基于强化学习的黑白棋人机博弈

摘要

随着人工智能的广泛应用，游戏行业也受到了极大关注。本文基于强化学习，构建了一个黑白棋的博弈机器人。

首先，分析了棋类游戏经典的算法，以及它们各自的优缺点，借鉴 AlphaGo Zero 的思想，我们没有使用外部数据集，采用自对局方式，完成模型训练，并且为了保证数据的多样性，对棋面数据进行了一定程度扩充；接着，为了能够训练这个智能体，采用与蒙特卡洛树搜索模拟的方式，当 AI 胜率达到 0.9 以上时，不断提升模拟次数，从而加强 AI 的棋力；最后构建了一个价值网络，价值网络有两个输出，一个是棋面估值函数，一个是每个位置落子概率，为了让学习率能够随着迭代而自动变化，引入了 KL 散度，实验结果表明引入 KL 散度的模型收敛更加平稳并且迅速。

在实验部分，对模型进行了测试，采用常见的第三方平台，测试结果表明，训练得到的模型能够较好学习到黑白棋的下棋技巧，总体胜率表现不错。同时，针对蒙特卡洛模拟次数这一超参数，进行了对比测试，提升这一超参数后，模型的棋力有所增强，能够打败原来的 AI，并且让对手跳过回合的概率更大。但是训练花费的时间更长，模型收敛速度更加缓慢。最后对比了扩展结点这一关键超参数，扩展结点越多，收敛速度越慢，但是模型的精度越好。

在工程实现上，将棋面抽象为二维矩阵，在对矩阵进行编码，便于网络传输；同时利用 HTML/CSS/JS 进行前端搭建，前后端采用 HTTP 通信，构建了一个交互界面。

关键字： 强化学习 蒙特卡洛树搜索 黑白棋 KL 散度

目录

1. 背景.....3
2. 相关工作.....3
3. 模型构建.....4
    3.1 游戏规则.....4
    3.2 自我对局.....4
    3.3 蒙特卡洛搜索.....5
    3.4 价值网络.....6
4. 实验结果.....9
    4.1 训练模型.....9
    4.2 测试.....10
    4.3 超参数对比.....12
5. 工程实现.....13
6. 总结.....14
参考文献.....15

1. 背景

游戏领域是人工智能研究的经典问题之一，对于人类来说，是展现智慧的一种方式。而游戏的设计中，其实已经蕴含了人工智能的种子，游戏中的 NPC 玩家（Non Player Character, NPC）本身就是一个智能体，需要不断和真实玩家进行交互，让真实玩家获得更好地体验。

如果将机器学习简单分为非监督学习、监督学习和强化学习，那么大部分的游戏智能体需要用到强化学习。游戏是动态发展的，智能体处在这样一个实时变化的环境中，需要不断试错，才能赢得游戏；通过数据不断驱动，让智能体自主学习游戏规则，通过不断改变策略，获得更多的奖励，最后赢得游戏。

2. 相关工作

在一些复杂的游戏上，例如象棋、围棋等游戏，计算机无可匹敌的计算优势。经典算法例如 Minimax[1]，轮到己方决策时，最大化收益，此时这一层为称为 MAX 层；当需要对手决策时，对手需要最小化我方收益，因此对应层称为 MIN 层。双方交替决策，此时的树形结构是极大极小搜索树。随着搜索深度的增加，搜索空间呈现指数级增加，因此有人提出使用$\alpha-\beta$剪枝算法对游戏空间进行搜索[2]。然而，基于$\alpha-\beta$剪枝的棋类程序难以处理决策空间较大的游戏，因为合法结点过多，即使剪枝，也需要大量搜索，计算量依旧不；同时剪纸可能提前剪断了后期价值较大的走法。在这种情况下，也可以考虑采样的方式，蒙特卡洛搜索树[3]在这种情况下也有不错的表现。

棋牌类游戏最重要的是棋面估值函数和决策函数[4]，大多数黑白棋的估值函数是静态估值[5]-[6]，即提前为每个位置分配一个权重，例如四个角落的权重最大，因为它们不可以被翻转，属于不动点。这类方式的弊端是没有考虑棋面的动态变化；随着机器学习的火热，价值神经网络在计算机围棋领域取得重大突破，AlphaGo Zero[7]战胜了人类的顶尖棋手，这说明价值神经网络具备很大的潜能。

在围棋[8]-[9]、五子棋[10]和象棋游戏中有许多使用价值神经网络和蒙特卡洛搜索的实现实例，但黑白棋相对较少，其原因有三点。第一，围棋领域中 AlphaGo Zero 的复现相对困难，即使做了模拟优化，其计算量相当大；第二，比起围棋，五子棋和象棋的搜索空间要小很多，在一定程度上减少了计算量，同时五子棋的规则相对简单，而象棋的落子点相对可控，不可以随意落子，这些因素使得两种游戏的实现难度降低；最后，黑白棋并不是中国的传统棋类，在国内相对不流行，因此相对资料会少一些。

黑白棋这类棋牌游戏属于零和博弈，并且能够在有限局结束，这类游戏有必胜策略[11]，但由于搜索空间非常大，这个“上帝视角”只能存在于数学理论中，通过蒙特卡洛模拟，选择部分游戏分支进行探索，这是一个可行方案。

3. 模型构建

3.1 游戏规则

黑白棋也称翻转棋（Reversi）或者奥赛罗（Othello），整个棋盘大小为 8×8，一共 64 个格子。游戏规则[12]如下：

（1） 黑方先手，黑白双方轮流选择一个合法位置落子。

（2） 一个合法位置是一个空白位置，并且与自己棋子构成的连线上至少夹着对方的一颗棋子，这条连线可以是横、竖或者斜对角线三种方向。

（3） 一步棋可以在多个方向翻转对方棋子，并且满足要求的棋子必须被翻转。

（4） 一方有合法落子的位置，必须选择落子；若没有合法落子的位置，则由其对手继续落子，直至有合法棋子可下。

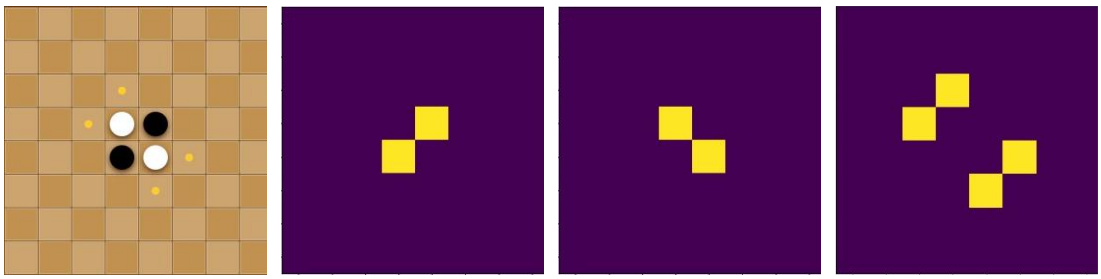
（5） 当棋盘没有多余位置或者双方都无法落子时，游戏结束，棋子数量多者获胜。

3.2 自我对局

模型训练需要数据集，一方面网络上对于黑白棋很少有大量共训练的数据集；另一方面，自我对局（self-play）可以避免人类经验固有的缺陷。

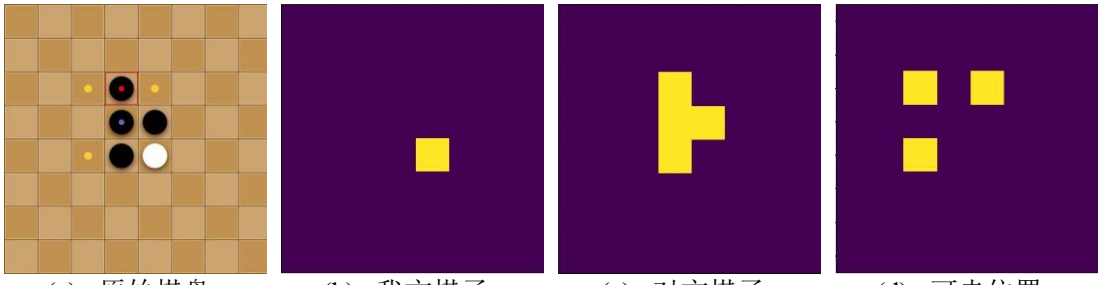
自我对局需要对棋盘做一定的抽象处理，将棋盘视为一个8×8的矩阵，为了消除数据本身差异，采用二值矩阵描述棋局，即 1 表示有，0 表示无。

因此，对于一个基本棋面，需要将其切割为四个平行棋面，分别表示我方棋子、对方棋子、可走位置和角色（黑子还是白子）。示意图如图 1 和图 2 所示。



(a). 原始棋盘　(b). 我方棋子　(c). 对方棋子　(d). 可走位置

图 1 棋面抽象示例图 1



(a). 原始棋盘　(b). 我方棋子　(c). 对方棋子　(d). 可走位置

图 2 棋面抽象示例图 2

从示意图中可以看到，每一轮都是站在自己的视角看待棋盘，例如示例 1，这是黑棋的视角，当黑棋落子后，棋局来到示例 2，此时为白子的视角。

为了保证棋面数据的多样性，还需要对棋面进行数据扩充，从基本语义看，棋面的具有对称和旋转的特性，即将棋面以 90° 为间隔，共旋转 3 次，棋面的语义不变；同理，将棋面做镜像对称处理，语义同样不变，因此可以对一个基本棋面做旋转和对称处理，进一步增加数据集，保证数据集的多样性。

3.3 蒙特卡洛搜索

利用树搜索，可以从棋盘一开始状态，不断搜索可能的状态空间，示意图如图 3 所示。试图选择对自己最有利的走子方法，如果计算能力和存储空间无穷大，可以遍历全部的路径，对于对手的每一步，都可以有最好的应对策略。黑白棋的标准棋盘大小为 8×8，则全排列有64!种，当然这里有大部分的走子是不符合规则的，据估计合法位置大约有10^{28}，而博弈树的复杂度约为10^{58}，这是一个巨大的空间[13]。

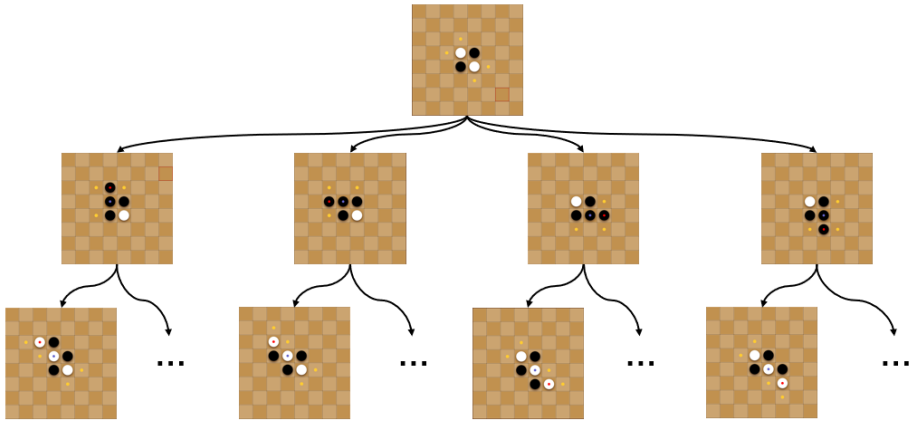


图 3 搜索树示意图

整棵树的深度非常有限，只有 60 层，因为初始化的棋局已占据 4 颗棋子，此后双方交替走子，每走一步，一定会减少一个空格，因此搜索树深度最大为 60。但是每层的结点数可能相当多，需要对部分结点进行深入搜索，其他结点忽略，以减少计算量。而蒙特卡洛搜索树（Monte Carlo Search Tree, MCST）是经典的棋盘游戏模拟方法。

蒙特卡洛搜索树对棋局的迭代分为 4 个步骤，包括选择（selection）、扩展（expansion）、仿真（simulation）和反向传播（backpropagation）。示意图如图 4 所示。

首先从根结点开始，选择一个子结点（选择）；之后在这个子结点基础上，继续扩展一个或多个子结点作为后续要搜索的结点（扩展）；从这一结点开始，不断向下迭代，每次随机选择落子点，直到游戏结束（仿真）；最后，根据游戏结果，计算结点胜率，并从最后一个叶结点，向前更新参数（反向传播）。

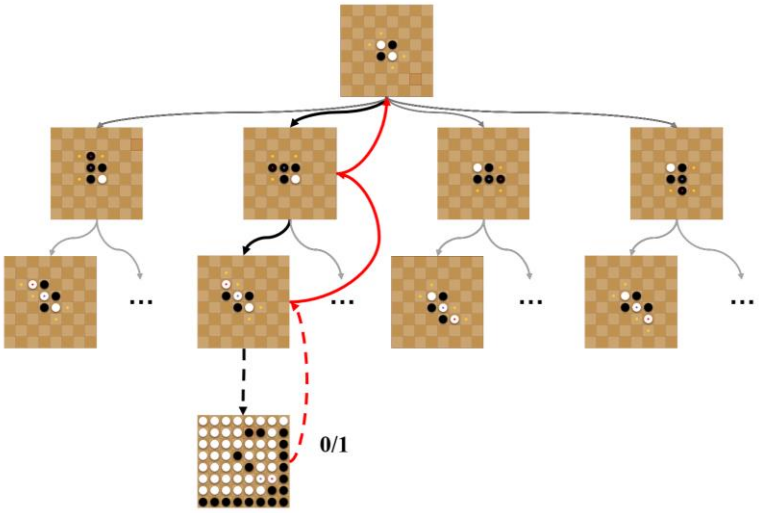


图 4 蒙特卡洛搜索树示意图

如何选择子结点成为关键，这里考量的因素主要有两个，一方面是结点的胜率，若结点的胜率Q越大，说明这一结点的子结点应该多考虑，因为获胜概率更大；另一方面是结点的访问概率，若一个结点被访问的次数越少，那么应该给予其更大的访问概率，因为其中可能暗藏了更好的走子方法。

结合上述考虑，可以确定一个选择的依据指标UCT，如式（1）所示。

$$UCT(v,v_i)=Q(v_i)+cP(v,v_i)\frac{\sqrt{N(v)}}{1+N(v_i)} \tag{1}$$

这里$UCT(v,v_i)$是结点v转移到结点v_i的价值评估函数；v是父结点，v_i是候选的子结点；c是超参数，作为平衡“好”的结点和没有被探索结点的一个权衡；其中$Q(v_i)$对应结点胜率，$\sqrt{N(v)}/(1+N(v_i))$是结点被访问的相对概率。

在反向传播阶段，递归更新结点参数，如式（2）所示。

$$Q(v_i)\leftarrow\frac{n}{n+1}Q(v_i)+\frac{-v_s}{n+1} \tag{2}$$

这里n是结点访问次数，v_s是叶结点胜负情况，取值为 1（胜利）、-1（失败）和 0（平局）。之所以需要取负号，因为这里是自我对局，每一轮都从当前玩家视角看待棋局，因此叶结点和父结点的胜负一定相反。

3.4 价值网络

式（1）中关键的参数$P(v,v_i)$是移动概率，需要有价值网络进行估算，这里本质是教会神经网络学会下棋。给定一个棋面S，神经网络需要输出每个点落子的概率，同时也要输出这个局面的评估得分。

为了避免过拟合，需要引入一定的随机噪声，如式（3）所示。

$$P(s,a)=(1-\varepsilon)P_a+\varepsilon\eta_a \tag{3}$$

其中，η_a是迪利克雷噪声（Dirichlet noise），ε是超参数，实际中取值不宜过大，这里选取 0.3，表示噪声占比。

输入参数是当前局面，通过标准化后，输入数据的维度是$4\times8\times8$，8×8是棋盘大小，而四个棋盘的堆叠，前三个分别是我方棋子、对方棋子和可落子的位置，最后一个维度表示当前棋局是黑子还是白子。

一组卷积核能提取一组特征，不妨设特征图片大小为(M,N)，卷积核大小为(m,n)，步长为(u,v)，在不考虑填充时，输出的特征图尺寸为(k,l)，则(k,l)应满足式（4）：

$$\begin{cases} k \le \dfrac{M-m}{u}+1 \\ l \le \dfrac{N-n}{v}+1 \end{cases} \tag{4}$$

考虑实际(k,l)只能取整数，因此对应数值需要向下取整，这里棋盘大小为8×8，棋盘面积较小，每一个格子对判断都很重要。这不同于一般的图片分类，一般图片大小可能在512×512数量级，故使舍去边缘信息，对最后判断的结果影响不大。因此对棋面进行卷积时，需要**填充周边**，保留边缘的信息，此时输出的特征图尺寸为：

$$\begin{cases} k = \left\lceil \dfrac{M}{u} \right\rceil \\ l = \left\lceil \dfrac{N}{v} \right\rceil \end{cases} \tag{5}$$

此时输出尺寸与卷积核大小无关。示意图如图 5 所示。此时蓝色部分是原始图像，通过在最外层填充，使得卷积后的图像也就是绿色部分，大小保持不变。

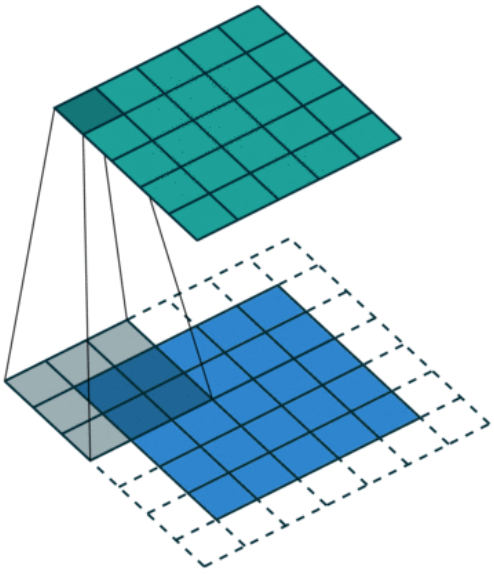


图 5 保持特征图大小卷积示意图

实际参数选择时，卷积核选择了常见尺寸3×3，两个方向的步长设置为 1，因此每次输出的特征图尺寸不变，保留了全部的棋面信息。

这里 Action FC 会输出每一个点落子概率，Evaluation FC2 会输出棋面s是估值得分v。训练的目标是让价值策略网络的输出概率p更加贴近蒙特卡洛搜索树

模拟得到的概率π；让局面评分能够更加准确的反映实际对局的结果z。

网络参数如表 1 所示。

表 1 网络参数

网络名称	参数尺寸	参数数量
Conv1	(8, 8, 32)	1152
Conv2	(8, 8, 64)	18432
Conv3	(8, 8, 128)	73728
Action Conv	(8, 8, 4)	512
Action Flatten	(256)	16384
Action FC	(64)	64
Evaluation Conv	(8, 8, 2)	256
Evaluation Flatten	(128)	8192
Evaluation FC1	(64)	64
Evaluation FC2	(1)	1

其中z是实际对局的身负，1 表示获胜、-1表示失败，0 表示平局。这里优化的参数由三个部分组成。

首先是价值策略，采用平方误差，即$(z-v)^2$；**其次**是网络预测概率p和 MCST 的输出概率差异，由于是一系列概率，故可以采用交叉熵作为损失评估，交叉熵越小，模型越接近真实模拟的分布，对于离散的情况，交叉熵的计算公式为：

$$H(q,p) = -\sum_{i=1}^{n} q_i \log(p_i) \tag{6}$$

令$\pi = (q_1, q_2, \dots, q_n)$，$P = (p_1, p_2, \dots, p_n)$，则可以写作矩阵形式，即

$$entropy = -\pi^T \log(P) \tag{7}$$

最后添加一个正则项，$L_2 = c\theta^2$，这里θ是模型中所有偏置参数，L_2用于控制模型的复杂度，避免模型过拟合。最后将三者和作为实际的 loss，即

$$loss = (z-v)^2 - \pi^T \log(P) + c\theta^2 \tag{8}$$

使用开源工具[14]，编写绘图脚本，网络结构如图 6 所示。

利用公共的卷积部分，最后分为两支分别输出棋面估值和落子概率，可以减少计算量；同时卷积层数在 3~4 层，能够控制训练参数数量，避免需要计算参数数量过多。

在神经网络训练过程中，自我对局并非 AI 自己与自己下棋，而是 AI 和 MCST 模拟的棋面下棋。对于每次子结点**扩展**的数量m，m越大，搜索的空间就越大，获胜的概率越大，但消耗的时间越多。

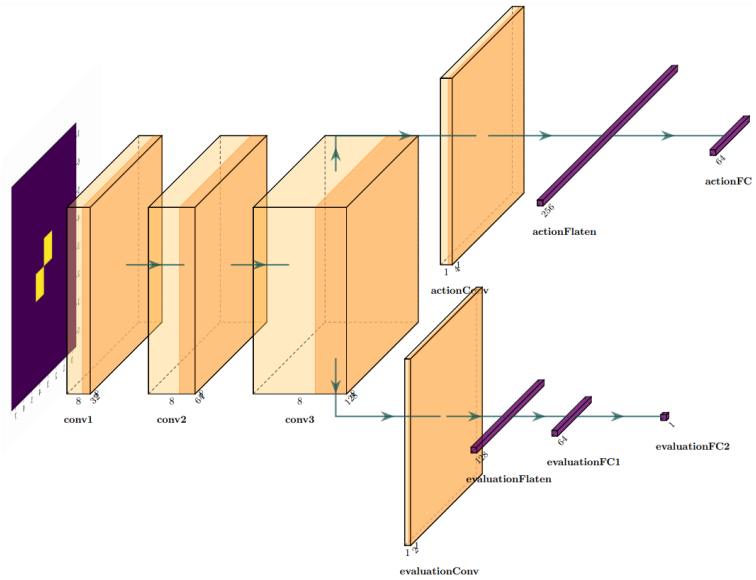


图 6 网络结构图

训练时，每个固定轮数进行一次测试，每当 AI 和 MCST 对局的胜率超过 90% 时，MCST 的m增加一定数量，提升难度，从而不断增强 AI 的棋力。

同时，由于训练过程中需要不断调整参数，因此对于学习率进行了**动态调整**，对于前后两次模型输出的落子点概率p和q，计算 Kullback-Leibler 散度：

$$kl = \sum_{i=0} p(x_i)(\log p(x_i) - \log q(x_i)) \tag{9}$$

KL 散度衡量了两个概率分布的相似度，这里kl不能过大或者过小，通过kl去约束学习率，让学习率先快后慢，一方面，在训练初期能够快速找到最优点附近；在后续精细调参中，需要缓慢调整。

最后将训练得到的价值网络和 MCST 共同构成下棋智能体。

4. 实验结果

4.1 训练模型

实验环境参数如表 2 所示。实验所用设备没有 GPU，因此所有的运算均在 CPU 上完成，耗时相对较长，经过 **70 h** 迭代，最终训练出胜率相对较高的模型。

表 2 实验环境参数

名称	规格参数
CPU	Intel(R) Core (TM) i7
DRAM	8 GB
SSD	512 GB

这里将子结点扩展数量m设置为 6，随后每次增加 2。训练中，每 50 次迭代便进行一次测试，测试中让训练的模型与 MCST 模型进行对局 10 轮，计算胜率；随着迭代次数增加，胜率会逐渐提升，当胜率大于 90%时，增加 MCST 模拟次

数。最终迭代次数为 5000 余次。m最后结果为 32。

训练过程中，总体 loss 曲线如图 7 所示。loss 前期下降相对剧烈，后期稳定在 2.0 左右。

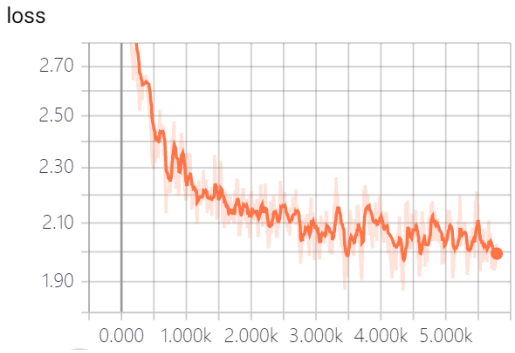


图 7 loss 曲线变化示意图

从交叉熵的迭代变化曲线中，可以看出交叉熵与 loss 变化趋势相近，说明模型也能较好拟合实际 MCST 模拟的结果，如图 8 所示。最终，交叉熵的的损失值稳定在 1.45。

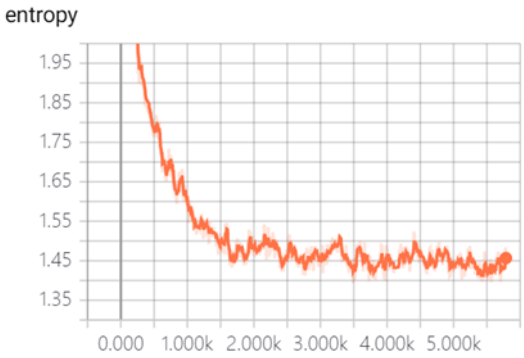


图 8 entropy 曲线变化示意图

4.2 测试

这里模型 MCST 参数m值被设定为合法落子数量的两倍，模拟次数一般在 4 至 20 左右，小于训练 MCST 中的m值。测试采用了第三方游戏平台，这里我们以 https://www.eothello.com/为例，对局过程如图 9 所示。这里我方为黑子，平台一侧为白子。

最后我们以 57：7 获得游戏胜利。从整个局面看，一开始，双方比分相差不大，在第 40 轮时，比分拉开；之后，比分回落，双方势均力敌，在第 59 轮（红框所示）时，我方让白棋没有落子出，因此我方继续走子，此时将比分进一步拉开，最后由于边缘大部分是我方棋子，因此最后一步翻转了对手 8 颗棋子，赢得游戏。

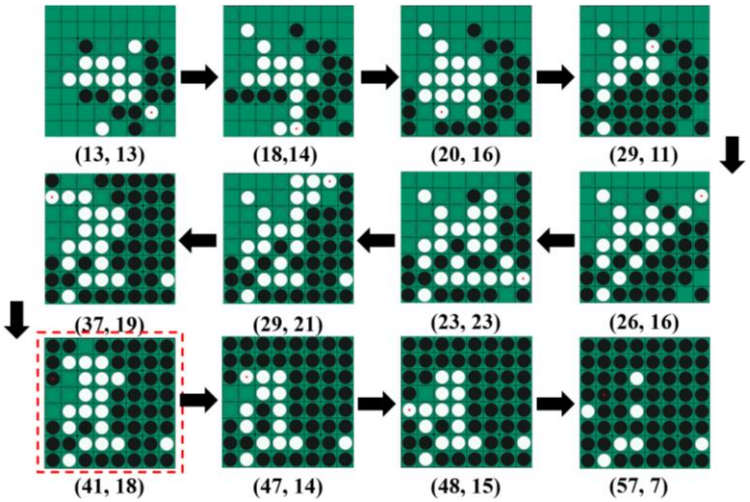


图 9 与第三方平台对局结果

游戏开始时，难以看出双方的优势，但随着游戏进行，黑棋逐渐占据了边缘位置，同时四个角点均为黑棋，这是黑白棋的技巧，边缘子通常比较稳定，在许多静态棋面估值的方法中，也会被赋予较高的权值。

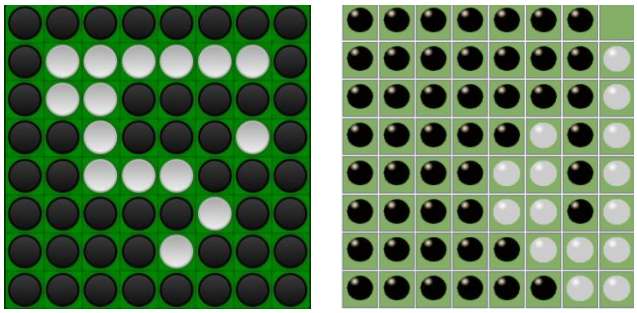
这也说明模型一定程度学习到了黑白棋获胜的关键因素，能够有效平衡得分和占据有利位置。

最后对比了常见平台的黑白棋，每个平台测试五轮，对局结果如表 3 所示。

表 3 各平台对局结果

序号	对局平台	先后手	得分
1	微信-朴素黑白棋	后手	5/5
2	https://www.eothello.com/	后手	5/5
3	https://cardgames.io/reversi/	交替	3/5
4	https://webgamesonline.com/reversi/index.php	先手	3/5
5	https://toytheater.com/reversi/	先手	3/5

这里由于部分平台不支持交换先后手顺序，因此测试中可能只能先手或者后手。在后几个平台的测试中，比分基本在 31：35 上下，双方的棋力相当，难以拉开较大差距，但也有获胜时差距较大的棋面，如图 10 所示。这里是平台 3 和平台 4 的展示。



(a) 49:15 (b) 48:15

图 10 获胜局最后棋面

4.3 超参数对比

模型中诸如初始学习率、卷积层数量和卷积核大小属于经验决策，这里重点比对结点扩展数量m。原始m初值为 6，每次增加 2；现在将m初值设置为 20，每次增加 10。

其余参数保持不变，经过 120h 迭代，loss 曲线变化如图 11 所示。

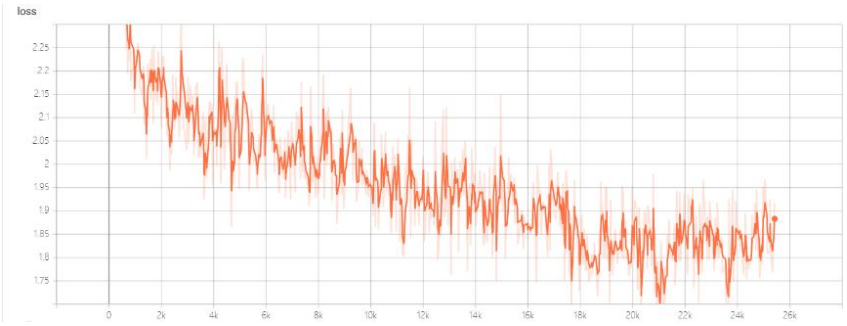


图 11 loss 曲线变化图

loss 曲线前期下降迅速，从 4，迅速降低至 2，最后进一步收敛在 1.9 附近，每当 AI 模型的胜率超过 0.9 时，会增加 MCST 模拟次数，因此每经过一定时间，loss 曲线会有一个短暂上升，然后继续下降，总体呈现阶梯下降趋势。

将原始的 AI 和修改m参数后的 AI 模型进行对局，对局过程如图 12 所示，原始 AI 先手（黑棋）。

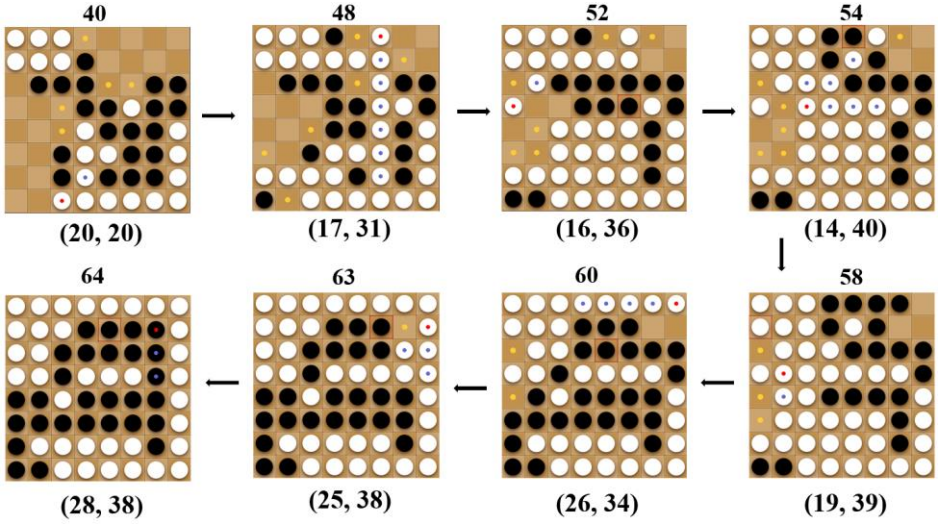


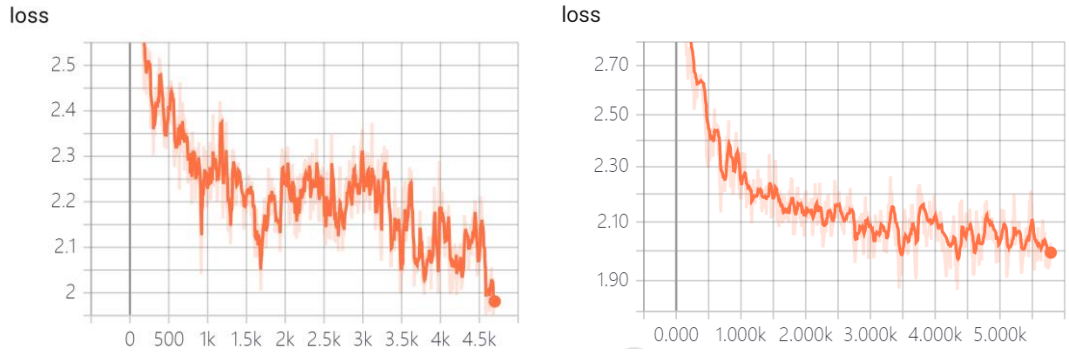
图 12 AI 对局过程图

最后白棋以 38：28 获得胜利。从整个对局过程可以看出，在对局前期和中期，黑白双方几乎势均力敌，比分基本持平；到了第 48 步，白棋关键的一步，翻转了黑棋 6 颗棋子，将比分拉开，随后优势一路领先；在第 62 步时，白棋让黑棋无路可走，于是再走了一步，来到 63，此时比分领先了 13 分，此时黑棋仅有一个点可以落子，最后棋局结束。

从整个棋面来看，从 40 步开始，白棋几乎占据了**边缘**的位置，并且最终占

据了三个**角点**（棋盘四个角）。这是黑白棋的下棋技巧，首先，角点位置是稳定子，因为对方不可以翻转该棋子；其次，边缘位置被翻转的可能性更小，并且有可能会成为翻转的关键，例如在图 10 的 48 步，这里的正是利用边缘子，翻转了黑棋一列的棋子；最后，需要在翻转对手棋子和占据有利位置之间做出平衡。

引入 KL 散度，目的是让模型自动修正学习率，自动调整学习率，让模型更快收敛，若去掉 KL 散度修正机制，训练模型与原有模型对比如图 13 所示。



(a) 没有 KL 散度　　(b) 引入 KL 散度

图 13 KL 散度对模型收敛影响

从图 13 中可以看出，没有 KL 散度修正的模型其 loss 曲线平滑度较差，并且收敛不显著，收敛速度相对较慢。在最终的对局表现中，同样的模拟次数下，没有 KL 散度修正的模型对局效果表现也相对较差。

5. 工程实现

棋盘被抽象为了二维矩阵，但实际传输中，需要对矩阵数据进行编码，因此需要将二维矩阵数据映射为 1 维数据，映射关系如式（3）：

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 24 & 25 & 26 & 27 & 282 & 29 & 30 & 31 \\ 32 & 33 & 34 & 35 & 36 & 37 & 38 & 39 \\ 40 & 41 & 42 & 43 & 44 & 45 & 46 & 47 \\ 48 & 49 & 50 & 51 & 52 & 53 & 54 & 55 \\ 56 & 57 & 58 & 59 & 60 & 61 & 62 & 63 \end{bmatrix} \tag{10}$$

以左上角为原点，向右和向下为正方向，分别是y和x方向，例如，$f(26) = position(3,2)$，反之亦然。

训练完成后，我们需要在实际中与 AI 进行对局，因此需要搭建交互前端。为了保证程序良好的移植性，前端采用 HTML/CSS/JavaScript 搭建，后端采用 Python-flask，二者采用 HTTP 协议通信，所有数据运算和验证均由**后端**完成，前端仅负责数据渲染和用户点击事件处理。架构示意图如图 13 所示。

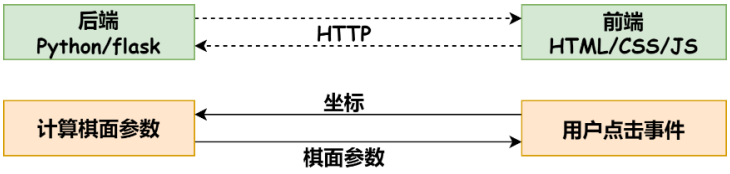


图 13 前后端架构示意图

实际人机交互中，我方为黑子，先手；机器为白字，后手。通过多轮与机器下棋，发现机器胜率较高，说明价值网络有一定作用。如图 14 所示。这里红色点表示落子点，紫色点表示反转对手的棋子所在点。

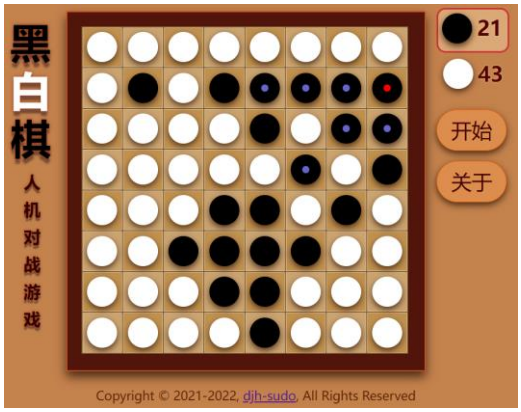


图 14 最终对局示意图

6. 总结

通过使用蒙特卡洛树搜索和强化学习，构建了一个下棋机的机器人，通过与其对局发现，这个智能体表现还不错，在实际的对局中，胜率较大。**代码提交记录也同步至仓库 https://github.com/djh-sudo/AI-Reversi**，下载代码后，启动后端，打开本地 8877 端口即可访问该页面。

限于时间和计算机的计算能力，MCTS 的模拟次数并不多，在计算能力允许范围内，可以增加模拟次数，得到的智能体下棋能力会更好，大部分的五子棋的模拟次数设定在了 4000 次，但由于五子棋迭代速度快，并且走子范围多，因此模拟次数较多，而黑白棋每次走子位置数量在 2-10 个左右，因此模拟次数可以考虑 400 左右，但模拟次数一旦较大，程序运行时间开销也会变多；

其次，模型有多个超参数，这些超参数的优化相对困难，许多是经验准则，只有通过不断调优才能得到一个更好的模型，尤其是学习速率、权衡参数这类超参数，对最后的精度有一定的影响；

最后是网络结构，我们的网络深度在 5-6 层，比起 AlphaGo Zero，这个网络结构要简单很多，因此其表达能力也会有一定的局限性。模型考虑落子在较为稳定位置和翻转对手的棋子数量等因素考虑较多，从实际与第三方平台的对局中，可以看出模型还有提升空间。

参考文献

[1] Kiefer J. Sequential minimax search for a maximum[J]. Proceedings of the American mathematical society, 1953, 4(3): 502-506.

[2] Knuth D E, Moore R W. An analysis of alpha-beta pruning[J]. Artificial intelligence, 1975, 6(4): 293-326.

[3] https://en.wikipedia.org/wiki/Monte_Carlo_tree_search

[4] 刘建伟, 高峰, 罗雄麟. 基于值函数和策略梯度的深度强化学习综述[J]. 计算机学报, 2019, 42(6): 1406-1438.

[5] 杜秀全, 程家兴. 博弈算法在黑白棋中的应用[J]. 计算机技术与发展, 2007, 17(1): 216-218.

[6] 刘佳瑶, 林涛. 黑白棋博弈系统设计[J]. 智能计算机与应用, 2020.

[7] Silver D, Hubert T, Schrittwieser J, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm[J]. arXiv preprint arXiv:1712.01815, 2017.

[8] 陈兴国, 俞扬. 强化学习及其在电脑围棋中的应用[J]. 自动化学报, 2016, 42(5): 685-695.

[9] 王力. 价值神经网络在计算机围棋的优化研究[D]. 北京邮电大学, 2018.

[10] 沈雪雁. 基于蒙特卡洛树与神经网络的五子棋算法的设计与实现[D]. 沈阳化工大学, 2021.

[11] https://en.wikipedia.org/wiki/Zermelo%27s_theorem_(game_theory)

[12] https://en.wikipedia.org/wiki/Reversi

[13] Allis L V. Searching for solutions in games and artificial intelligence[M]. Wageningen: Ponsen & Looijen, 1994.

[14] https://github.com/HarisIqbal88/PlotNeuralNet