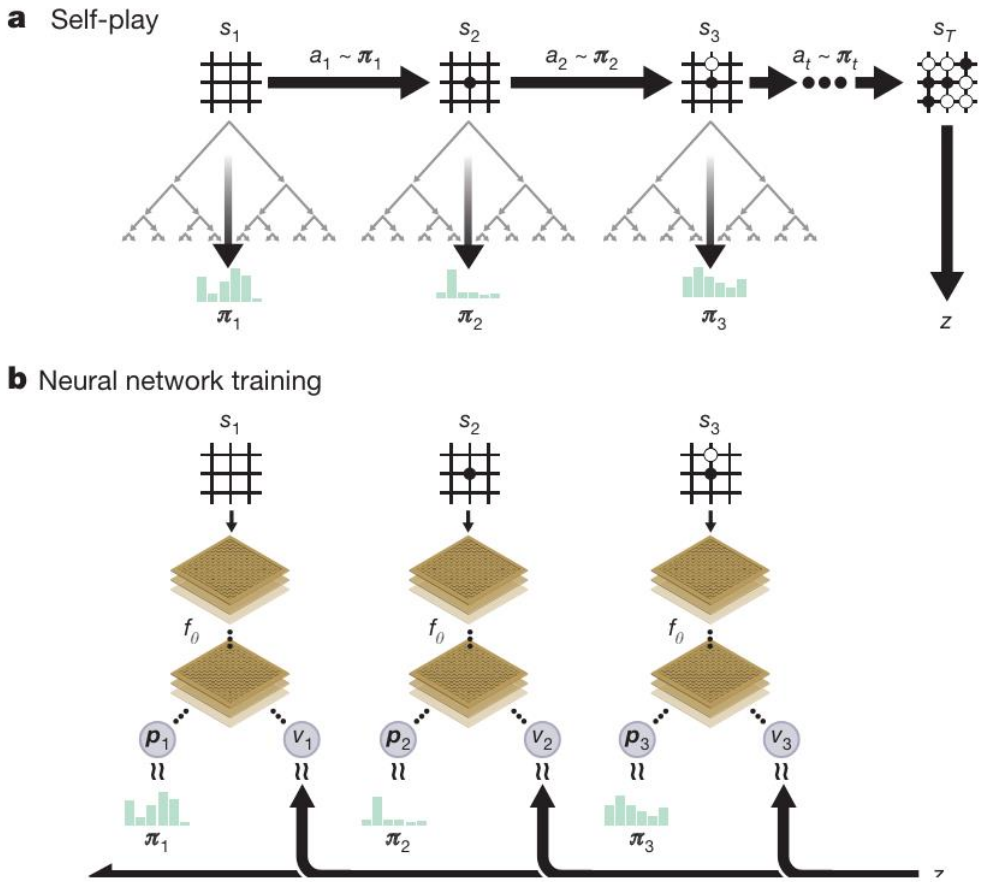


问题扩展求解

阶段一：自我博弈

每一次决策，多次运行树搜索。每次树搜索，将当前状态作为树的根节点，根据UCB选择要执行的动作，在树上遍历，直至到达一个叶节点。使用critic网络计算叶节点的v值，使用actor网络计算叶节点的先验动作概率，创建子节点。将叶节点的v值沿着搜索树进行回溯，更新路径上每个节点的Q值。

多次搜索之后选取根节点访问次数最多的子节点作为要执行的动作。

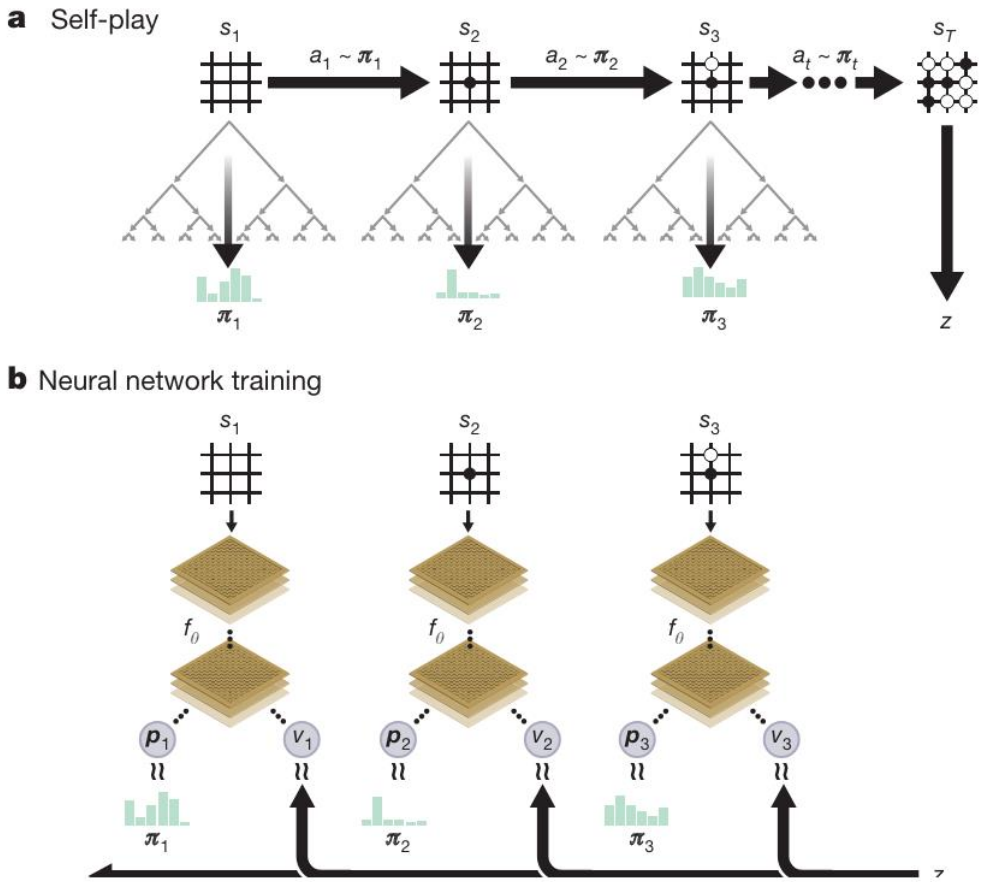


问题扩展求解

阶段二：网络训练

完成一个episode之后，得到最终的episode结果。将整个episode的数据搜集起来（在每个状态下搜索树搜索得到的动作概率分布，以及每个状态对应的最终episode结果）。使用这些数据对critic网络和actor网络进行更新。

网络更新之后和之前最好的网络进行对抗，如果胜率超过某个值，则接受新的的网络。



问题扩展求解

将Alpha Zero应用到格斗游戏中，需要做一些修改：

1. Alpha Zero是为棋类游戏交替决策设计的，而格斗游戏是两人同时决策。为解决这个问题，比较直观的做法是做两棵搜索树，同时进行决策和状态转移。但是这样实现起来比较繁琐，需要主线程和两个搜索树线程之间同步。我们使用了一个简化的做法：在搜索树的每一层（某一方决策，白方或黑方）预测己方动作的同时，使用网络根据对手的观测预测对手的动作。这样做是有一定道理的，因为我们训练网络的目标是为了让网络的输出接近树搜索的输出，因此网络的输出在一定程度上可以代表树搜索的输出。

2. Alpha Zero是为棋类游戏这种只有最后才可以获得奖励信号设计的，但是在格斗游戏中我们每一步都可以获得奖励信号。而且Alpha Zero值网络的输出是游戏的胜率，而格斗游戏中值网络的输出是从当前状态到游戏结束的期望回报。为解决这个问题，我们为树搜索的每一步都添加奖励信号，在回溯时，使用衰减累计奖励更新树上路径的每个点。同样在收集训练数据时，也使用衰减累计奖励作为episode每个状态的目标v值（区别于Alpha Zero各个状态的v值是相同的）。

问题扩展求解

具体实现时，我们使用之前训练的PPO作为critic网络和actor网络的初始化，然后使用simulator进行状态转移模拟，每14帧作为一个决策时刻，将己方动作和对手动作输入到模拟器中得到下一时刻的状态。

我们遇到了一些问题：

1. 预训练的critic网络是评估和MctsAi进行对抗时期望的累计回报，这个值在和自己对抗时与实际值相差极大。
2. 正如之前所说的，整个游戏过程做决策的时刻其实并不“整齐”，并不是理想中的在某一个决策时刻，双方同时做决策然后转移到下一个决策时刻，而是可能某一方连续做了多个决策，另一方才做决策。这就导致了simulator计算的状态并不准确。
3. 与棋盘类模拟相比，使用simulator和java进程进行通信模拟极慢。每个episode有3600帧，每14帧就要模拟一步，每一步要搜索10次，每搜索一次大概也要用simulator模拟一步，这就导致整个训练时间非常长。

问题扩展求解


```
Simulate to the end of the game:
P1 HP:  277
P2 HP:  378
player1_return[0]
-5.460486533179507
V value
6.958555698394775
Self Play:  40%|                                  | 4/10 [29:20<43:39, 436.50s/it]
Player True action: 1
Opponent action: 1
Player False action: 22
Opponent action: 22
Player True action: 6
```

Critic网络输出和实际值差距很大！

模拟时间很长！

反思了一下之前的做法：

1. 格斗游戏和棋盘类问题有很大不同，棋盘永远不会遇到之前的某个状态，但是格斗游戏有可能遇到相同的obs。（比如双方都做成格挡的动作，动作做完之后就又回到一开始的状态）

2. 格斗游戏不需要像棋盘游戏那样推演太深，实际上只要下一步动作能够有较大的奖励即可，没必要从某个状态一直推演到游戏结束最终能收到多少奖励。

问题扩展求解

还是采用自我博弈的方法，没必要做的这么复杂，将自己的之前某个checkpoint塞进环境中和它对抗就可以完成自我博弈。

借鉴了Alpha Star的思想，构建一个对手池，将内置的各个AI放进对手池中，**为了防止过拟合到有限的对手池，在保存checkpoint时，将自己的checkpoint也放进对手池中。**

每次和环境交互时从对手池中选一个对手进行对抗，选择对手的方法借鉴Alpha Star：更倾向于和自己水平相近的对手对抗，同时聚焦在更难的对手上。

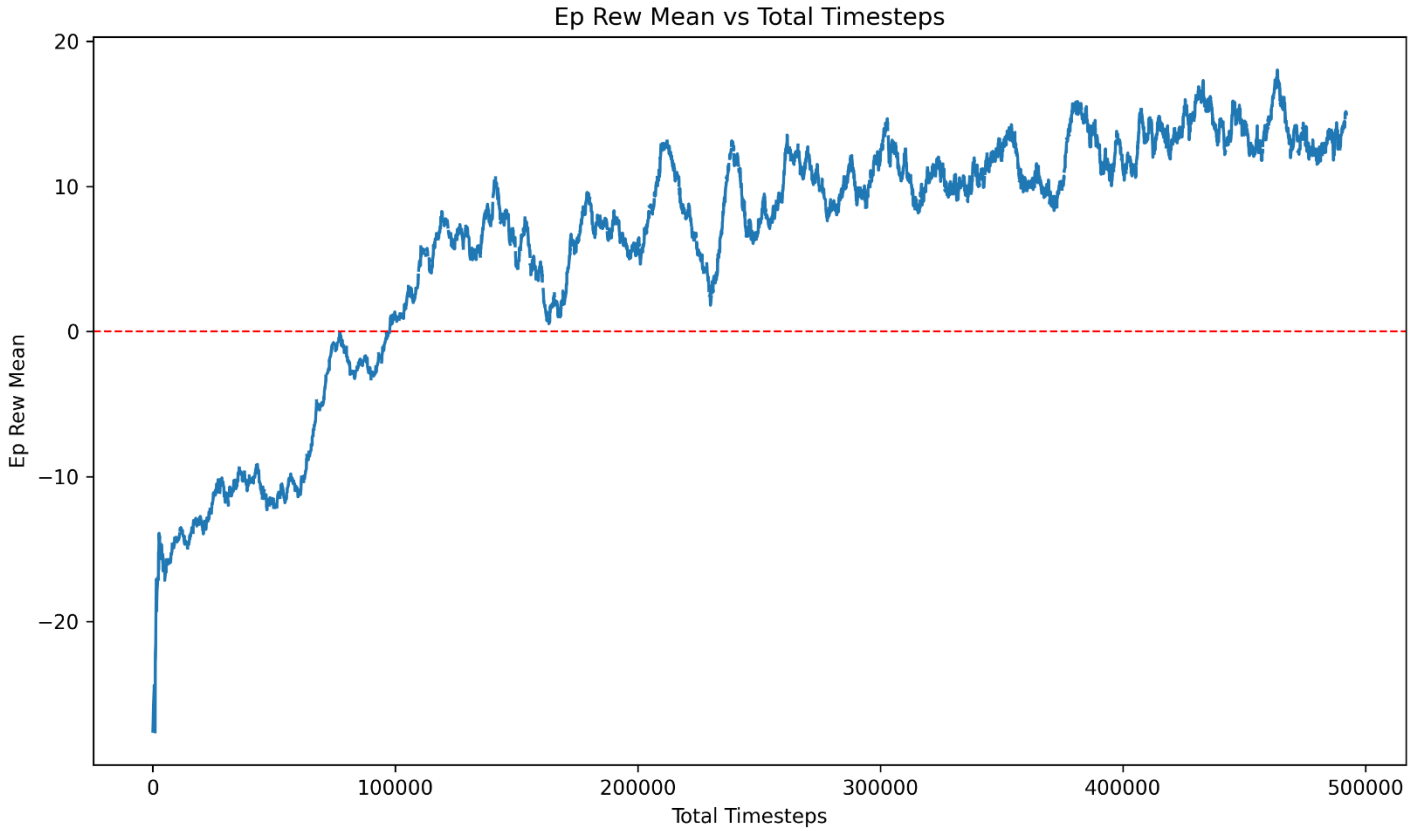
我们一开始忽略了应该和自己水平相近的对抗，而是直接聚焦在困难的对手上。我们使用游戏结束时[对手血量-己方血量]作为对手难易程度的评分，然后使用softmax作为选择对手的概率，导致训练过程十分缓慢。

为了提升样本的利用率，我们换用了off-policy的DQN。训练了50万步左右，基本上可以打败各种内置Ai。

05 效果评估

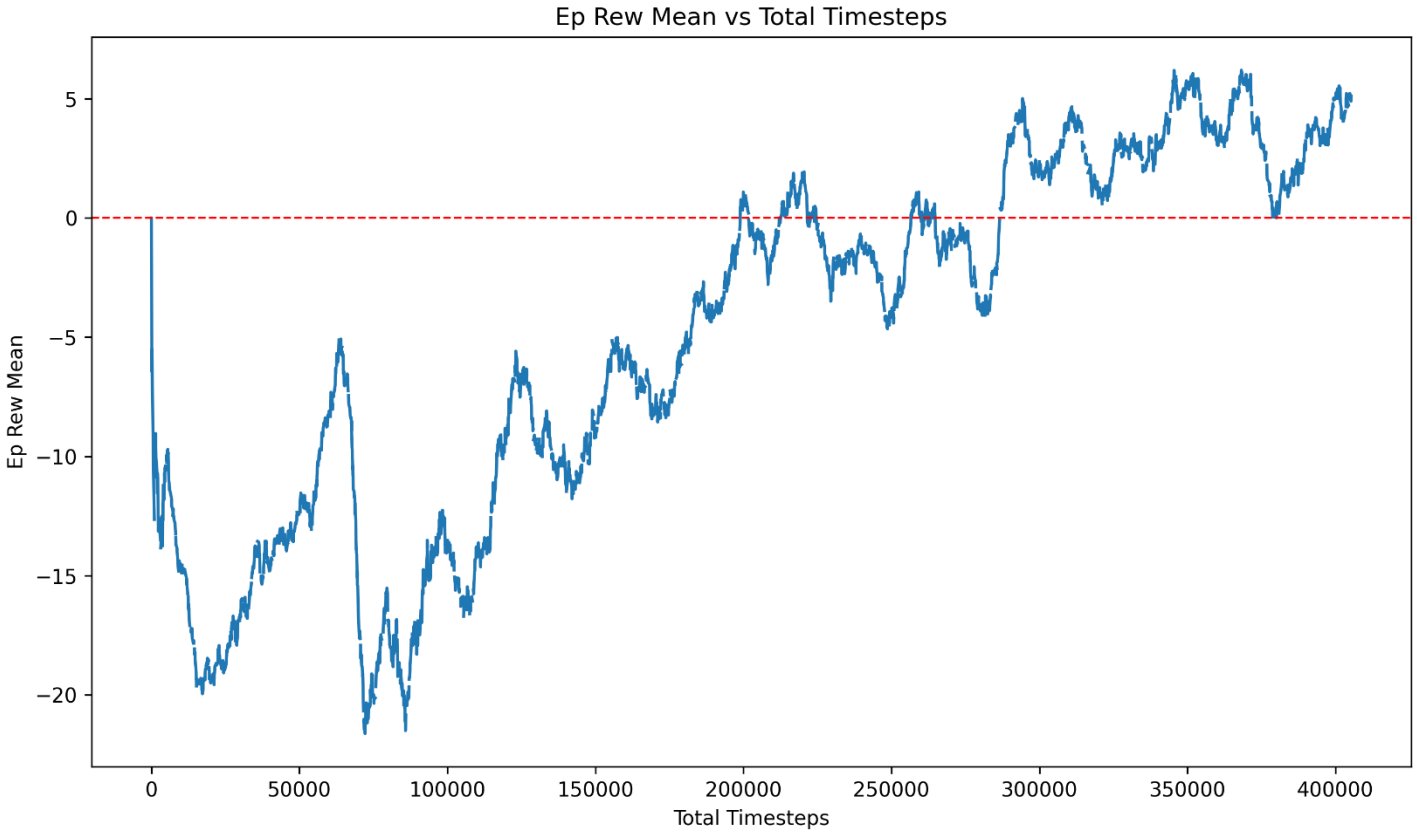
效果评估

对手池DQN训练曲线：



效果评估

对手池（困难对手优先）DQN训练曲线：



效果评估

使用固定对手MctsAi进行训练的PPO，对抗内置各个Ai的胜率统计如下表：

Opponent	Win Rate (%)	Average HP Difference	RemainHP	Advantage	Damage
BCP	100.00	301.52	0.75	0.38	1.00
BlackMamba	0.00	-308.26	0.00	-0.39	0.23
DiceAI	96.00	161.63	0.47	0.20	0.93
Dora	6.25	-205.19	0.04	-0.26	0.45
FalzAI	20.00	-112.56	0.29	-0.14	0.43
HaibuAI	57.14	31.46	0.46	0.04	0.62
JayBot_GM	77.55	66.46	0.20	0.08	0.97
KotlinTestAgent	1.01	-234.95	0.02	-0.29	0.39
LGIST_Bot	27.08	-77.64	0.05	-0.10	0.75
MctsAi	85.57	130.27	0.36	0.16	0.97
SimpleAI	30.00	-47.91	0.32	-0.06	0.56
Thunder	4.04	-159.98	0.28	-0.20	0.32
Toothless	4.08	-268.68	0.01	-0.34	0.32
TOVOR	94.79	132.94	0.56	0.17	0.77
UtalFighter	89.80	166.87	0.45	0.21	0.97

效果评估

使用对手池训练的DQN，对抗内置各个Ai的胜率统计如下表：

Opponent	Win Rate (%)	Average HP Difference	RemainHP	Advantage	Damage
BCP	100.00	264.31	0.66	0.33	1.00
BlackMamba	0.00	-278.92	0.00	-0.35	0.30
DiceAI	100.00	231.23	0.60	0.29	0.98
Dora	98.84	283.24	0.71	0.35	1.00
FalzAI	71.95	77.29	0.25	0.10	0.94
HaibuAI	97.00	158.80	0.44	0.20	0.96
JayBot_GM	100.00	215.79	0.54	0.27	1.00
KotlinTestAgent	98.82	271.99	0.68	0.34	1.00
LGIST_Bot	89.41	147.02	0.40	0.18	0.97
MctsAi	80.00	107.00	0.30	0.13	0.97
SimpleAI	81.25	76.41	0.22	0.10	0.97
Thunder	100.00	228.47	0.57	0.29	1.00
Toothless	100.00	291.92	0.73	0.36	1.00
TOVOR	100.00	226.20	0.59	0.28	0.97
UtalFighter	87.36	114.86	0.30	0.14	0.98

效果评估

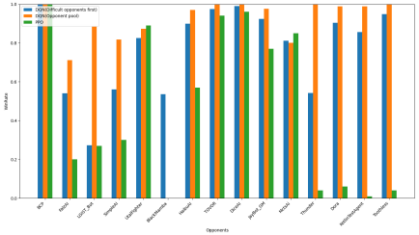
使用对手池（困难对手优先）训练的DQN，对抗内置各个Ai的胜率统计如下表：

Opponent	Win Rate (%)	Average HP Difference	RemainHP	Advantage	Damage
BCP	100.00	213.20	0.53	0.27	1.00
BlackMamba	52.44	9.32	0.16	0.01	0.86
DiceAI	99.00	184.86	0.56	0.23	0.90
Dora	90.43	146.40	0.42	0.18	0.95
FalzAI	54.12	3.18	0.19	0.00	0.82
HaibuAI	89.90	113.56	0.49	0.14	0.80
JayBot_GM	92.31	164.63	0.43	0.21	0.99
KotlinTestAgent	85.56	149.54	0.44	0.19	0.93
LGIST_Bot	27.27	-53.23	0.09	-0.07	0.78
MctsAi	82.14	105.40	0.29	0.13	0.97
SimpleAI	55.21	8.60	0.20	0.01	0.83
Thunder	54.26	11.90	0.18	0.01	0.85
Toothless	94.79	184.14	0.50	0.23	0.96
TOVOR	97.53	167.83	0.60	0.21	0.82
UtalFighter	82.56	114.36	0.31	0.14	0.97

效果评估

University of Chinese Academy of Sciences

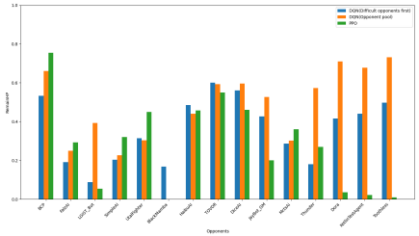
胜率



效果评估

University of Chinese Academy of Sciences

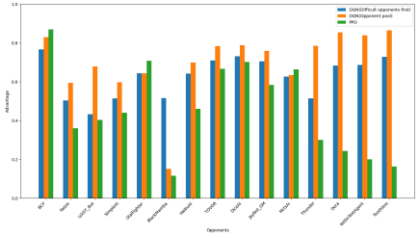
剩余血量



效果评估

University of Chinese Academy of Sciences

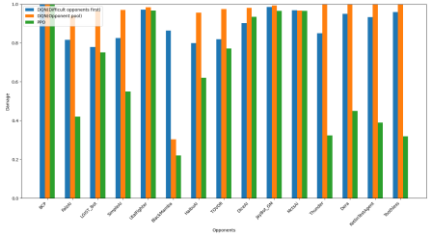
优势



效果评估

University of Chinese Academy of Sciences

伤害



效果评估

为了验证使用对手池训练的智能体是不是学习到了通用的格斗策略，我们在训练时没有将RHEA_PI添加进去，也就是说智能体没有和RHEA_PI的对抗经验，我们尝试测试训练得到的智能体和RHEA_PI对抗，但是无法正常工作。

```
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
fighting against RHEA_PI
fighting against RHEA_PI
fighting against RHEA_PI
fighting against RHEA_PI
fighting against RHEA_PI
```

```python
# just reset is anything ok
self.pipe.send("reset")
self.round_num += 1
# obs = self.pipe.recv()
if self.pipe.poll(100):
    obs = self.pipe.recv()
else:
    # print("fail in reset and let's do it again.'
    obs, _ = self.reset(self.opponent)
# BEGIN MY CODE
# To get the initial frame data
self.init_frame_data = self.engine.frameData
while self.init_frame_data.getEmptyFlag():
    self.step(25)  # NEUTRAL
# END MY CODE
# Need to return info
return obs, {}
```

06 未来工作

未来工作

环境通过Python和Java进程通信运行，收集数据很缓慢，即使使用多进程多环境收集数据仍然很慢。

使用数据增广技巧：在每一步将自己的经验加入经验池的同时，也将对手视角的经验加入经验池。

从我的视角，我能够通过探索学习到一些攻击方式，从对方视角，可以学习到对方的一些有效的攻击方式让自己变得更强。

这样同样的时间可以获得两倍的数据。

未来工作

在测试过程中，我们发现和BlackMamba的对抗胜率很低，我们看了BlackMamba的源码，它是使用一个6层的神经网络，而我们的网络是4层。

所以我们考虑 等优化。

另外我们的自

```python
class CustomFeatureExtractor(BaseFeaturesExtractor):
    def __init__(self, observation_space: spaces.Box, features_dim: int = 256):
        super().__init__(observation_space, features_dim)
        self.layernorm1 = nn.LayerNorm(observation_space.shape[0])
        self.layer1 = nn.Linear(observation_space.shape[0], 256)
        self.layernorm2 = nn.LayerNorm(256)
        self.layer2 = nn.Linear(256, 512)
        self.layernorm3 = nn.LayerNorm(512)
        self.layer3 = nn.Linear(512, 1024)
        self.layernorm4 = nn.LayerNorm(1024)
        self.layer4 = nn.Linear(1024, 512)
        self.layernorm5 = nn.LayerNorm(512)
        self.layer5 = nn.Linear(512, features_dim)
        self.dropout = nn.Dropout(p=0.2)


    def forward(self, observations: torch.Tensor) -> torch.Tensor:
        x = observations
        for i in range(1, 6):
            layer = getattr(self, f'layer{i}')
            layernorm = getattr(self, f'layernorm{i}')
            x = F.leaky_relu(layer(layernorm(x)))
        x = self.dropout(x)
        return x
```

未来工作

尝试不使用内置对手进行训练，只是用自我对抗，测试效果。

```
Saving model_5632 checkpoint to ./checkpoint
Adding model_5632 to the opponent pool
Opponent: model_0, Win: 42, Lose: 23, Draw: 1, Avg HP Diff: 35.666666666666664
Opponent: model_512, Win: 18, Lose: 8, Draw: 1, Avg HP Diff: 45.81481481481482
Opponent: model_1024, Win: 62, Lose: 40, Draw: 0, Avg HP Diff: 31.92156862745098
Opponent: model_1536, Win: 7, Lose: 3, Draw: 0, Avg HP Diff: 54.7
Opponent: model_2048, Win: 10, Lose: 4, Draw: 0, Avg HP Diff: 40.642857142857146
Opponent: model_2560, Win: 6, Lose: 2, Draw: 0, Avg HP Diff: 46.75
Opponent: model_3072, Win: 2, Lose: 0, Draw: 0, Avg HP Diff: 82.5
Opponent: model_3584, Win: 2, Lose: 1, Draw: 0, Avg HP Diff: 49.0
Opponent: model_4096, Win: 5, Lose: 3, Draw: 0, Avg HP Diff: 34.0
Opponent: model_4608, Win: 2, Lose: 0, Draw: 0, Avg HP Diff: 78.5
Opponent: model_5120, Win: 2, Lose: 1, Draw: 0, Avg HP Diff: 38.666666666666664
Opponent: model_5632, Win: 0, Lose: 0, Draw: 0, Avg HP Diff: 0
Fighting model_3072, At the end, own_hp 215: opp_hp 105. you win.
```

感谢