

电子科技大学  
计算机科学与工程学院

标准实验报告

(实验) 课程名称 人工智能

电子科技大学教务处制表

# 电子科技大学

# 实验报告

学生姓名：朱若愚 学号：2022150501027 指导教师：段立新

实验地点：A2-412

实验时间：2024.5.26

一、实验室名称： 计算机学院实验中心

二、实验项目名称：强化学习实验

三、实验学时：5 学时

四、实验原理：

## 1、腾讯开悟平台简介

开悟是腾讯牵头构建的 AI 多智能体与复杂决策开放研究平台，依托腾讯 AI Lab 和「王者荣耀」在算法、算力、实验场景方面的核心优势，为学术研究人员和算法开发者开放的国内领先、国际一流研究与应用探索平台。开悟平台使用《王者荣耀》的游戏环境进行 AI 模型的训练。

“王者荣耀”是一款在手机上游玩的多人在线战斗竞技场(MOBA)游戏。在“王者荣耀”中，玩家使用移动按钮来控制英雄的移动，并使用其他按钮来控制英雄的普攻和技能。除了通过操作主界面观察周围环境，玩家还可以通过左上角小地图了解全地图情况，其中可观察的炮塔、小兵和英雄以缩略图的形式显示。游戏存在战争迷雾机制，只有目前单位属于友方阵营或者处于友方阵营观测范围内才能被观测到。

游戏开始时，每个玩家控制英雄，从基地出发，通过杀死或摧毁其他游戏单位（例如敌方英雄、小兵、炮塔）获得金币和经验，购买装备和升级技能，以此提升英雄的能力。获胜目标是摧毁对手的炮塔和基地水晶，同时保护自己的炮塔和基地水晶。

## 2、强化学习算法原理

- 近端策略优化(Proximal Policy Optimization, PPO)

---

**Algorithm 1** PPO, Actor-Critic Style

---

```
for iteration=1,2,... do
  for actor=1,2,...,N do
    Run policy  $\pi_{\theta_{old}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  end for
  Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{old} \leftarrow \theta$ 
end for
```

---

[https://blog.csdn.net/weixin\\_44436360](https://blog.csdn.net/weixin_44436360)

---

**Algorithm 1** Proximal Policy Optimization (adapted from [8])

---

```
for  $i \in \{1, \dots, N\}$  do
  Run policy  $\pi_{\theta}$  for  $T$  timesteps, collecting  $\{s_t, a_t, r_t\}$ 
  Estimate advantages  $\hat{A}_t = \sum_{t' > t} \gamma^{t'-t} r_{t'} - V_{\phi}(s_t)$ 
   $\pi_{old} \leftarrow \pi_{\theta}$ 
  for  $j \in \{1, \dots, M\}$  do
     $J_{PPO}(\theta) = \sum_{t=1}^T \frac{\pi_{\theta}(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t - \lambda \text{KL}[\pi_{old}|\pi_{\theta}]$ 
    Update  $\theta$  by a gradient method w.r.t.  $J_{PPO}(\theta)$ 
  end for
  for  $j \in \{1, \dots, B\}$  do
     $L_{BL}(\phi) = -\sum_{t=1}^T (\sum_{t' > t} \gamma^{t'-t} r_{t'} - V_{\phi}(s_t))^2$ 
    Update  $\phi$  by a gradient method w.r.t.  $L_{BL}(\phi)$ 
  end for
  if  $\text{KL}[\pi_{old}|\pi_{\theta}] > \beta_{high} \text{KL}_{target}$  then
     $\lambda \leftarrow \alpha \lambda$ 
  else if  $\text{KL}[\pi_{old}|\pi_{\theta}] < \beta_{low} \text{KL}_{target}$  then
     $\lambda \leftarrow \lambda / \alpha$ 
  end if
end for
```

---

[https://blog.csdn.net/weixin\\_44436360](https://blog.csdn.net/weixin_44436360)

Advantage Actor Critic (A2C) 结合基于值的方法和基于策略的方法的混合体系结构,通过减少方差来帮助稳定训练:包含控制智能体行为的演说家 Actor(基于策略的方法)和衡量所采取的行动有多好的评论家 Critic (基于价值的方法)。

在 Actor-Critic 框架下,考虑 PPO,近端策略优化(Proximal Policy Optimization, PPO)算法原理:通过避免过大的策略更新来提高智能体的训练稳定性。为此,我们使用一个比率来表示当前策略和旧策略之间的差异,并将该比率剪切到特定范围 $[1-\epsilon, 1+\epsilon]$ 。这样做可以保证我们的策略更新不会太大,训练更加稳定。

近端策略优化(PPO)的思想是,我们希望通过限制每个训练时期对策略的更改来提高策略的训练稳定性:我们希望避免进行太大的策略更新。

有两个原因:

- (1) 根据经验,我们知道训练期间较小的策略更新更有可能收敛到最优解。
- (2) 在策略更新中迈出太大的一步可能会导致从悬崖上掉下来(得到一个糟糕的策略),并且需要很长时间甚至没有恢复的可能性。

所以使用 PPO,我们会保守地更新策略。为此,我们需要使用当前和以前的

策略之间的比率计算来衡量当前策略与前者相比的变化。并且我们将这个比率固定在一个范围内 $[1-\epsilon, 1+\epsilon]$ ，这意味着我们消除了当前策略偏离旧策略太远的动机(因此有了“近端策略”术语)。

在强化中优化的目标：

$$L^{PG}(\theta) = E_t[\log \pi_{\theta}(a_t|s_t) * A_t]$$

其中 $\log \pi_{\theta}(a_t|s_t)$ 表示在当前状态 $s_t$ 下采取行动 $a_t$ 的概率的 log 值,  $A_t$ 表示优势, 如果 $A > 0$ , 表示在当前状态下该行动要优于其他可能的动作。我们的想法是, 通过对这个函数采取梯度上升步骤(相当于对这个函数的负值采取梯度下降), 我们将推动智能体采取导致更高回报的行动, 并避免有害的行动。

然而, 问题来自于步长: 1) 步长太小, 训练过程太慢; 2) 步长太大, 训练有很大的波动性。

使用 PPO, 我们的想法是用一个称为 Clipped surrogate objective function 的新目标函数来约束策略更新, 该目标函数将使用剪辑将策略更改约束在一个小范围内。这个新函数的设计是为了避免破坏性的大权重更新:

PPO's Clipped surrogate objective function:

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

让我们研究每个部分, 了解它是如何工作的。

(1) 比率函数 (The ratio function)  $r_t(\theta)$ :

这个比率的计算方法如下:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

它是用现在的策略下在状态 $s_t$ 下采取动作 $a_t$ 的概率除以在先前的策略下在状态 $s_t$ 下采取动作 $a_t$ 的概率。 $r_t(\theta)$ 表示当前策略与旧策略之间的概率比:

如果 $r_t(\theta) > 1$ , 那么状态 $s_t$ 下采取动作 $a_t$ 在当前策略中比在旧策略中更有可能。

如果 $r_t(\theta)$ 在 0 至 1 之间, 那么与旧策略相比, 当前策略采取行动的可能性

更小。

因此，此概率比是估计旧策略和当前策略之间差异的简便方法。

(2) 函数中的未裁剪部分 (The unclipped part)  $r_t(\theta)\hat{A}_t$ :

这个比率可以代替我们在策略目标函数中使用的对数概率。这就给出了新目标函数的左边部分:将比率乘以优势。

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t [r_t(\theta)\hat{A}_t]$$

然而，在没有约束的情况下，如果当前策略中采取的行动比以前更有可能，这将导致很大的策略梯度步骤，因此会导致过度的策略更新。

(3) 函数中裁剪部分 (The clipped objective)  $clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)$ :

因此，我们需要通过惩罚导致比率远离 1 的变化来约束这个目标函数(在本文中，比率只能在 0.8 到 1.2 之间变化)。通过裁剪比率，我们确保不会有太大的策略更新，因为当前的策略与旧的策略不能有太大的不同。要做到这一点，我们有两个解决方案:

1) TRPO (Trust Region Policy Optimization, 信任区域策略优化) 利用目标函数外的 KL 散度约束来约束策略更新。但该方法实现复杂，计算时间长。

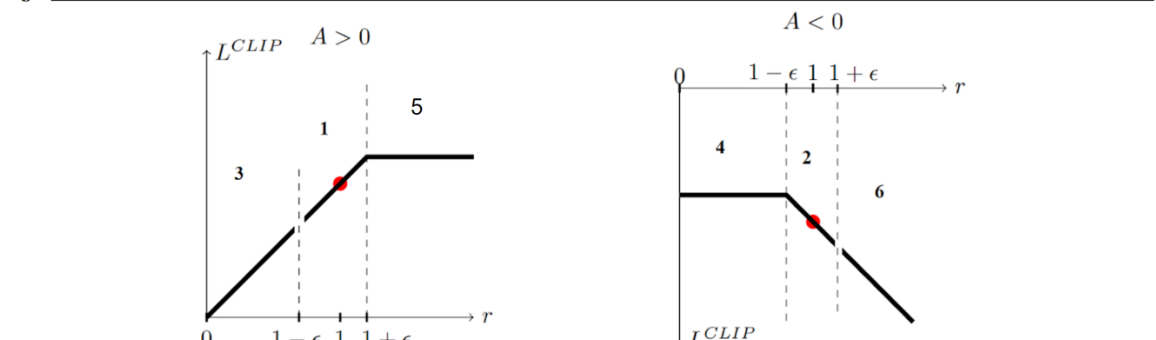
2) PPO 裁剪概率比率直接在目标函数中使用它的 Clipped surrogate objective function。

这个裁剪部分是  $r_t(\theta)$  被夹在  $[1 - \epsilon, 1 + \epsilon]$  之间的一个版本。使用剪切替代目标函数，我们有两个概率比，一个是非剪切的，另一个是剪切在  $[1 - \epsilon, 1 + \epsilon]$  两者之间范围的，Epsilon 是一项超参数，可帮助我们定义此剪辑范围 (在文中  $\epsilon = 0.2$ )。

然后，我们取剪切目标和未剪切目标的最小值，因此最终目标是未剪切目标的下界。取被剪切目标和未被剪切目标的最小值意味着我们将根据比例和优势情况选择被剪切目标或未被剪切目标。

可视化剪切替代目标函数:

	$p_t(\theta) > 0$	$A_t$	Return Value of $\min$	Objective is Clipped	Sign of Objective	Gradient
1	$p_t(\theta) \in [1 - \epsilon, 1 + \epsilon]$	+	$p_t(\theta)A_t$	no	+	✓
2	$p_t(\theta) \in [1 - \epsilon, 1 + \epsilon]$	-	$p_t(\theta)A_t$	no	-	✓
3	$p_t(\theta) < 1 - \epsilon$	+	$p_t(\theta)A_t$	no	+	✓
4	$p_t(\theta) < 1 - \epsilon$	-	$(1 - \epsilon)A_t$	yes	-	0
5	$p_t(\theta) > 1 + \epsilon$	+	$(1 + \epsilon)A_t$	yes	+	0
6	$p_t(\theta) > 1 + \epsilon$	-	$p_t(\theta)A_t$	no	-	✓



因此，我们只用未剪切的目标部分更新策略。当最小值是被裁剪的目标部分时，我们不更新策略权重，因为梯度将等于 0。因此，我们仅在以下情况下更新我们的策略：

- 1) 比值在 $[1 - \epsilon, 1 + \epsilon]$ 这个范围内。
- 2) 我们的比率在范围之外，但优势使我们更接近这个范围：①低于比率，但优势是 $> 0$ ；②超过比率，但优势 $< 0$ 。

总而言之，由于这个剪切替代目标，我们限制了当前策略与旧策略的变化范围。因为我们移除了概率比移动到区间外的动机，因为剪辑迫使梯度为零。如果比率 $> 1 + \epsilon$ 或者 $< 1 - \epsilon$ ，梯度将等于 0。

最终的剪切代理目标损失是这样的，它是剪切替代目标函数、价值损失函数和熵的组合：

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)]$$

## 五、实验目的：

实验要求理解强化学习基本原理并学习强化学习经典算法，尝试使用近端策略优化(Proximal Policy Optimization, PPO)方法基于腾讯开悟平台训练强化学习智能体解决王者荣耀中墨家机关道 1v1 对战问题，理解强化学习算法原理并通过实战部署掌握算法的应用。

## 六、实验内容：

以小组的形式组建战队，在给定的资源下训练并提交一个模型，控制**鲁班七号英雄**进行墨家机关道 1v1 对战。主要考查单智能体解决方案，模型结构设计，强化学习算法设计和训练方式探索。

### 实验准备：

1. 阅读开发指南，熟悉环境介绍（包括观测空间、动作空间、奖励机制）、代码包介绍、训练介绍。

2. 配置环境，跑通源码。

### 实验任务一：王者荣耀 1V1 算法模型设计

#### 1. 实现 Value Loss:

在 Value Loss 的实现中，需要用到的函数有：

TensorFlow 的 `reduce_mean` 函数：计算张量 `tensor` 沿着指定的数轴（`tensor` 的某一维度）上的平均值，主要用作降维或者计算 `tensor` 的平均值。这里 `reduce_mean` 用于消去 batch 维度。

TensorFlow 的 `square` 函数：计算某个元素的平方。这里用于计算均方误差。

涉及的变量有：

**reward**：与环境交互所返回的实际奖励值（一般会使用时序差分方法，即一部分即时奖励加上后续状态价值的估计值），这里作为价值函数的更新目标。

**fc2\_value\_result\_squeezed**：价值网络所输出的对状态的预测价值。

#### 【手动实现】

`model.py` 中的 “`self.value_cost`” 需要手动实现。

根据均方误差实现价值网络的损失函数，其计算公式如下：

$$L_{value} = \frac{1}{N} \sum_{i=1}^N \left( V_{\pi}(s) - \hat{V}(s) \right)^2$$

其中  $N$  为 batch size,  $V_{\pi}(s)$  是状态  $s$  的真实价值，这里使用环境返回的 `reward` 作为估计值。  $\hat{V}(s)$  为价值网络的输出。

#### 2. 实现 Policy Loss:

在 Policy Loss 的实现中，需要用到的函数有：

TensorFlow 的 `exp` 函数：计算某个元素的指数。

TensorFlow 的 `clip_by_value` 函数：将张量中的每一个元素的值都压缩在 `min` 和 `max` 之间。小于 `min` 的让它等于 `min`，大于 `max` 的元素的值等于 `max`。

涉及的重要变量：

`label_list`：六个动作头对应的实际采样动作。

`label_size_list`：六个动作头对应的动作空间。

`legal_action_flag_list`：六个动作头对应的合法动作，元素为 0 表示不合法、元素为 1 表示合法。

`fc2_label_list`：策略网络的输出。这里与 `label_list` 不同，`label_list` 输出的为实际采样的动作，而 `fc2_label_list` 对所有可能采取的动作都输出了一个数值。

### 【操作技巧】

可以利用游戏规则直接屏蔽掉不合理的预测 `action`，比如 CD 中的技能不能释放，进而加快训练速度，避免无意义的探索。在实际代码中，可以使用如下计算技巧：

$$\pi(a|s) = \text{softmax}[\text{NN}(s) - (1 - \text{legal}_{\text{action}}) * 1e10]$$

即对于不合法的动作的输出减去一个较大的值，进而在经过 `softmax` 函数后，使得该动作对应的输出近乎为 0。代码对应于“`legal_action_flag_list_max_mask`”、“`label_logits_subtract_max`”。

这里将 `fc2_label_list` 中的元素减去一个最大值，指数函数的增长速度非常快，从而容易增大数值误差，这里采用一个简单技巧增加数值稳定性。这个技巧基于以下的等式：

$$\frac{e^{x+c}}{e^{x+c} + e^{y+c}} = \frac{e^x e^c}{e^x e^c + e^y e^c} = \frac{e^x e^c}{e^c (e^x + e^y)} = \frac{e^x}{e^x + e^y}$$

根据上式，我们可以减去任意的数值而不改变的 `softmax` 的输出。这里的 `c` 选择为其中的最大值是确保数据稳定性的一个方便的方法。

为方便计算，我们还需要对概率取对数，因此对于所有的输出加了一个较小的值，以防在后续计算对数时出现错误。代码对应于“`label_exp_logits`”。

在 `inference` 时，策略网络会输出所有动作对应的采样概率，而在计算策略梯度时，我们只需要实际采样动作对应的概率值。因此在这里使用了动作的 `one hot` 表示来进行计算。代码对应于“`policy_p`”。



### 【手动实现】

model.py 中的 “final\_log\_p”、“ratio”、“ratio\_mean”、“clip\_ratio”、“surr1”、“surr2” 需要手动实现。下面分别进行介绍。

#### (1) final\_log\_p

final\_log\_p 为 policy\_log\_p 与 old\_policy\_log\_p 的差值，用于之后计算 ratio。

#### (2) ratio

ratio 是重要性权重  $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ ，根据上述计算的 final\_log\_p 进行计算，需要使用到 TensorFlow 的 exp 函数。

#### (3) ratio\_mean

ratio\_mean 为 ratio 的均值，用作统计信息，使用 TensorFlow 的 reduce\_mean 函数进行计算。

#### (4) clip\_ratio、surr1、surr2

clip\_ratio、surr1、surr2 为实现 Dual-clip PPO 所需的核心变量。Dual-clip PPO 在  $\hat{A}_t < 0$  时的目标函数为

$$E[\max(\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t), c\hat{A}_t)]$$

在  $\hat{A}_t > 0$  时的目标函数为

$$E[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t), c\hat{A}_t]$$

这里我们通过 clip\_ratio、surr1、surr2 这三个变量来实现该目标函数。

clip\_ratio: 使用 TensorFlow 的 clip\_by\_value 函数，将张量 ratio 的取值限制在 0.0 到 3.0（这里的 3.0 即为上式中的 c）的范围内。

surr1: 将优势函数 advantage 乘以剪辑系数 clip\_ratio，得到剪辑后的代理目标函数 surr1。

surr2: 使用 TensorFlow 的 clip\_by\_value 函数将 ratio 剪辑到 [1-clip\_param, 1+clip\_param] 的范围内，得到另一个剪辑系数，并将其与优势函数相乘，得到另一个剪辑后的代理目标函数 surr2。clip\_param 可通过 self.clip\_param 获得。

surr1 与 surr2 取最小值即为最终的目标函数。

### 3.实现 Entropy Loss:

在 PPO 中，entropy loss（熵损失）是一种用于增加探索性的正则化项。熵是信息论中的一个概念，表示随机变量的不确定性。在强化学习中，策略可以被看作是一个随机变量，因为它输出一个动作的概率分布。如果策略分布更加确定，那么我们可以更加准确地预测策略的行为。然而，当策略分布过于确定时，它就变得过于保守，可能会错过探索更优的动作。为了平衡探索性和确定性，我们可以使用 entropy loss 作为正则化项，通过最大化策略的熵值来增加策略的不确定性，从而促进更好的探索。

在 PPO 中，entropy loss 通常与 PPO 损失函数中的策略损失项结合使用。通过最小化 PPO 损失函数，我们可以同时实现对策略的更新和探索性的加强。

基于熵的强化学习方法寻找一个最优策略，使得长期累计奖励和策略熵之和最大。

$$\pi^* = \arg \max_{\pi} J(\pi)$$

$$J(\pi) = \sum_t \mathbb{E}_{r(s_t, a_t) \sim \rho_{\pi}} [r(s_t, a_t) + \alpha \cdot \mathcal{H}(\pi(\cdot | s_t))]$$

其中  $\mathcal{H}(X) = -\sum_x p(x) \log p(x)$ 。

Entropy Loss 的实现做拓展性要求，不做强制性要求，后续给出其实现代码，同学们可以认真思考这里的实现细节。

## 实验任务二：王者荣耀 1V1 算法模型调优

### 1. 调整 Reward 权重

读懂 config.json 中每个子奖励的含义，调节权重观察不同权重策略下智能体的行动特点，并对探究结果进行总结分析。

config.json 中定义了 reward 的权重，同学们可以通过修改各个 reward 的权重去训练出玩法风格截然不同的 agent，比如提高击杀奖励，agent 会变得更好战，提高金钱奖励 agent 会提高刷钱效率。同学们可以通过消融实验的方法，去测试每个 reward 对 agent 训练的影响。但是不同 reward 之间不是相互独立的，同时修改多个 reward 的效果不能简单的用单个 reward 的实验结果进行简单的线性叠加。如何能达到一个相对平衡的点并取得游戏的胜利，是同学们可以思考的方向。

在本次实验中，只要求对不同 reward 之间的权重进行调整，不需要同学们

去修改 reward 具体的生成逻辑和计算方法，只需理解即可。

## 2.超参数调整

超参数的调整一直被认为是深度学习中的“玄学”，同样的算法，同样的参数，针对不同应用场景都可能会有较大的表现差异。

app/sgame\_1v1/common/configs/config.py 和 conf/configure.ini 中包含了很多参数的配置，同学们需要首先学习并了解每个参数的含义以及对训练/推演的影响，判断出哪些是可以去优化的，再利用实验去验证。

(1) 调节裁剪阈值 $\epsilon$

(2) 调节样本利用轮数

(3) 调整 GAE 的参数 $\lambda$

此外，也可以考虑调整三个子损失之间的权重。

## 3.调整网络结构

app/sgame\_1v1/common/models/model.py 中有对 Graph 的定义，具体参考 \_bulid\_infer\_graph 和 \_inference 函数，同学们可以在这里进行修改 graph 的操作；app/sgame\_1v1/actor\_learner/game\_controller.py 中有对 Graph 的引用。

神经网络的宽度与深度一定程度上决定了其模型容量，也对最终的性能有一定影响。整体上较大的宽度性能较好，而深度的关系不太明确。

\_inference 函数中含有具体的网络结构定义，可以尝试对其做适当修改验证自己的想法。

在这一实验中，可以考虑读懂\_infernce()函数，给出函数的逻辑、流程、实现的功能；给出自己对网络结构的修改。对比修改前后代码的结果：(1) 修改后 vs 修改前 (2) 修改后 vs baseline0 (3) 修改前 vs baseline0 的结果对比。

**代码修改与训练经验分享：**关注 algorithm.py 这一文件。

- 可以修改奖励权重和 loss 的权重。
- 对源代码中的 ppo 算法进行优化，可以去网上查找资料学习。
- 对源代码中的网络结构进行修改。
- 修改之后可以首先对比不同版本代码相同训练时间（短一点比如 10-20 小时）的表现，然后选择相同训练时间效果比较好的代码进行长时间的训练。
- 基于现有模型更改参数后继续训练。

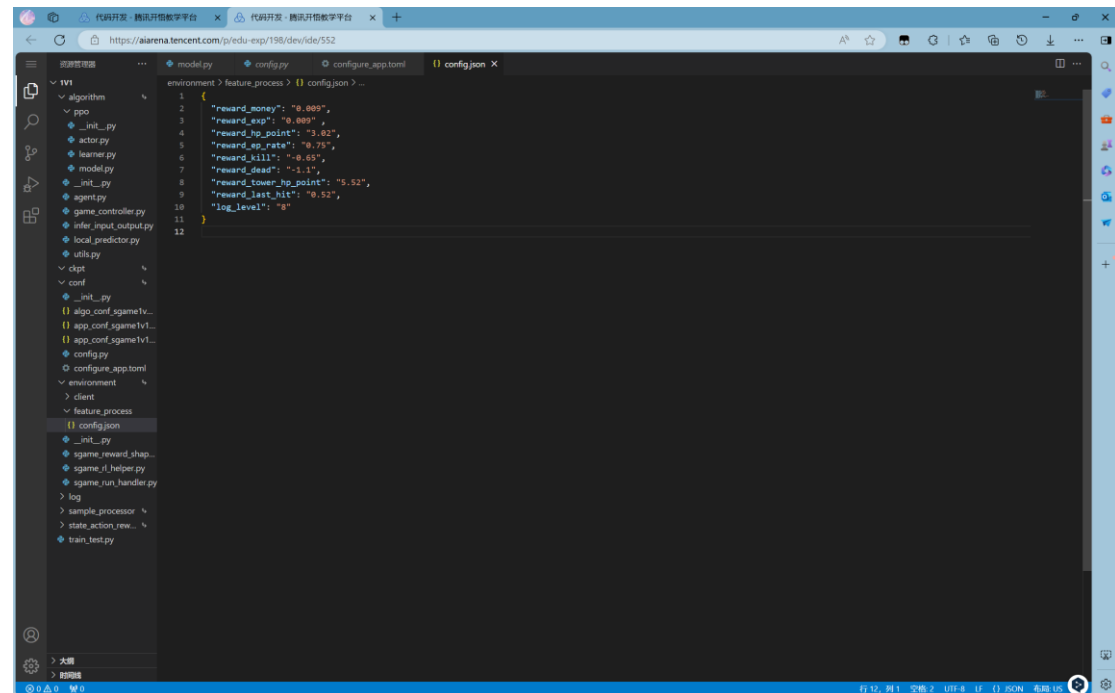
## 七、实验器材（设备、元器件）：

PC 微机一台

## 八、实验步骤：

负责神经网络结构的调整，rewards 的改变以及超参数调整的探索。

### 1. rewards 相关部分修改



“reward\_money”、“reward\_exp”、“reward\_last\_hit”如果调整过高，会导致人物更关注于小兵而忽略了敌方英雄的进攻与防御塔的攻击。

“reward\_hp\_point”过高和“reward\_dead”过低会导致英雄过分关注自己血量，会出现死亡一次就不出来的情况。

“reward\_kill”过高，自己人物会更倾向于攻击敌方，有时会忽略敌方小兵的共计，也会追着对面攻击忽略地方防御塔的共计。

### 2. 神经网络结构调整



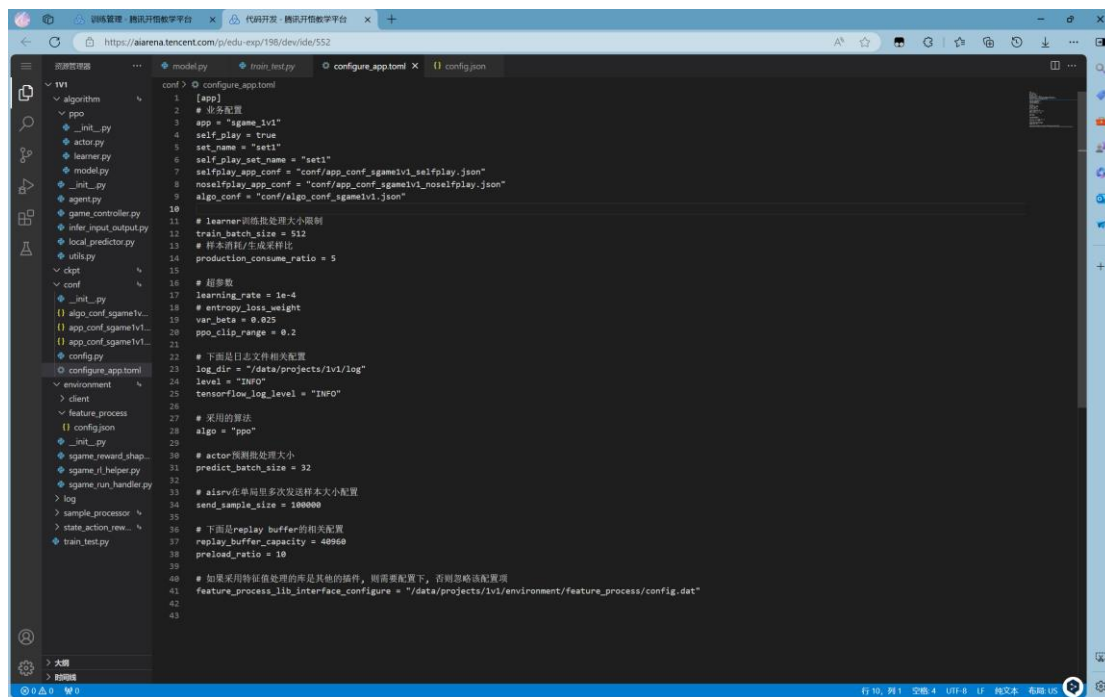
```

601     ## action layer ##
602     for index in range(0, len(self.label_size_list) - 1):
603         with tf.variable_scope("fc2_label_%d" % (index)):
604             fc2_label_weight = self._fc_weight_variable(shape=[self.lstm_unit_size, self.label_size_list[index]],
605                                                         name="fc2_label_%d_weight" % (index))
606             fc2_label_bias = self._bias_variable(shape=[self.label_size_list[index]],
607                                                  name="fc2_label_%d_bias" % (index))
608             fc2_label_result = tf.add(tf.matmul(reshape_lstm_outputs_result, fc2_label_weight), fc2_label_bias,
609                                     name="fc2_label_%d_result" % (index))
610             result_list.append(fc2_label_result)
611
612     with tf.variable_scope("fc2_label_%d" % (len(self.label_size_list) - 1)):
613         fc2_label_weight = self._fc_weight_variable(shape=[self.lstm_unit_size, self.target_embed_dim],
614                                                     name="fc2_label_%d_weight" % (len(self.label_size_list) - 1))
615         fc2_label_bias = self._bias_variable(shape=[self.target_embed_dim],
616                                             name="fc2_label_%d_bias" % (len(self.label_size_list) - 1))
617         fc2_label_result = tf.add(tf.matmul(reshape_lstm_outputs_result, fc2_label_weight), fc2_label_bias,
618                                 name="fc2_label_%d_result" % (len(self.label_size_list) - 1))
619
620     #for t in tar_embed_list:
621     #    print('t shape', t.shape)
622
623     tar_embedding = tf.stack(tar_embed_list, axis=1) ##(Batch_size, unit_num, embed)
624     ulti_tar_embedding = tf.layers.dense(tar_embedding, self.target_embed_dim,
625                                       use_bias=False) ##(Batch_size, unit_num, embed)
626     reshape_fc2_label_result = tf.reshape(fc2_label_result, shape=[-1, self.target_embed_dim, 1],
627                                         name="reshape_target_embed")
628     fc3_label_result = tf.matmul(ulti_tar_embedding, reshape_fc2_label_result) ##(Batch_size, unit_num, 1)
629     target_output_dim = int(np.prod(fc3_label_result.get_shape()[1:]))
630     reshape_fc3_label_result = tf.reshape(fc3_label_result, shape=[-1, target_output_dim], name="target_output")
631     result_list.append(reshape_fc3_label_result)
632
633     with tf.variable_scope("fc1_value"):
634         fc1_value_weight = self._fc_weight_variable(shape=[self.lstm_unit_size, 128], name="fc1_value_weight")
635         fc1_value_bias = self._bias_variable(shape=[128], name="fc1_value_bias")
636         fc1_value_result = tf.nn.relu(tf.matmul(reshape_lstm_outputs_result, fc1_value_weight) + fc1_value_bias,
637                                         name="fc1_value_result")
638
639     with tf.variable_scope("fc2_value"):
640         fc2_value_weight = self._fc_weight_variable(shape=[128, 1], name="fc2_value_weight")
641         fc2_value_bias = self._bias_variable(shape=[1], name="fc2_value_bias")
642         fc2_value_result = tf.add(tf.matmul(fc1_value_result, fc2_value_weight), fc2_value_bias,
643                                 name="fc2_value_result")
644     result_list.append(fc2_value_result)

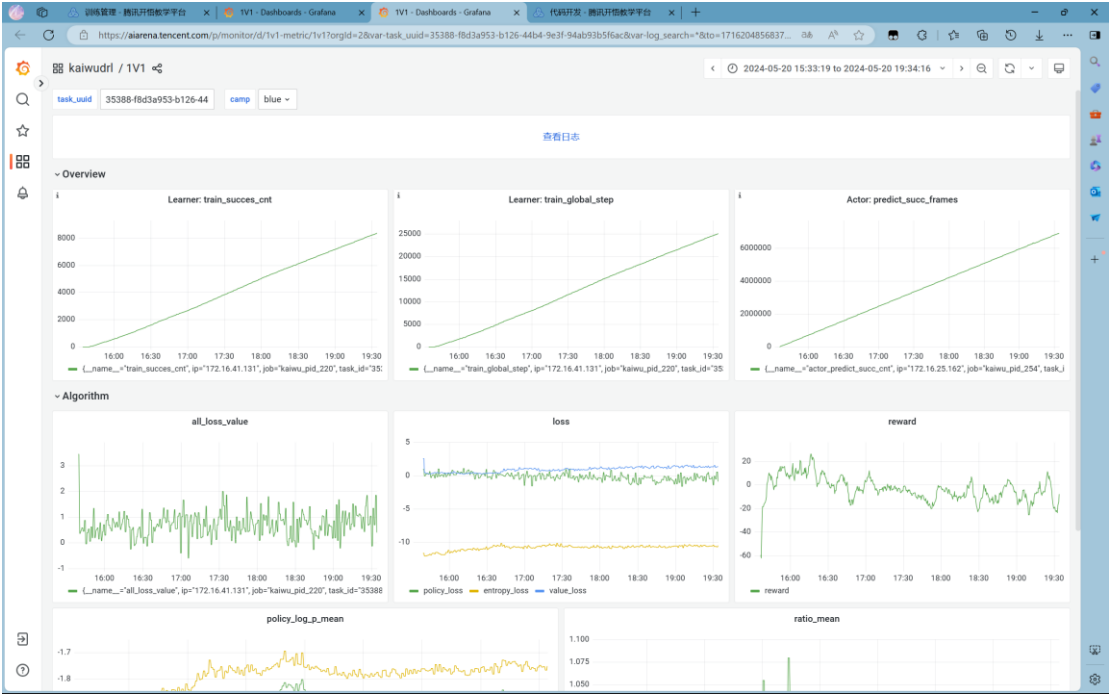
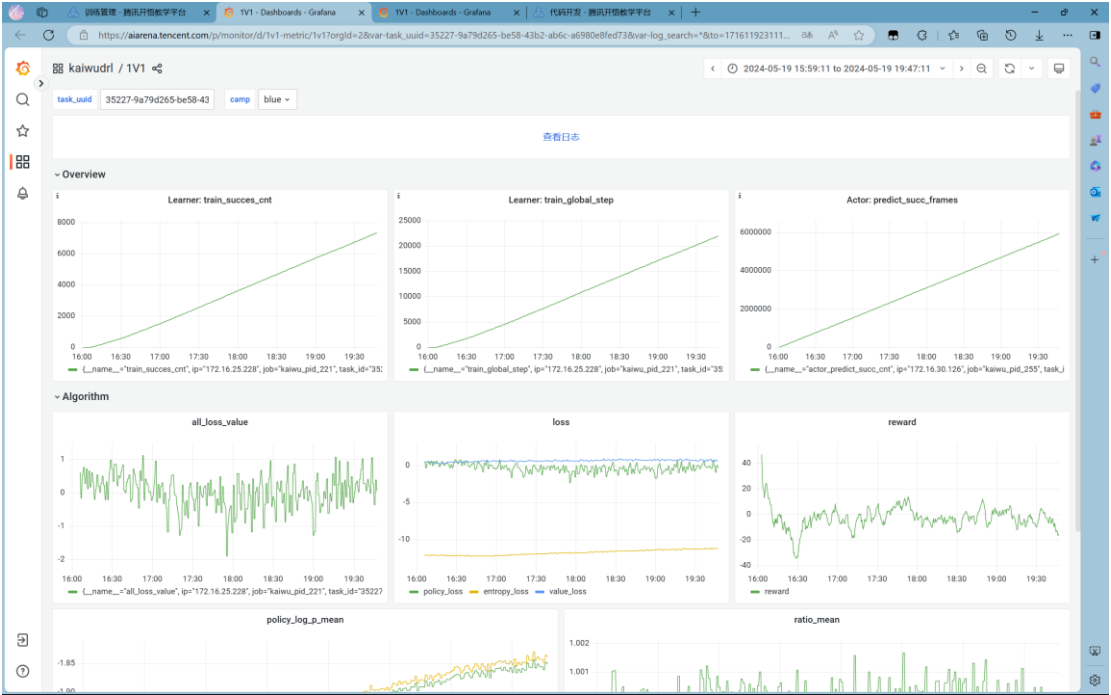
```

包括在己方英雄、敌方防御塔、小兵以及动作等部分增加神经网络的层数以及增加节点数量。但会导致训练过程中训练时间过长。

### 3.超参数的调整

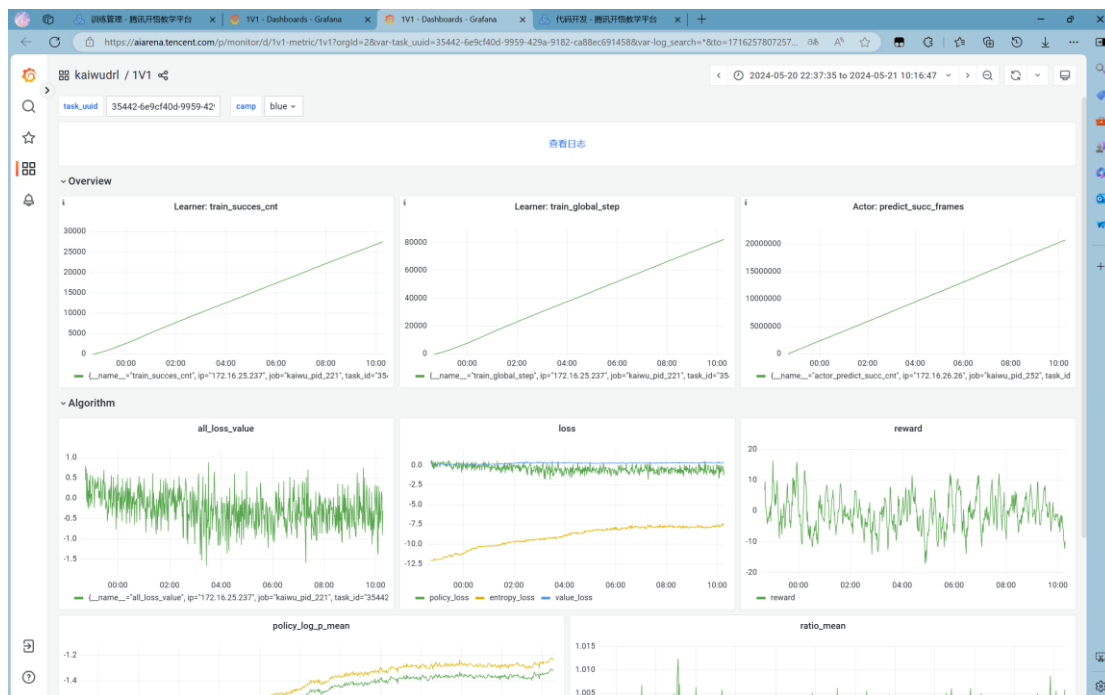
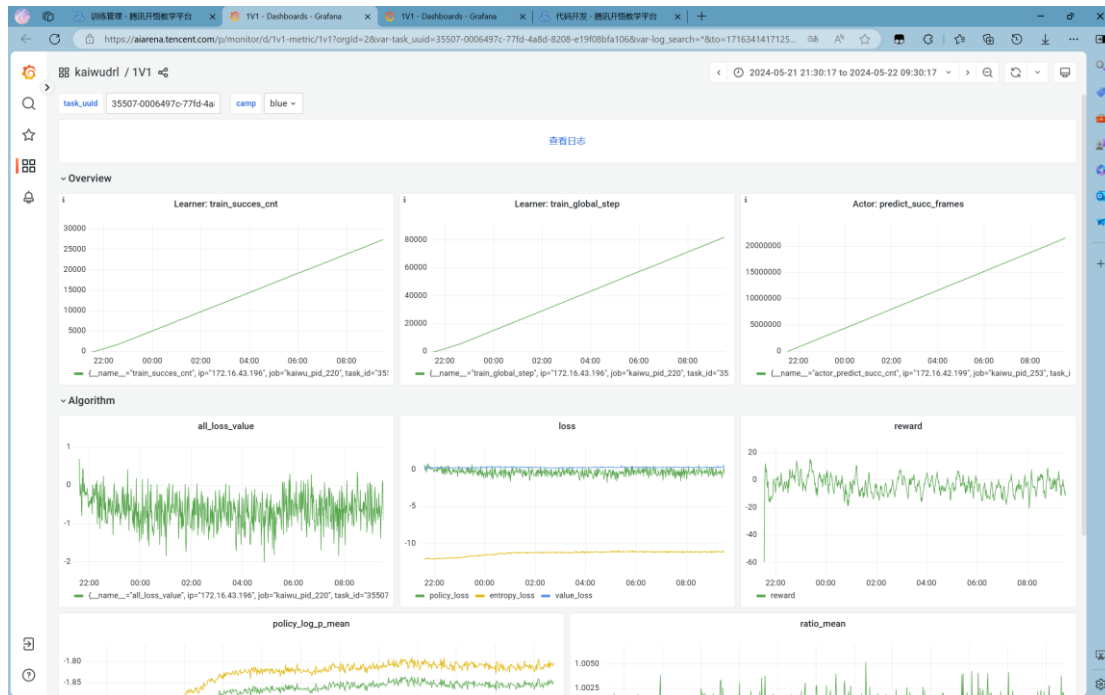


增大了学习率，使得收敛更快。



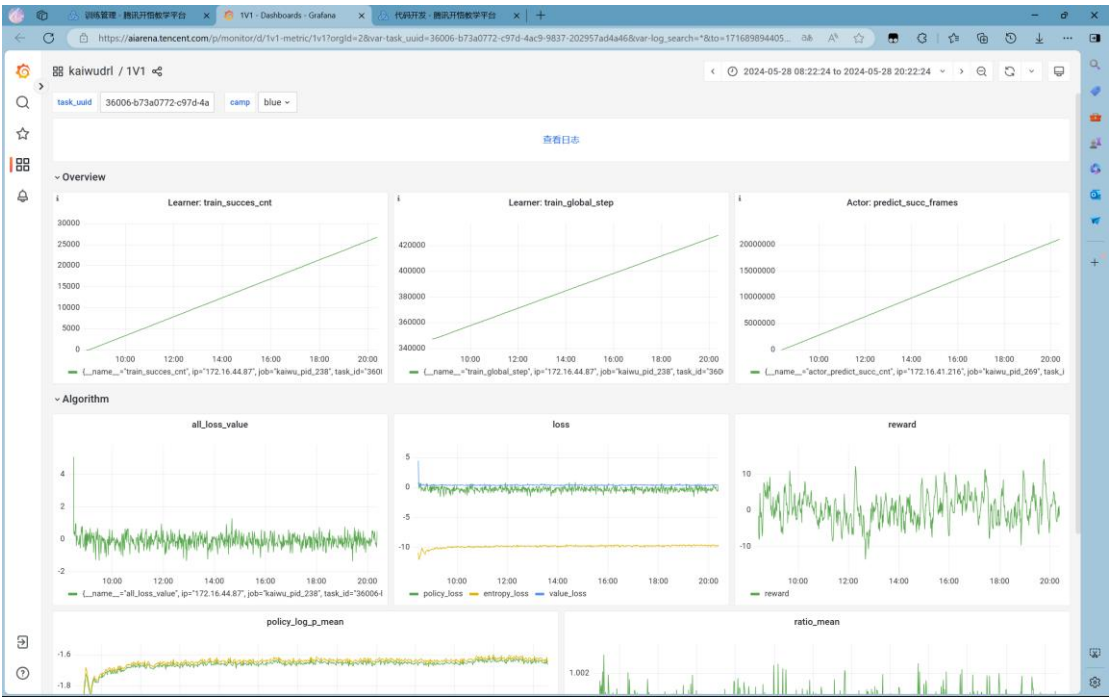
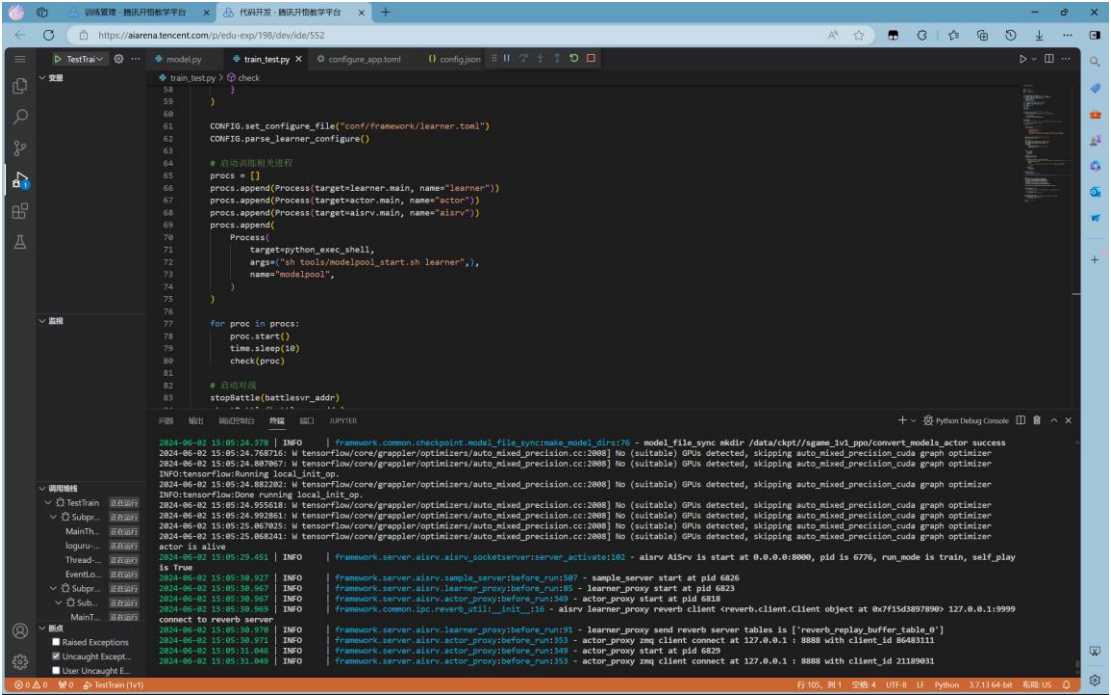


尝试增加 `var_beta` 即 `entropy_loss` 的权重，但是会使得 `value_loss` 和 `policy_loss` 对 `loss` 的影响减少。模型随机性变强，收敛变得困难。



九、实验数据及结果分析：

1. 代码运行以及训练






2.测评结果

腾讯开悟

English 使用手册

数学测评 > 测评详情



### 第一次测评

[关联实验]王者荣耀1v1

测评状态: ● 已结束    测评方式: 挑战测评    测评对象: 战队    模型提交: 已截止

测评描述: 使用学生提交的模型与课程管理员设置的挑战模型进行对抗, 通过胜率评估模型的能力。

测评细节

挑战模型: baseline-level-0

地图: 皇家机关道

阵容: 鲁班七号

框架: TensorFlow

算法: PPO

测评轮数: 3

模型列表

测评任务


测评成绩

阵容A	阵容B (挑战模型)	A胜 / B胜 (总局数)	胜率
腾讯不知凡几 522model2_12h	baseline-level-0	6 / 0 (6)	1

腾讯开悟

English 使用手册

数学测评 > 测评详情



### 第二次测评

[关联实验]王者荣耀1v1

测评状态: ● 已结束    测评方式: 挑战测评    测评对象: 战队    模型提交: 已截止

测评描述: 使用学生提交的模型与课程管理员设置的挑战模型进行对抗, 通过胜率评估模型的能力。

测评细节

挑战模型: baseline-level-1

地图: 皇家机关道

阵容: 鲁班七号

框架: TensorFlow

算法: PPO

测评轮数: 3

模型列表

测评任务


测评成绩

阵容A	阵容B (挑战模型)	A胜 / B胜 (总局数)	胜率
腾讯不知凡几 527model1	baseline-level-1	6 / 0 (6)	1

腾讯开源

English使用手册

数学测评 > 测评详情



### 第一次天梯测评

[关联实验]王者荣耀1v1

测评状态: 已结束 测评方式: 天梯测评 测评对象: 战队 模型提交: 已截止

测评描述: 使用学生提交的模型相互进行对抗, 通过胜率评估模型的能力。

测评细节

地图: 皇家机关道 阵容: 鲁班七号 框架: TensorFlow 算法: PPO 测评轮数: 3


模型列表 测评任务 测评成绩

排名	战队名称	成员	胜局 (总局数)
1	武汉QG超玩会	陈鑫、高嘉炫、熊智铭	53 (60)
2	一位难求全球赛水平下降一万倍	姜阳宇、秦宇韬、马小强	46 (60)
3	从来没有觉得训练模型开心过	徐晨辉、汪一萍、黎少峰	37 (60)
4	转生成为AI然后打上国服鲁班	尹成峰、张毅晋、江昊捷	35 (60)
5	感觉不如原神	黄思宇、朱若愚、高畅	31 (60)
6	知道你操作弱好进来学习AI的思路	韩明赫、王彦哲、谢耀云	30 (60)
6	开了吗 我说灵智	高家满、赵振豪、符骏成	30 (60)
8	我的AI比我强	谭泽天、廖子杰、邓兴	25 (60)
9	你的操作很起模但你的AI却弥补了这一点	何越天、熊孝然、陈佳文	24 (60)
10	王者荣耀	刘子昂、丁子昂、刘子昂	13 (60)

腾讯开源

English使用手册

数学测评 > 测评详情



### 第三次测评

[关联实验]王者荣耀1v1

测评状态: 已结束 测评方式: 挑战测评 测评对象: 战队 模型提交: 已截止

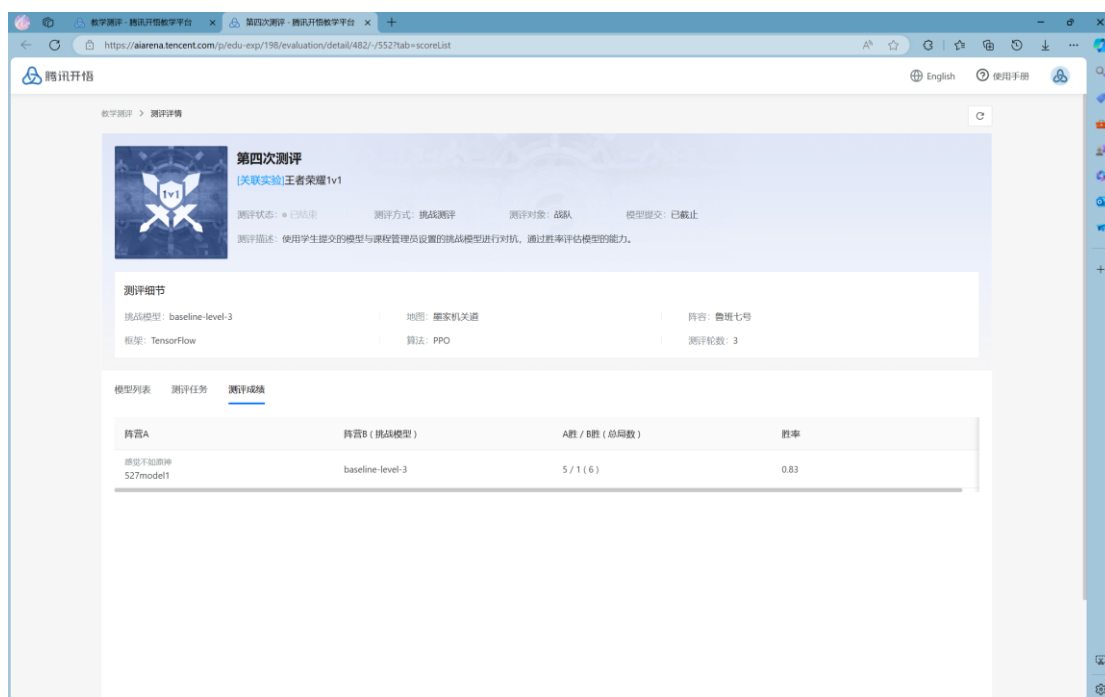
测评描述: 使用学生提交的模型与课程管理员设置的挑战模型进行对抗, 通过胜率评估模型的能力。

测评细节

挑战模型: baseline-level-2 地图: 皇家机关道 阵容: 鲁班七号 框架: TensorFlow 算法: PPO 测评轮数: 3

模型列表 测评任务 测评成绩

阵容A	阵容B (挑战模型)	A胜 / B胜 (总局数)	胜率
感觉不如原神 527model1	baseline-level-2	5 / 1 (6)	0.83



## 十、实验结论:

经过强化学习训练所得模型可以完成对战任务, 正确移动自己的位置并根据情况选择恰当敌方目标选择合理方式进行进攻, 完成自己经验积累, 金钱获取等并且共计敌方防御塔获得胜利。

## 十一、总结及心得体会:

PPO 算法具有稳定性强、计算效率高和实用性广泛等特点, 可以通过限制策略更新步长, PPO 算法能有效避免训练过程中的大幅波动, 保证学习过程的稳定性, 相比于 TRPO, PPO 简化了计算过程, 降低了算法的计算复杂度和资源需求, 在多种任务和复杂环境中表现优异, 具有广泛的适用性。

## 十二、对本实验过程及方法、手段的改进建议:

尝试使用 PPO-Clip 或者 PPO-Penalty 对现有算法进行改进, 强化提高稳定性和收敛性, 强化在连续动作空间中的任务和进一步推动代理学习到更多的探索行为。

报告评分:

指导教师签字: