# 开悟比赛-五杀蔡文姬队技术整理与分享

陈华玉 清华计算机系 黄彬 清华计算机系

沈晓腾 清华大学自动化系 严谕梓 清华大学电子工程系

指导老师: 朱军，阎栋

## 一、 简介

在 2022 年 9 月-2023 年 4 月举办的腾讯第三届开悟 MOBA 多智能体强化学习大赛中，我们队伍(五杀蔡文姬)在初赛中有幸获得了第五名的成绩。这得益于实验室老师与腾讯官方的大力支持， 也得益于我们队伍本身的技术探索与积累。本文首先会简单叙述比赛的基本情况，随后从各关键模块出发，简要介绍本队伍在开悟比赛中的探索历程与心得体会。

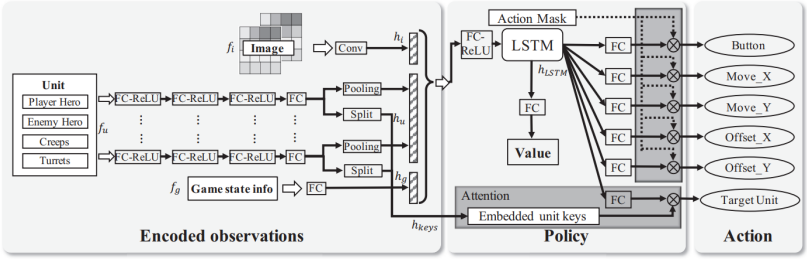
## 二、 参赛概况

本次比赛共有四名同学参赛。主要参赛目的是通过开悟平台学习多智能体强化学习技术，积累大规模强化学习训练相关的项目经验。得益于之前一些比赛经历，团队时间投入大约在 10 小时/人/星期。

在为期 7 个星期的初赛中，每支队伍被分配了 256 核 CPU 以及一块 GPU，每周每只队伍有五天的计算资源使用量。后续将介绍队伍的一些模型，算法，工程方面的设计选择。需要注意的是，由于队伍的计算资源有限，因此在比赛全程中机会没有任何机会对单一的更改进行对照试验，因此下文中所有技术的探索以及好坏的比较，都掺杂了极大主观性因素且在有强时间限制下得出的结论。需要辩证看待。

## 三、网络设计

神经网络结构部分，我们团队大范围借鉴或者使用了王者绝悟公开论文[1]中提及的神经网络架构（见下图）。简单来说，就是通过共享的网络参数，单独处理游戏中同一类型的数据，最后通过 maxpool 操作将高维数据降维度，并产生相应的注意力头。假设全图有 20 个野怪，每个野怪经过神经网络编码后得到了一个 32 维度的向量数据，相应的处理方法是将 32*20 维度的向量数据经过 maxpooling 降维成为 32*1 维度的数据。




基于此模型结构，考虑我们针对本次比赛的赛题特点做了如下针对性改进。

1. 由于注意到初赛需要用极少的训练资源在有限的时间内联合训练五个英雄，且存在多种排列组合，计算资源严重不足。我们通过将 LSTM 替换为 GRU 网络，并将 hidden size 降低为原论文的 25%，用 MLP 层补缺替代，节省了接近 50%的参数量，从而可以在后期大幅度提高训练收敛速度。
2. 结合参数量大小限制，我们放弃掉了每个英雄单独一个神经网络的考虑，改用所有英雄共用一个神经网络。但是为了保证模型表征能力，我们又将与英雄相关的 encoder 部分整体加宽，最后宽度是 1024->512->256。
3. 对于己方和敌方的 obs 信息，我们在神经网络浅层共享参数，在较深层才使用不同的参数，从而进一步降低参数量。
4. 我们加深了 value network，从原论文中的两层增加为四层（512->128->64->1）。

## 四、奖励体系

在本次比赛中，我们对于奖励函数进行了一定的微调。微调奖励函数的总体原则是，肉眼分析比赛对局，观察英雄在哪方面的能力较弱，比如我们在比赛中发现英雄不重视补兵清线，则会增加 money 权重项。最终我们确定的 reward 权重列举如下。

| Parameter | Value |
| --- | --- |
| reward_money | 0.008 |
| reward_exp | 0.006 |
| reward_hp_point | 4.0 |
| reward_ep_rate | 0.75 |
| reward_kill | -0.6 |
| reward_dead | -1.0 |
| reward_tower_hp_point | 20.0 |
| reward_last_hit | 0.5 |
| log_level | 8 |

我们发现在训练后期逐步增加 gamma 权重会得到较好的效果，因为我们在训练最后两个星期讲 gamma 提高至 0.997。我们如官方推荐使用了零和博弈机制。注意到智能体在比赛后期获得相同奖励的时间要比前期少很多，我们也将所有奖励统一 乘以 0.6 的指数衰减因子，但对此我们并未进行任何详尽的对比试验，无法保证超参最优。

## 五、特征与规则

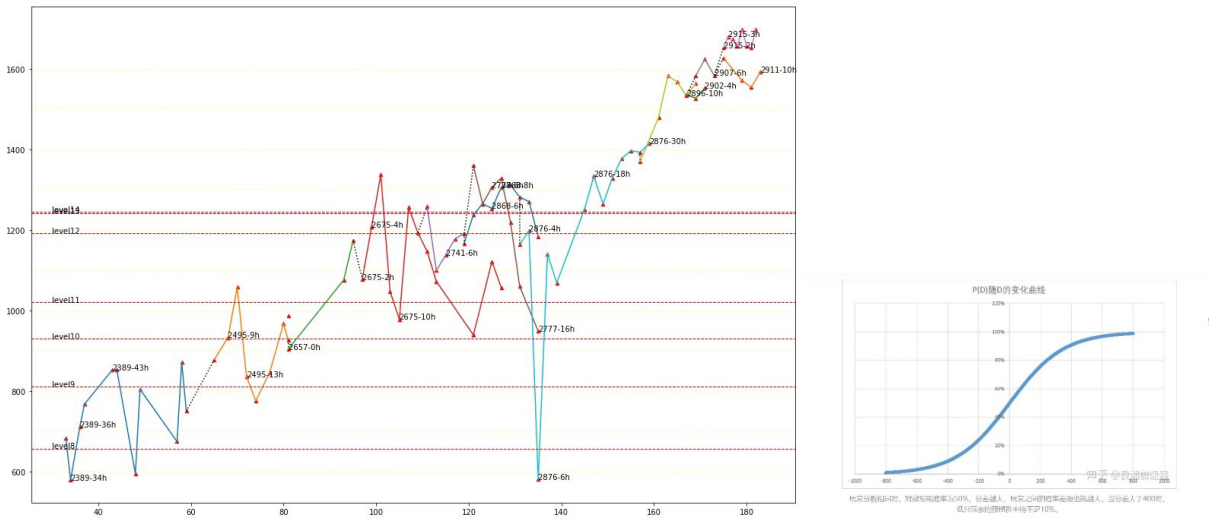
本次比赛中我们队伍没有进行任何特征工程和规则设计。但是在执行强化学习训练时，我们通过人为给选择 “空” 动作增加 loss 乘法，从而软性引导智能体少选择这个动作。

## 六、 强化学习算法

对于强化学习算法，我们直接使用了官方提供的 PPO 算法。对于训练超参数上，我们分批多次在训练过程中逐步降低学习率（至 5e-6）和熵损失权重（至 1e-4）。发现这两个手段有助于稳定训练，损失梯度估计方差。对于估计优势函数的 GAE 算法，我们全程设置 lambda 为 0.95。

## 七、 系统工程架构

在本次比赛中，队员们需要手工将对局产生的英雄模型上传至托管平台，且需要手工选择模型进行对战，人工判断模型的好坏与克制关系，并利于这些知识开展下一阶段的实验。这其间需要消耗大量的人力资源，降低了实验的周转效率。对此我们实现了多风格智能体水平评估与托管对战的自动化部署，较大程度上节省了人力，增加了实验周转效率（此处多风格仅仅代指在不同如奖励权重和学习率超参数下训练的多个智能体断点，并非 league-style 训练中的并行训练智能体）。具体来说，我们可以全自动化地将正在进行的实验模型定期拉到一台实验室服务器上，随后定时自动编号并上传托管平台。相关爬虫程序可以自动从托管平台选择模型进行对战并把对战数据保存至本地。在本地我们也部署了相关算法可以实时计算每个模型的 ELO 分数，并将分数排序反馈给爬虫程序作为其选择对战模型的参考（选择托管对战模型的大致逻辑是只保留在一定阈值以上 ELO 分数的模型进行对战，且在这个较好 ELO 分数的模型中按照分数越高选择概率越大的基本逻辑进行随机采样）。除了用于托管模型选择外，相关 ELO 分数也可以利用既有脚本文件绘制模型迭代示意图方便人对于训练状况分析。




模型迭代示意图示意图，横轴为模型从一开始初始化之后训练的总时间，纵轴为模型 ELO，不同颜色的线段表示一次单次训练，虚线表示模型继续训练或模型版本回退，红点为算法选择出的有代表性的断点模型与相应训练时间，模型半小时保存一次，训练数据与模型全部备份存储在本地服务器中。

## 八、 模型迭代过程

模型全程按照 PPO 强化学习方法训练，无除超参调节外的迭代过程。

## 九、 训练效果分析

总体而言，针对初赛的赛题特点，我们认为在比赛中下列几个关键点是本队伍收益最高，认为相对重要的。

1. 由于比赛计算资源不足但赛题复杂，因此尽可能地精简模型参数，保证模型能以最快的速

度收敛且不至于过拟合到少量数据上。(没有精确计算过队伍模型参数量，但粗略估计应该不超过 3M)。

2. 由于训练时长相对短，因此花费大量时间探索最优超参或者模型结构，可能受益远不如迅速找到可以使模型稳定提升性能的非最优参数，用相应资源增加训练时间。根据不精确回忆，在七个星期的比赛进程中，我们队伍实际用于探索策略的计算资源占三个星期（主要集中在比赛前期)，其余四个星期都在针对单一模型不断进行训练。在训练前期，在官方模型学习率的参数下，模型可以维持大约 2 个星期的稳定提升，随后模型进入波动提升期，在此过程中需根据积累的调参经验对于 PPO 参数学习率等少数几个超参数以及奖励权重进行调节。能力波动提升会一直持续到比赛结束，且远远达不到收敛预期。
3. 训练后期出现训练不稳定时对于学习率降低和熵损失权重进行逐步探索调整是有效且必要的。
4. 没有采用过于复杂的特征工程或者训练迭代流程，保持代码的干净稳定从而最大限度地减少出 bug 的概率。
5. 初赛中的一个重要设置是，队伍可以自行选择英雄携带的主动技能。然而绝大多数队伍忽略了这个自由度，在训练过程中全程让英雄使用了默认主动技能。我们队伍在训练中给训练对局配置了随机主动技能，这个设置对于队伍在基准挑战赛中取得相对好的成绩至关重要，因为基准挑战赛中基准英雄所携带的主动技能并非一定是默认主动技能。(无论在基准赛挑战赛中还是在队伍对战中，出于方便考虑我们的英雄都还是装备了官方默认推荐主动技能，我们的英雄具备使用其他任意主动技能的潜在能力，由于英雄在训练过程中对手的主动技能随机，因此我们的智能体在对战官方基准线时有更好的泛化性优势)。

## 十、 总结与展望

由于时间原因，原定计划探索尝试的多头奖励训练机制，更加细致的奖励函数调参等等技术点没有时间尝试，在后续比赛中可以寻找机会补齐。

## 参考文献

[1] Mastering complex control in moba games with deep reinforcement learning

[2] Towards Playing Full MOBA Games with Deep Reinforcement Learning