目录

[说明： 2](#_Toc76222787)

[摘要 2](#_Toc76222788)

[关键词： 2](#_Toc76222789)

[技术方案的介绍和分析 3](#_Toc76222790)

[一、系统整体构成 3](#_Toc76222791)

[二、五子棋识别 5](#_Toc76222792)

[三、人为设定函数——博弈树、α-β剪枝 9](#_Toc76222793)

[四、ANN和进化计算 11](#_Toc76222794)

[五、强化学习 16](#_Toc76222795)

[六、机械臂 18](#_Toc76222796)

[实验结果的报告和分析 19](#_Toc76222797)

[一、五子棋识别 19](#_Toc76222798)

[二、人为设定函数——博弈树、α-β剪枝 22](#_Toc76222799)

[三、ANN和进化计算 24](#_Toc76222800)

[四、强化学习 34](#_Toc76222801)

[五、机械臂 37](#_Toc76222802)

[后记 41](#_Toc76222803)

**路漫漫其修远兮，吾将上下而求索——李桐人工智能大作业总结**

# 说明：

开始组队时，误了时间，第一时间没有组到队，后便决定不妨自己组成一队，所有事情自己亲自去体会一番，从0开始，一点点学习和进步倒也还是不错的。李桐小组只有李桐一人。

文章整体按照评分规则分为两部分，即 技术方案的介绍和分析 以及 实验结果的报告和分析。

# 摘要

本文从第一视角的试错、探索入手，逐步介绍各个阶段的学习过程，包括 五子棋视觉部分（用摄像头拍照方式识别五子棋下棋过程中当前落子的位置）、五子棋算法部分（博弈树搜索、α-β剪枝，ANN模型，进化计算，强化学习），以及机械臂控制。各方面学习内容较多，有较多的尝试，得到了较好的结果。

# 关键词：

人机交互 五子棋 博弈 视觉 机械臂

# 技术方案的介绍和分析

## 一、系统整体构成

在这部分之中，将后文所述的技术方案的介绍和分析各个部分进行了提炼。

**五子棋识别**

**1. 边缘检测，提取轮廓。**

利用计算机视觉方法提取出棋盘的轮廓，在实验中调节参数使得轮廓是自己想要的。

**2. 透视变换，棋盘矫正。**

经过透视变换，将棋盘进行矫正，变为正方形。

**3. 分割图片，确定位置。**

将正方形19路棋盘图片，分为19\*19个小图片，每个图片的名字就是他的位置，如tp文件夹中的0,0.jpg，指0行0列的棋子。通过调节上述轮廓的参数，使得每个图片的中心都是棋子的位置。

**4. 输入图片，判断棋子。**

每次需要我方下棋的时候调用eye2.py，进行拍照，保存图片，通过上述步骤获得图片，然后提取特征输入训练好的网络中，获得标签值。形成棋盘的矩阵，和旧棋盘矩阵进行对比，就得到了新棋子的位置。

**人为设定函数——博弈树、α-β剪枝：**

需要电脑进行下棋的时候，遍历棋盘上的每一个点，如果存在棋子，则对这个点所在的四个方向形成的四条线分别进行评估。对于具体的一条线，以选取点为中心，取该方向上前面四个点，后面四个点，组成一个长度为9的数组。对于这个线上存在的不同类型的棋形分别进行赋分。

每一个节点对应有一个α和一个β，α表示目前该节点的最好下界，β表示目前该节点的最好上界。利用α-β剪枝剪枝方法进行剪枝。搜索深度为2，传回取得最高分数的位置。

**ANN和进化计算**

**1. 确定特征，构造网络**

每一种颜色的棋子的连五，活四，冲四，活三，眠三，活二，眠二七种牌型的数量作为特征，构造1\*14的特征。label值则是谁获胜，即白棋赢了，label为1，黑棋赢了label为0。构造三层全连接。

**2. 双机对弈，构造数据**

利用上述采用α-β剪枝算法的五子棋博弈，黑白双方都是电脑采用此种方法落子，每局结束时提取特征和标签。构造训练集，进行拟合。之后每次需要落子时，遍历棋盘，对于每个位置提取特征后，输入ANN函数，分数最大的位置即为落子处。

**3. 初次进化，算法僵化**

将ANN的两个矩阵直接相乘，14\*7，7\*1的矩阵，相乘得到一个14\*1的矩阵，也就是直接得到了14个特征的系数。利用上述的方法，分别得到黑子和白子的两个系数矩阵，然后进行了一些细微的调整，各得到了另外4个系数矩阵，这便是所需进化的系数。之后便开始了进化计算。

每一代之中每个算法和其他算法对战1次。即每一代之中对战25次。根据对战的优胜场数进行排名，取前三名留下来，前两名各进行一次变异作为另外两个算法。

**4. 再次进化，多次迭代**

按以上方法算法进化60多代之后由于算法变得僵化了，于是提升对手的难度，让算法和ANN、α-β剪枝算法等进行对弈、进化

**强化学习**

1. 特征: 1\*722矩阵，前361是白字的状态，后361是黑子的状态。

2. 利用DQN的方法，损失就是预测Q值和（max(下一步的真实Q值)乘系数 +奖励值）的差的平方。 预测Q值就是此次神经网络Q值，真实Q值就是曾经神经网络预测的Q值。

3. 加了一个随机事件，一部分按分数最大走，一部分随机，防止套路化。

**机械臂**

**1.位置矫正：**

ArmPi 机械臂开机，电脑连接“HW 开头”的热点，连接到这个热点后，能执行位置校正，使摄像头拍摄到的画面位置方向与实际放置方向保持一致。使校准程序画面的红色“十”字光标与地图中心位置的黑色“十”字图标重合。

**2.检测棋子：**

将棋子放在指定位置上，首先检测检测到放在固定位置的棋子没有移动一段时间后，开始夹取。

**3.计算角度：**

利用夹持器的位置、舵机之间的距离，五子棋坐标等计算能否在可允许移动的范围内找到舵机的移动的俯仰角、计算夹持器需要旋转的角度并且移动过去。

**4.移动夹取：**

之后将机械臂移动到五子棋上，然后夹取五子棋，然后将机械臂抬起。再将机械臂移动到指定位置的上方，将机械臂放下，张开爪子，放下物体，最后还原即可。

## 二、五子棋识别

**方法简述：**

1. **边缘检测，提取轮廓。**

利用计算机视觉方法提取出棋盘的轮廓，在实验中调节参数使得轮廓是自己想要的。

1. **透视变换，棋盘矫正。**

经过透视变换，将棋盘进行矫正，变为正方形。

1. **分割图片，确定位置。**

将正方形19路棋盘图片，分为19\*19个小图片，每个图片的名字就是他的位置，如tp文件夹中的0,0.jpg，指0行0列的棋子。通过调节上述轮廓的参数，使得每个图片的中心都是棋子的位置。

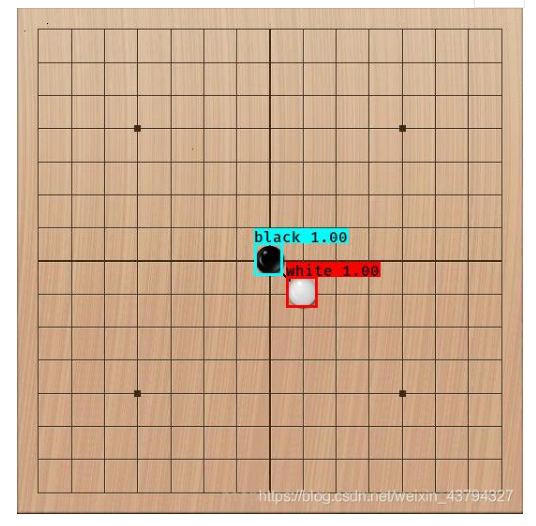
1. **输入图片，判断棋子。**

每次需要我方下棋的时候调用eye2.py，进行拍照，保存图片，通过上述步骤获得图片，然后提取特征输入训练好的网络中，获得标签值。形成棋盘的矩阵，和旧棋盘矩阵进行对比，就得到了新棋子的位置。

**方法详细描述：**

在这个问题上我根据网上的资料以及实现的方法方面想到了三种方法。第一种方法是采用YOLO-tiny（基于keras）的监督学习算法，这是学长们曾经用过的方法。但是基于keras 的YOLO-tiny，我的了解实在太少，网上的介绍也不多，我尝试运行了给出的代码，也有一些库的下载方法太过于繁琐，后来就放弃了。经过了解，tiny yolo是一种非常轻量级的网络，目的是为了其多目标识别的能力所以使用。但是个人认为不必要去费力学习一个不常用的库，没有必要，而且还只能追随前人的脚步。

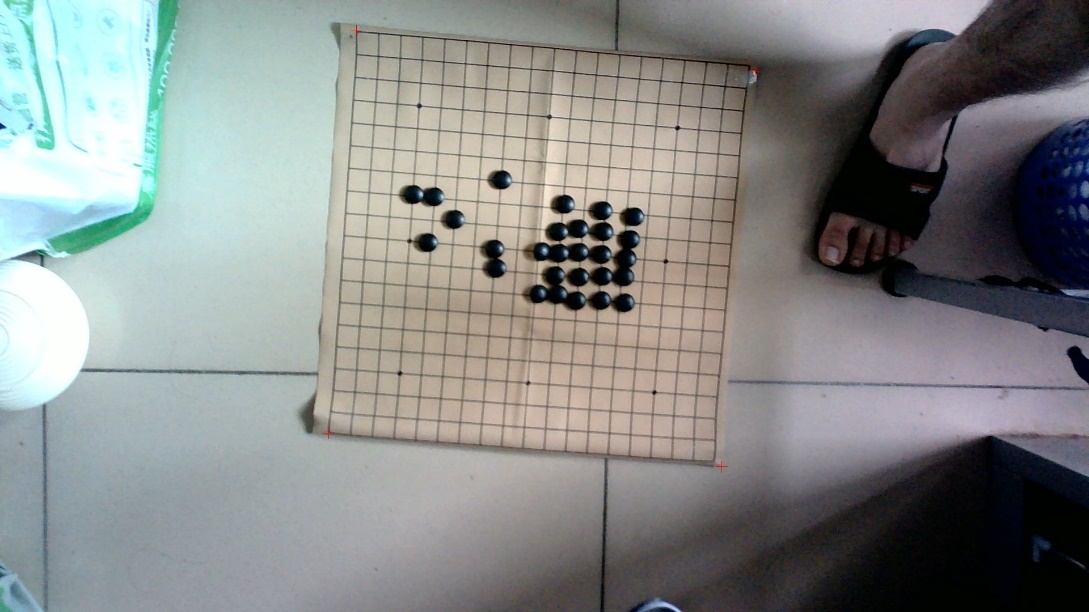
图1.2.1 YOLO



第二种方法是基于opencv的棋盘检测。利用纯粹的计算机视觉的方法进行识别。但是由于要求还要采用监督学习的方法，于是我加以改善，得到了我的方法。

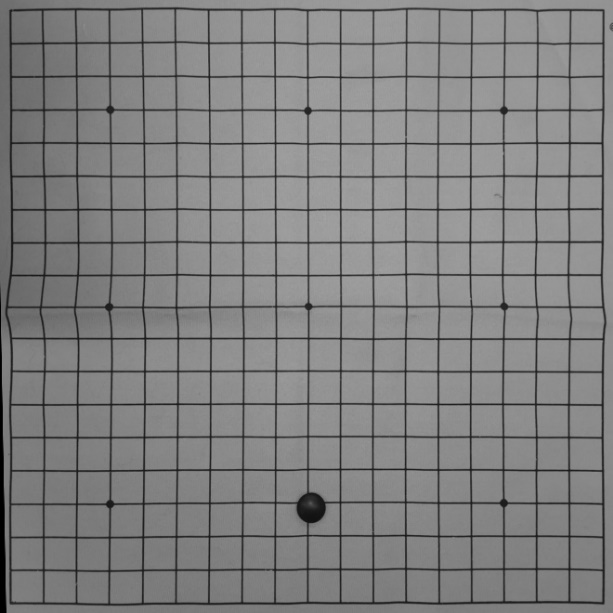
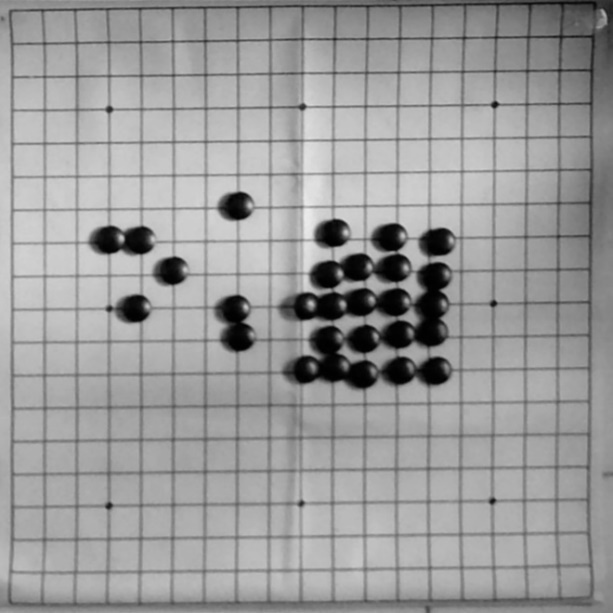
下面介绍我的方法：

图1.2.2 棋盘标定



首先是边缘检测，提取轮廓，找到的最大四边形及其面积（寻找凸包，将凸包拟合为多边形，如果是凸四边形就计算面积）

图1.2.3 透视变换后（1） 图1.2.4 透视变换后（2）



当前找到的棋盘四个角的坐标，之后进行透视变换，变换为正常的矩形棋盘。

将透视矫正后的棋盘，分成19\*19个的图象，代表着19\*19棋盘上的19\*19个位置，存储在文件夹之中，命名的方式与其位置有关，这样就解决了棋子的位置问题，下一步要解决的就是棋子的颜色问题。我手动构建了数据集、利用KNN模型进行了监督学习，当输入图像的时候给出预测结果。

下面是构造棋子识别的训练集的部分，我放置了三盘棋子，一盘黑子、白字交替，两盘单纯的黑子和白子，这样的话会构造出更多的可能。比如有一些空白子的边缘，既有白子又有黑子。

图1.2.5 白子

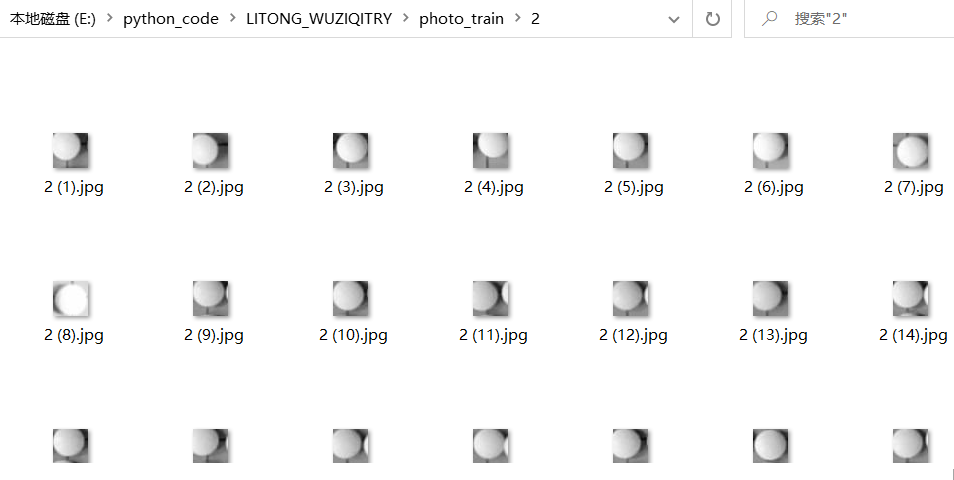


图1.2.6 黑子

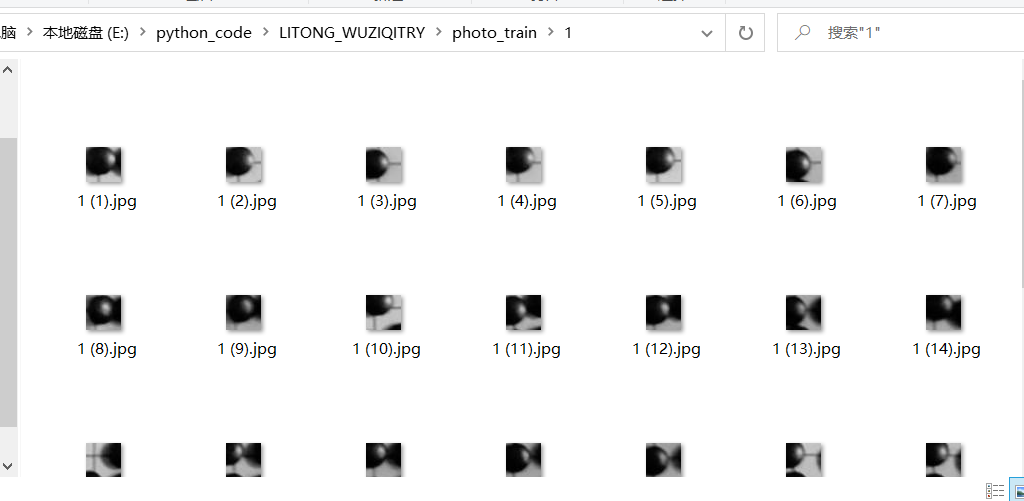
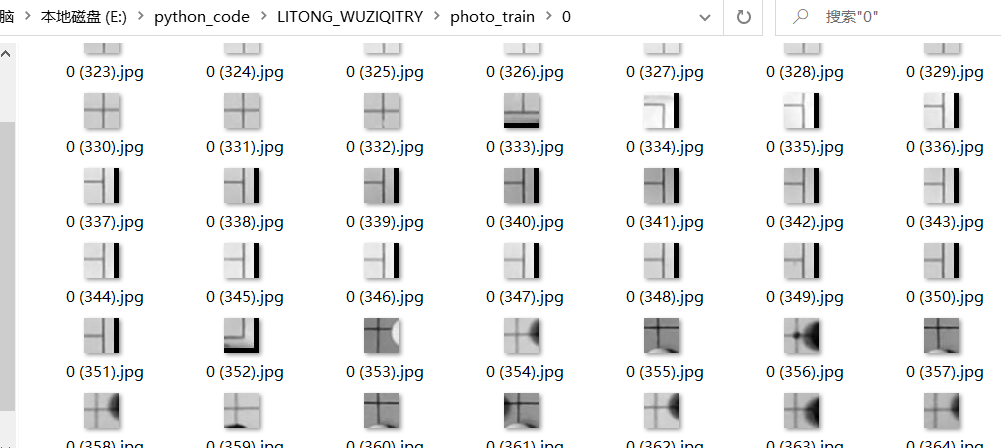


图1.2.7 无子



需要注意的是，我为了模拟在对决中并不会完美的放置棋子，而且棋盘也并非是完美的进行映射和切割，所以特地在摆放棋盘的时候并不是规规矩矩的，有一些扰动的因素的存在，故意摆放的有一些参差，所以在空子的图片中会有一些是有黑子、白子的边缘。由于只有三类，而且很好分类，所以白子、黑子的训练集大概有40个图片，无子的部分由于有棋盘边缘的影响，交叉点的影响、星位的影响、棋子边缘的影响、棋子阴影的影响，所以大概给出了400个不同图片。

之后利用图片的直方图转换成一维的矩阵，就构造好了训练集。然后利用了sklearn种的KNN库直接进行了训练，速度很快，不到1s即可完成。（这部分单独的代码在pre2）

这里说一说这种方法的合理性，构造的训练集是图象，然后利用calcHist函数来计算图像直方图，并且将其处理后转化为一维的向量作为矩阵。然后进行监督学习，相当于进行了分类，分为了白子，黑子和无子三种情况。标签0为无子，标签1为黑子，标签2为白子，预测值更靠近哪一个就分为哪一类。另外在无子的情况中加入了一些干扰因素，所以泛化能力也是很强的。

## 三、人为设定函数——博弈树、α-β剪枝

**方法简述：**

需要电脑进行下棋的时候，遍历棋盘上的每一个点，如果存在棋子，则对这个点所在的四个方向形成的四条线分别进行评估。对于具体的一条线，以选取点为中心，取该方向上前面四个点，后面四个点，组成一个长度为9的数组。对于这个线上存在的不同类型的棋形分别进行赋分。

每一个节点对应有一个α和一个β，α表示目前该节点的最好下界，β表示目前该节点的最好上界。利用α-β剪枝剪枝方法进行剪枝。搜索深度为2，传回取得最高分数的位置。

**方法详细描述：**

五子棋中的获胜方式是五子一线，而如何获胜取决于当前棋局中的形势，最常见的基本棋型大体有以下几种：连五，活四，冲四，活三，眠三，活二，眠二。所以我们可以，创建黑棋和白棋两个数组，对整个棋盘进行遍历，对于每一个白棋或黑棋，以它为中心，记录符合的棋型个数。根据棋盘上黑棋和白棋的棋型统计信息，按照一定规则进行评分。评分之后按照规则再选择下一步的位置。

具体评估方法如下：

遍历棋盘上的每一个点，如果存在棋子，则对这个点所在的四个方向形成的四条线分别进行评估。

对于具体的一条线，以选取点为中心，取该方向上前面四个点，后面四个点，组成一个长度为9的数组。对于这个线上存在的不同类型的棋形分别进行赋分。

表1.3.1 必杀赋分表部分值

|  |  |  |
| --- | --- | --- |
| 类型\棋子颜色 | 黑色 | 白色 |
| 五子连珠 | 1000 | -1000 |
| 活四 | 9050 | -9050 |
| 冲四 | 9040 | -9040 |

表1.3.2 赋分表

|  |  |  |
| --- | --- | --- |
| 类型\棋子颜色 | 黑色 | 白色 |
| 活三 | 400 | 100 |
| 眠三 | 10 | 10 |
| 活二 | 6 | 6 |
| 眠二 | 2 | 2 |

赋分的规则大概如上两图所示，每种棋子各有7种棋形。每次的搜索中，设置搜索深度为2，取最好的评估值。

至于α-β剪枝部分，理论方面大多数人都清楚，具体的实现方面可以解释为如下：每一个节点对应有一个α和一个β，α表示目前该节点的最好下界，β表示目前该节点的最好上界。在最开始时，α为负无穷，β为正无穷。然后进行搜索，max层节点每搜索它的一个子节点，就要更新自己的α（下界），而min层节点每搜索它的一个子节点，就要更新自己的β（上界）。如果更新之后发现α>=β了，说明后面的子节点已经不需要进行搜索了，直接break剪枝掉。博弈树的算法是最简单通俗的，同时效果也是非常不错的。

## 四、ANN和进化计算

**方法简述：**

1. **确定特征，构造网络**

每一种颜色的棋子的连五，活四，冲四，活三，眠三，活二，眠二七种牌型的数量作为特征，构造1\*14的特征。label值则是谁获胜，即白棋赢了，label为1，黑棋赢了label为0。构造三层全连接。

1. **双机对弈，构造数据**

利用上述采用α-β剪枝算法的五子棋博弈，黑白双方都是电脑采用此种方法落子，每局结束时提取特征和标签。构造训练集，进行拟合。之后每次需要落子时，遍历棋盘，对于每个位置提取特征后，输入ANN函数，分数最大的位置即为落子处。

1. **初次进化，算法僵化**

将ANN的两个矩阵直接相乘，14\*7，7\*1的矩阵，相乘得到一个14\*1的矩阵，也就是直接得到了14个特征的系数。利用上述的方法，分别得到黑子和白子的两个系数矩阵，然后进行了一些细微的调整，各得到了另外4个系数矩阵，这便是所需进化的系数。之后便开始了进化计算。

每一代之中每个算法和其他算法对战1次。即每一代之中对战25次。根据对战的优胜场数进行排名，取前三名留下来，前两名各进行一次变异作为另外两个算法。

1. **再次进化，多次迭代**

按以上方法算法进化60多代之后由于算法变得僵化了，于是提升对手的难度，让算法和ANN、α-β剪枝算法等进行对弈、进化

**方法详述：**

ANN：

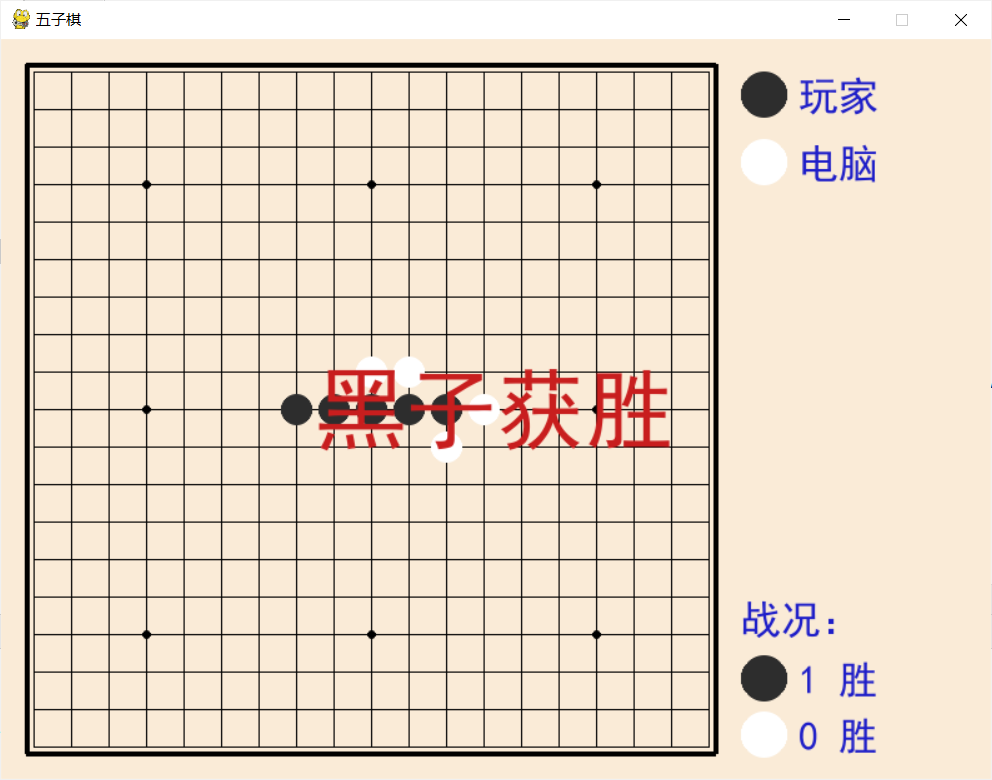
在基于人工神经网络（ANN）的算法之中，我看到了一个别人的算法是这样做的：

利用两个博弈树搜索，让电脑自己和自己进行对战，然后获得棋盘结果，把棋盘由19\*19的变成1\*361的，然后获胜者就是标签值，多进行一些对战，多得到一些训练数据。初始的取值是随机的，后面的取值是根据博弈树搜索的，所以每个棋盘并不是相同的，然后再按棋盘和标签作为x，y值，然后进行人工神经网络的拟合。

这个方法我也试了试，在看的时候就觉得这种方法问题不少，理论支撑基本没有，全是自己进行的臆想，经过自己的实验，果不其然，效果很差。

问题和效果如下。

图1.4.1 笨蛋的ANN



上图是结果，效果可以说是太差了，甚至达到了离谱的程度。哪怕你直连五子，电脑也不会堵。我觉得问题在于，这种算法之中，每个位置所得到的结果就只是和赢局中出现的次数成正比，而这根本没有什么道理、也没什么正确性可言。按照这种思路的话，只要局数足够多，那么甚至可以说这种方法得到的下一步棋子几乎可以说是随机的，和棋盘当前的棋形没什么关系，这样想来电脑输得是真不冤枉。算法失败的原因就陈述到此，这一部分就不进行参数的分析了。具体的优化如下。

回想博弈树的算法中每一种颜色的棋子都有连五，活四，冲四，活三，眠三，活二，眠二。我们遍历棋盘，对于每一个白棋或黑棋，以它为中心，记录符合的棋型个数，再进行赋值，其实这相当于是一个代价函数，评估优势，进行选择。

既然如此的话，函数！我们就可以想到ANN！我们为什么不能够还是延续这种思想呢？基于这种思想，我们需要的训练的参数就是这些种棋形的赋值！

两种颜色的棋子，黑色和白色，各有7种棋形，所以我选取了7\*2=14种特征。

首先还是构造数据集：

还是利用博弈树的算法，让两个电脑玩家相互对战，然后当棋局结束的时候，寻找好14种特征的个数，构造出一个1\*14个矩阵（在保存、传递的过程中别传错了），作为一个x的值，这里需要注意的是，是构建一个评估函数，适用于黑白两种棋子，这里规定把黑棋的形势放在前面，白棋的形势放在后面，即黑棋的特征在前，白棋的特征在后。label值则是谁获胜，即白棋赢了，label为1，黑棋赢了label为0。

之后构造了一个3层全连接，之后再利用上述的数据集进行拟合便好了。

Neuro\_net(

(hidden\_layer): Linear(in\_features=14, out\_features=7, bias=True)

(output\_layer): Linear(in\_features=7, out\_features=1, bias=True))

此处就不列举具体的参数了，可以在ANN文件夹的train.py中获得。在具体的算法中，取消博弈树的赋值，改成了获取当前的各种特征，然后加载模型，输入特征再获取得分。于是这个算法便也结束了。

由于上个ANN算法中，我采取了三层的全连接，所以有两个大矩阵，还有两个激活函数的存在，这就至于这里的进化算法很难进行进化，而且上文中得到的算法已经很不错了，于是我有了这样的一个做法。

进化计算：

我将两个矩阵直接相乘，14\*7，7\*1的矩阵，相乘得到一个14\*1的矩阵，也就是直接得到了14个特征的系数。不过由于我没有利用激活函数，所以参数之中还可能存在问题，于是便开始了进化计算。

采取的方法是这样的，利用上述的方法，分别得到黑子和白子的两个系数矩阵，然后进行了一些细微的调整，各得到了另外4个系数矩阵，之后便开始了进化计算。

每一代之中每个算法和其他算法对战1次。即每一代之中对战25次。根据对战的优胜场数进行排名，取前三名留下来，前两名各进行一次变异作为另外两个算法。

进化了60多代之后，算法僵化，于是又和上述ANN、α-β剪枝进行进化。

## 五、强化学习

**方法简述：**

**第一代算法**

**那全部棋子作为特征，361\*1的矩阵，效果很差，不多赘述。**

**第二代算法**

1. **特征: 1\*722矩阵，前361是白字的状态，后361是黑子的状态。**
2. **利用DQN的方法，损失就是预测Q值和（max(下一步的真实Q值)乘系数 +奖励值）的差的平方。 预测Q值就是此次神经网络Q值，真实Q值就是曾经神经网络预测的Q值。**
3. **加了一个随机事件，一部分按分数最大走，一部分随机，防止套路化。**

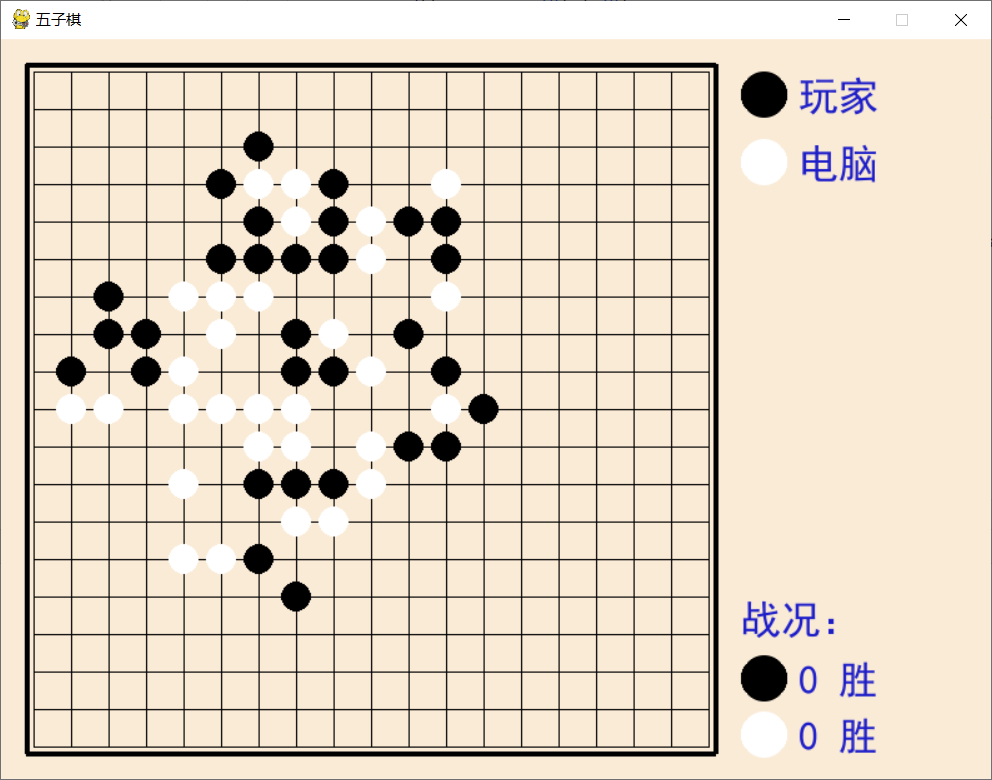
**方法详述：**

在开始实验的时候，我发现网上的强化学习五子棋大概有DQN和MCTS两种。DQN就是搭建一个人工神经网络，输入是当前状态，输出是状态转移价值。或者输入是当前状态，输出是当前状态的Q值。通过多次迭代训练，使得神经网络输出逼近真实的Q值。还有另外的一部分代码采用的是开源的RL(强化学习) Alphago\_Zero 算法。思路和原本的MCTS相同，从根节点 R 开始，递归选择最优的子节点直到达到叶子节点 L。如果 L 不是一个终止节点（也就是，不会导致博弈游戏终止）那么就创建一个或者更多的字子节点，选择其中一个 C。从 C 开始运行一个模拟的输出，直到博弈游戏结束。用模拟的结果输出更新当前行动序列。

我查到了一些MCTS的开源代码，也进行了修改，也进行了很多代的迭代优化、强化学习，不过最终我还是感觉不要用MCTS的算法，由于我用的是一个19\*19的棋盘，真正跑出一个还不错的模型需要的时间需要好多天，甚至不是三两天就能成的，另外一个原因是移植的成本要更大，需要大量的改造，才能拿到我的算法框架之中，于是就放弃了这样的方法。

于是我采取的就是DQN的方法。一开始找到了一位学长的代码，构造出来的效果难以言喻的差，可能由于缺少随机的因素的存在，导致了最开始到后面的效果都不是特别好，都像是在随机下。而且在训练过程中，前后几代几乎没有区别，似乎是按照一样的规则在进行落子。这里的问题我觉得和一开始的ANN问题相似，都是特征选取不好、训练的网络不是很有效、层数太多，以至于开局几次强化的效果没有表现出来。由于效果不理想，于是更改了另外一个DQN。

图1.5.1 对战第一次



数据集的构造是存下每一步的棋盘，然后分成黑子的、白子的各自的，并且转化为一维，即1\*（2\*19\*19）的，前361是白字的状态，后361是黑子的状态。

神经网络的训练的损失就是预测Q值和（max(下一步的真实Q值)乘系数 +奖励值）的差的平方。 预测Q值就是神经网络一次前向传播输出的Q值，真实Q值就是神经网络曾经预测的Q值。这个生成真实Q值的网络不需要训练，只需要迭代一定次数以后，复制一份预测网络的参数即可。就好比一个笨老师教一个学生，学生学会了以后当了老师，教新的学生，然后青出于蓝而胜于蓝，这个学生越来越强。

另外对比前面的各种算发这里加了一个随机事件，一部分步子是按照最大值去走的，一部分步子是随机走的。当没有什么特别好的方法的时候就随机走，防止固定套路的获胜。

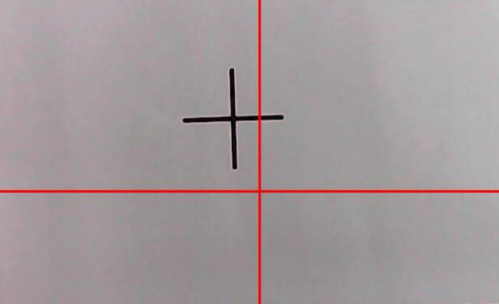
这里的效果比第一个的DQN好得多，不过因为特征选取的不是最好的，迭代次数也不够，所以效果还是不是很好。

## 六、机械臂

**1.位置矫正：**

ArmPi 机械臂开机，电脑连接“HW 开头”的热点，连接到这个热点后，能执行位置校正，使摄像头拍摄到的画面位置方向与实际放置方向保持一致。使校准程序画面的红色“十”字光标与地图中心位置的黑色“十”字图标重合。

图1.6.1 校正位置



**2.检测棋子：**

将棋子放在指定位置上，首先检测检测到放在固定位置的棋子没有移动一段时间后，开始夹取。

**3.计算角度：**

利用夹持器的位置、舵机之间的距离，五子棋坐标等计算能否在可允许移动的范围内找到舵机的移动的俯仰角、计算夹持器需要旋转的角度并且移动过去。

**4.移动夹取：**

之后将机械臂移动到五子棋上，然后夹取五子棋，然后将机械臂抬起。再将机械臂移动到指定位置的上方，将机械臂放下，张开爪子，放下物体，最后还原即可。

# 实验结果的报告和分析

## 一、五子棋识别

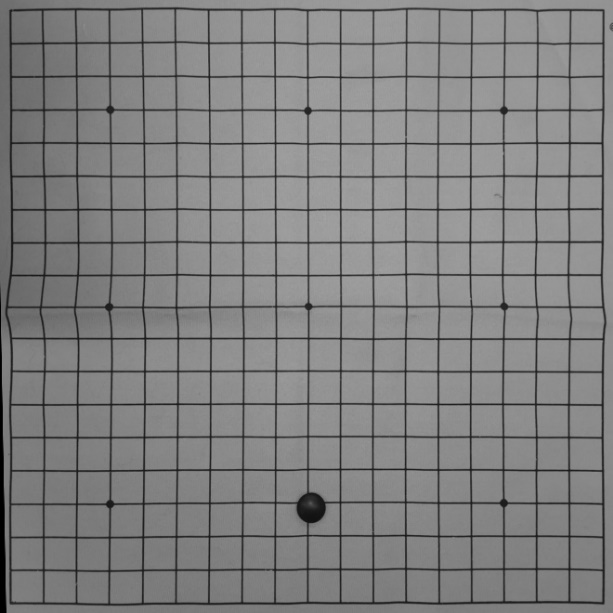
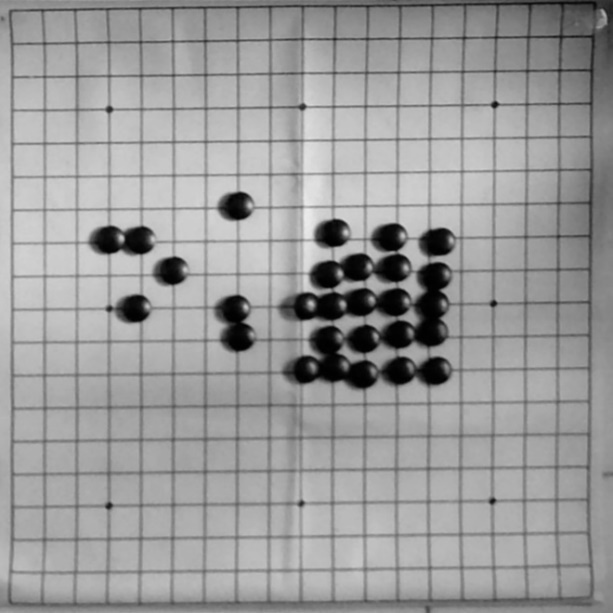
**实验结果:**

1.初始照片过于混乱时识别效果差

2.折叠式棋盘识别效果差

3.正常识别情况

图2.1.1 透视变换后（1） 图2.1.2 透视变换后（2）



由上图可以看到，第一个的拍摄过程是十分混乱的，不过还是得到了正确的提取。

图2.1.3 折叠式棋盘





**实验分析：**

首先是棋盘识别的问题，这个是可以理解的。

当棋盘下面是其他方形物品时也会得到一些奇怪的图形，甚至棋盘会有整个的扭曲。

当原本的照片就过于混乱，识别后的照片已经通过了仿射变换，原本就扭曲的图像也不能恢复原样。比如上图中虽然图片是方形的，但是可以明确看到，放射变换后的棋盘不是方形的，这是因为原本的棋盘就没有摆放好。另外可以看出棋子边缘的阴影也很多，这是因为原来便有阴影，之后仿射变换后原本的阴影便随着棋子的变大而变大了。

不适用折叠式的棋盘，因为折叠式棋盘中线的存在，只能够识别随机一半的棋盘，另一半会产生问题，或者会识别成一个奇怪的图形。这方面的原因我仔细思考之后觉得和算法本身并没有什么关系，更多的原因在于本身，折叠式的棋盘中间的线比较粗，确实容易被认为分割了整个大方形。当棋盘下面有方形物品、有其他直线的时候也会给机器一个错觉。我认为这方面并不需要完全避免，只需要在操作过程中注意一下即可。

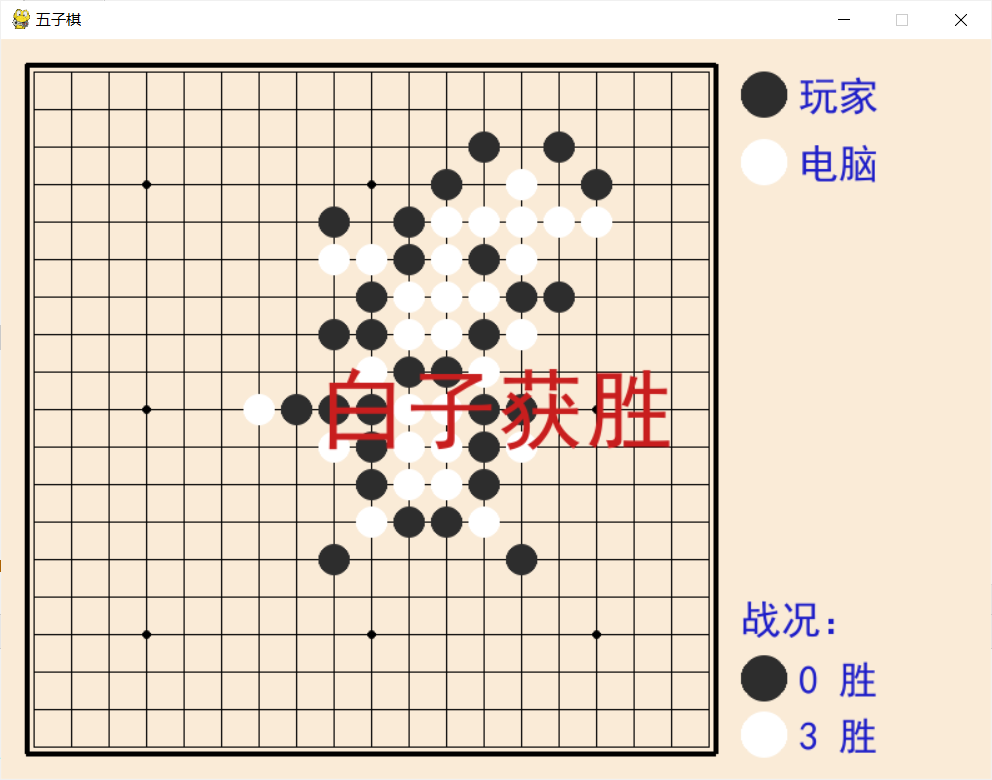
**（参数分析）**这里涉及到的参数主要是分割棋盘的参数，这里取的仍是前面用到的35像素。这里说明一下这个参数的来历。（660-10）/19=34.2用棋盘长度减去前后空余的像素大小，但是一开始选取的并不是34像素，而是30个像素，得到的棋子的位置有误，有的棋子的位置偏高，有的棋子的位置偏低，采用加偏置项的方法失败了。后来开始进行了调参，参数为35的时候刚好可以解决问题。另外还会涉及一些参数，比如纯计算机视觉方法中用到的霍夫检测中圆的半径，整个棋盘的像素等等，不过这类参数影响不大，不在这里进行说明了。

**（优化分析）**其实目前的整体算法还可以，识别起来基本没有问题。不过我自己觉得是我稍微取巧了的，采用了计算机视觉识别棋盘+机器学习识别棋子的方法。后续进一步优化可以优化为全面的机器学习识别棋盘、棋子、坐标。

## 二、人为设定函数——博弈树、α-β剪枝

**实验结果：**

图2.2.1 博弈树搜索



这是我和以博弈树算法为核心的对战中的截图，真的是一不小心，就会崩盘，甚至可以说，无论是前期的布局还是中盘的厮杀，我都不是电脑的对手。基本上没有什么问题，运行的结果非常让人满意。唯一可能有一些问题的地方在于每次都是取搜索的最大值，缺少一些变化吧。不过这个问题中，并不涉及进化之类的问题，我就也没有进行改动，毕竟只要电脑选取最优的打败对手即可。

**实验结果分析：**

这个实验的结果是我非常满意的，因为我已经是打不过电脑了的，水平已经在我之上了。不过我觉得这个算法结果很好的原因主要是在于人工赋值的参数的选择上，下面我会分析参数以及日后的优化。

**（参数分析）**这里涉及的参数有两类。一类是每种棋形的分数，这个合理即可，让多子连珠的分数更高就好了。不过这里采用的方法是能得到更好的就得到更好的，所以类似于活三和眠三的得分会比较大，避免因为一个眠三+一个活2，就放弃了一个活三的情况的出现。否则电脑可能也会选择实际情况的次优解，不过由于进行了两层的搜索，搜索空间比较大了，参数的取值范围还是很包容了。

第二种参数是这里的搜索层数，搜索层数太多的时候搜索起来会特别慢！在实验的过程中，我依次设置了搜索深度为1，2，3，4，体验特别明显，1层的时候立刻就可以算出来了，2层的时候几乎是秒出，但是运算时间大概会达到4-5s左右，3-4层就更不用说了，体验特别差，运算时间特别长。3-4层时间长是可以理解的，因为搜索的范围实在是太大了，2层之所以出现时间的波动我觉得可能的原因是有的时候剪不了枝、或者剪枝剪的少，棋形很复杂，所以需要计算很长时间。尤其是关键一步的时候，电脑不能不堵住我的时候，算的时间会长一些。最后选择了2层，因为综合的效果会更好，算力足够了而且用时相比比较合适。

**（优化分析）**这个问题其实很简单，优化的方面不多。可以考虑优化的方向变成当有一些点的得分相似，就在这几个点中随机进行选择，增加多样性。

另外还可以改成用户自己设置层数的算法，即为自己选择难度，难度高搜索的层数高，难度低搜索的层数少。这一点的优化实现难度很低很容易实现。

另外还可以增加一些其他的剪枝策略，比如当对方现在有活三和眠四，而我方没有必胜下法的时候，即没有活四、眠四、活三的时候，让电脑直接在能够围堵的地方选择更好的进行落子。这样也可以省区大量时间，尤其解决是上文提到的“电脑不能不堵住我的时候，算的时间会长一些”的问题。

## 三、ANN和进化计算

**ANN实验结果：**

图2.3.1 超厉害的ANN（1）

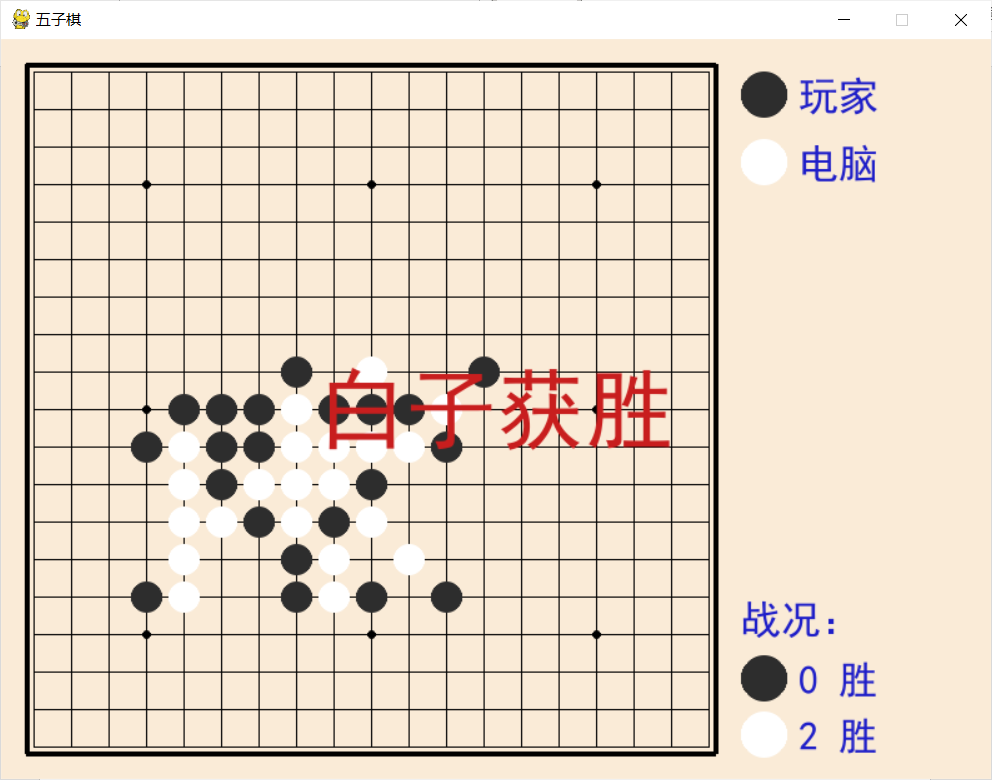
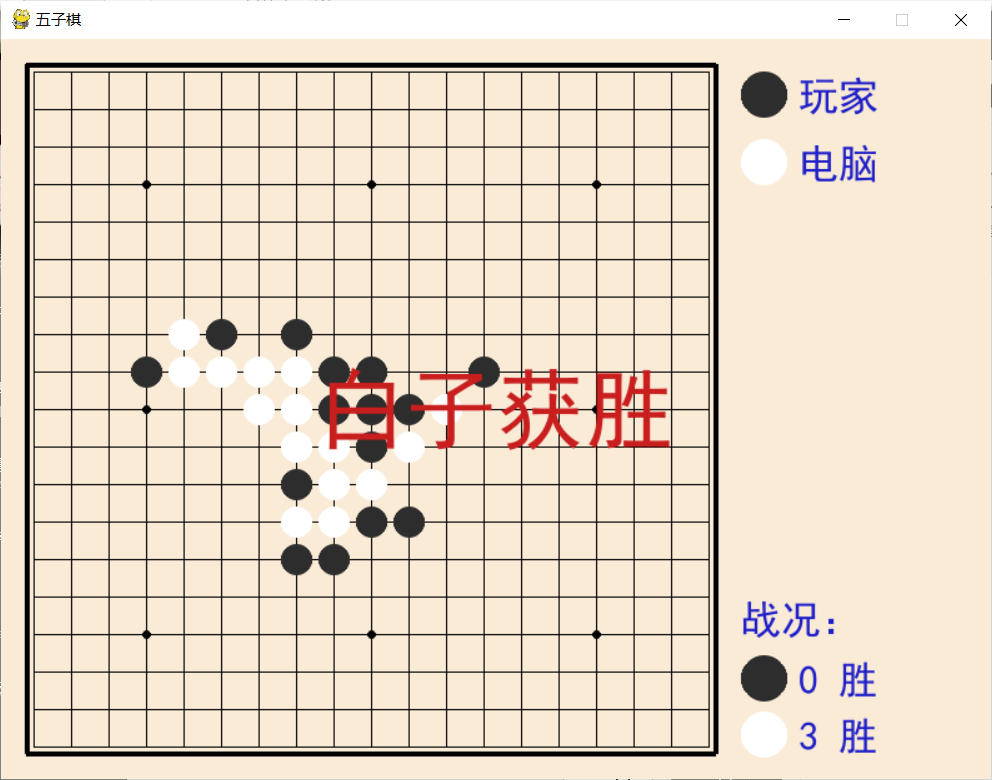


图2.3.2 超厉害的ANN（2）



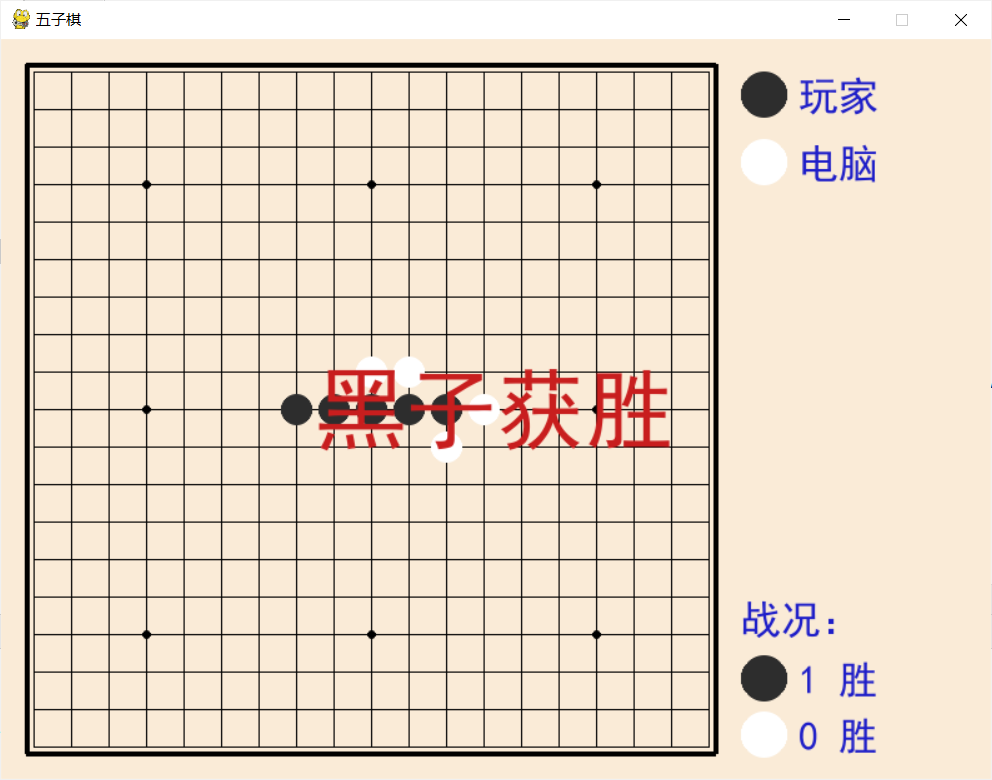
上面是我和ANN对战的两个图，结果相当震撼了，下的又快，下的又好，我都懵了。别说什么前期中期还是后期了。我根本挺不了几回合就没了。前期把我堵得死死的。而我但凡错了一步，算法的攻势就是滔滔不绝，上面两个图里我堵住的基本都是四子连珠了，输的都是必杀局了。个人认为结果很好，没有什么问题了。

**ANN实验结果分析：**

ANN最终的结果甚至可以称得上是我最好的几个结果之一。为什么上面这个结果会这么好呢？

首先回想一下之前提到过的我看到的笨蛋一样的ANN。

图2.3.3 笨蛋的ANN



这个算法是采用的思想是整个棋盘的形势即把棋盘由19\*19的矩阵变成361\*1的矩阵作为特征，输赢结果作为标签。上图是我运行的结果，我尝试了好几次，效果可以说是太差了，甚至达到了离谱的程度。哪怕你直连五子，电脑也不会堵。我觉得问题在于，这种算法之中，每个位置所得到的结果就只是和赢局中出现的次数成正比，而这根本没有什么道理、也没什么正确性可言。按照这种思路的话，只要局数足够多，那么甚至可以说这种方法得到的下一步棋子几乎可以说是随机的，和棋盘当前的棋形没什么关系，这样想来电脑输得是真不冤枉。

那么为什么我的ANN的结果会这么好呢？结合之前失败的例子，我认为主要还是因为采用了合适的特征和合适的参数值。五子棋中的获胜方式是五子一线，而如何获胜取决于当前棋局中的形势，最常见的基本棋型大体有以下几种：连五，活四，冲四，活三，眠三，活二，眠二。这是我们在博弈树处的思路，一方面，的确输赢是由棋局形势决定的，所以自然而然应该选择这个作为特征，另一方面，延续博弈树的思想我们也会选择这个作为特征。这是特征选择方面的优势。

参数方面的优势：这里的训练集是是根据了300多盘接近400盘的电脑自己对战产生的，是由此总结出来的经验学习，进行拟合出来的的参数，所以效果很好。而跟上面博弈树算法的参数相比，是对它的参数的进一步优化。

**（优化分析）**我觉得这个算法已经很美丽了！不需要特别多的改善。如果想要改善的话可以从特征的角度和训练集的角度出发。特征可以增加更高维的，用更多的数据训练，自然可以得到更好的。

训练集的话，是利用上述博弈树算法生成的，如果时间允许的话，可以将算法设置为4层，乃至更高层，得到更有质量的数据集，同时还可以将算法中增加变化性，增加上文提到的随机选择的方法，提高训练集的多样性。

**进化计算实验结果**

1.前期进化速度慢。

2.没进化时可以五子连珠直接获胜，进化10多代的时候效果不是很好，电脑输得也很快，甚至在发现赢不了之后 “自暴自弃”，如果开局让两子，下两步臭棋，电脑可以在中期厮杀中取得极大的优势。

3.进化10多代之后还是会错过必杀之局

4.进化60多代之后算法运行极快，但是算法僵化

5.进化60多代后和我对局基本不败，但是十分套路化

6.改为和ANN进行博弈后，进化缓慢，同时十分难以获得胜利，基本全败，大部分时间在做无用的进化。

7.最终我坚持不了几手就被打败了

**进化计算实验结果分析：**

1.开局进化缓慢是因为两方的实力相当，都需要搜索很久才能得到最好的结果。所以每局游戏进行时间很长。在进化的过程中，我还进行了几次干预，比如提高五子连珠的系数，比如按照我的思路给14个特征系数赋值。运行了10多代之后，最后留下来的都是经过我修改的特征系数的后代。

2.

图2.3.4 进化算法（1）

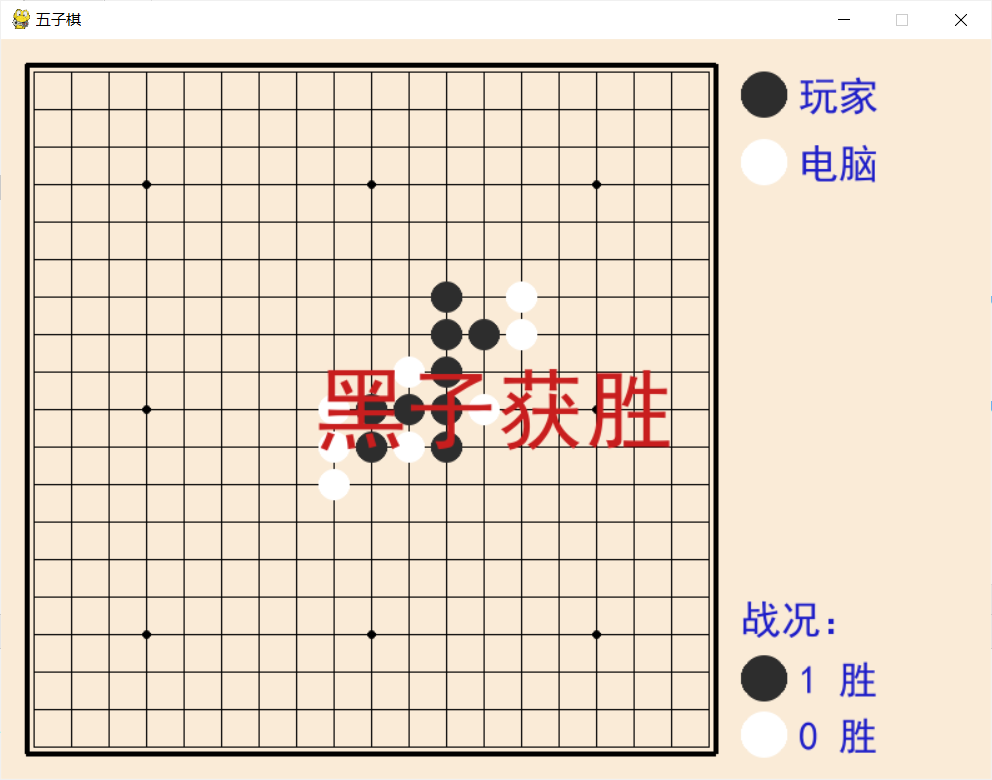
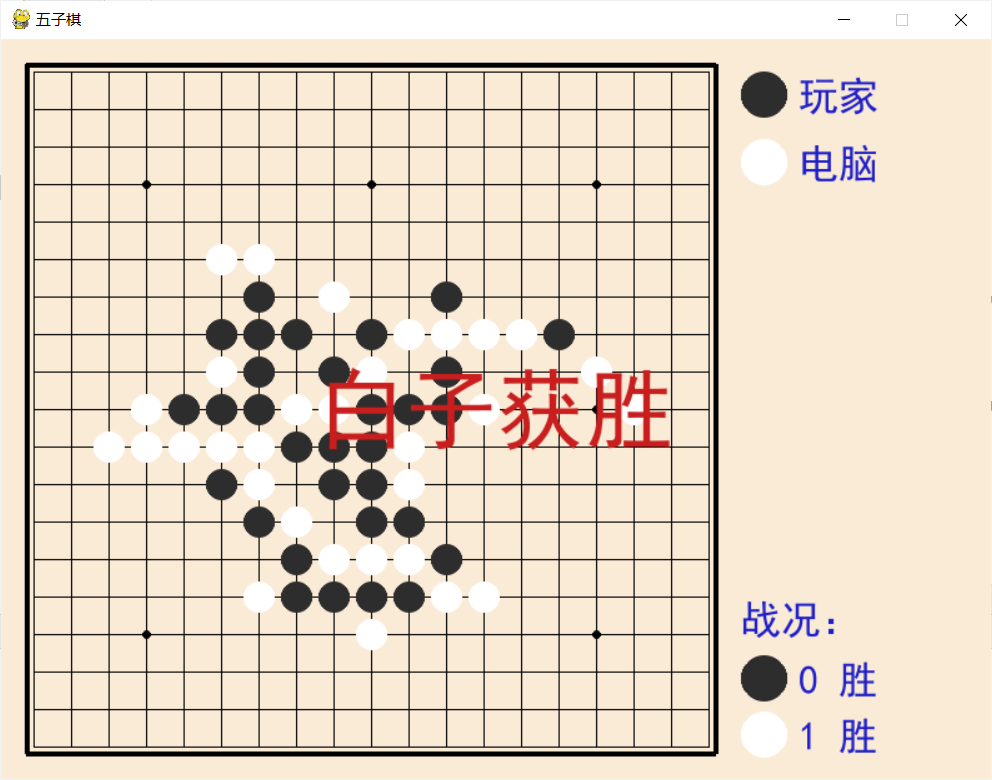


图2.3.4 进化算法（2）



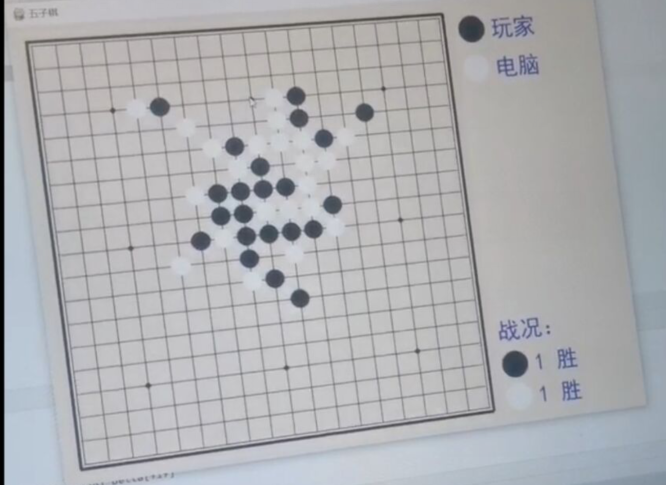
这是我的两个截图，可以看到上图之中白子的获胜要更为艰难。实际上是这样的，第一个截图是我没有让子的时候取到的最差的一把，电脑很快就输了，甚至在发现赢不了之后 “自暴自弃”，这倒是可以理解的，电脑是这样的思路：赢不了了，去堵反而得分更低，而构造自己的连子反而得分更高。有时也会为了自己的得分更高，拒绝堵子。

其实这已经是进化了的结果了，第一代的时候也几乎是我雷霆之势，电脑还没有堵我，我就赢了，进化了10多代之后开始懂得堵住我一些了。

第二张图是我开局让了两子，开局起势之后下了两步臭棋。在这把之中，其实算法错过了一个必杀之局，不过最后还是赢了。

3.下面举一个我在录像中录到的电脑错过必杀局的例子：

图2.3.5 必杀1



这里是开局，此时已经是两个活三了，白子大优势，轮到我下子了。

图2.3.6 必杀2



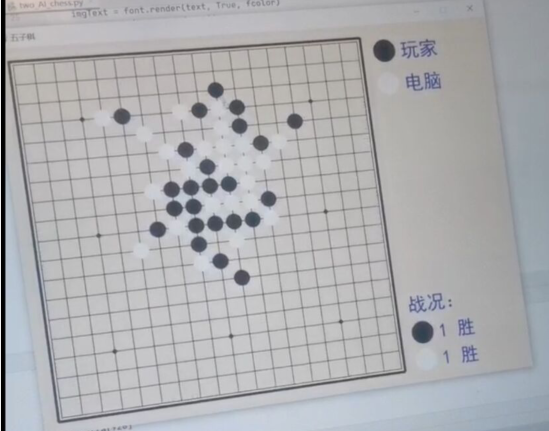
如果电脑下在这几步我一定是必败无疑了。于是我下在了4之9的位置上，也就是左上的绿色点的位置上。此时电脑下在3之9的位置，也就是最上面的绿色上我也是输了。不过电脑为了更高的分数居然下在了7之9，虽然能构成眠四，但是已经是错过了对我的必杀之势了。

图2.3.7 必杀3



对于电脑来说，下在7之9构成了1个眠四和一个眠三，可能分数要比一个活四的分数更高，所以它采取了这种方法。不过却让我就避免了一场必输。

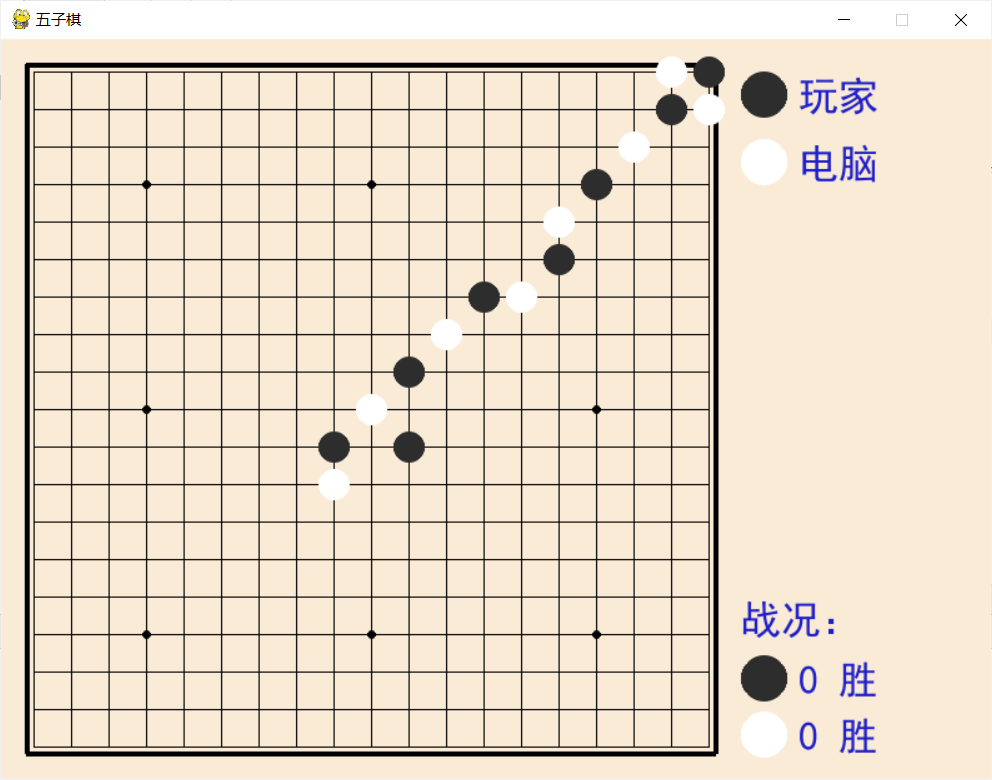
图2.3.8 必杀4



不过这局的最后我还是输了的。总结一下，进化的代数不够，以至于开局的电脑的布局能力不够，只是单纯的想着围住对面。但是中期由于棋子已经有了很多了，不同方向很多，所以厮杀能力很强。

这里举出一个进化算法进化时的截图，由于两个棋子都想着堵住对面，开局的时候不想着趁着没什么危险，赶紧自己布局，而是一直在赌。于是出现了如图2.3.9的这样的现象。

图2.3.9 进化中的有趣的图



4.

图2.3.10 进化60代后（1） 图2.3.11 进化60代后（2）

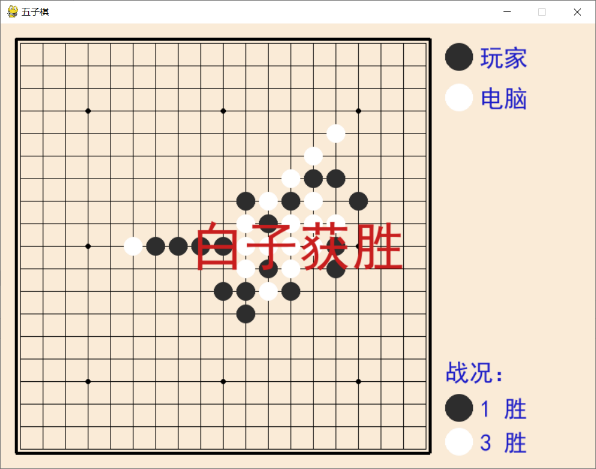
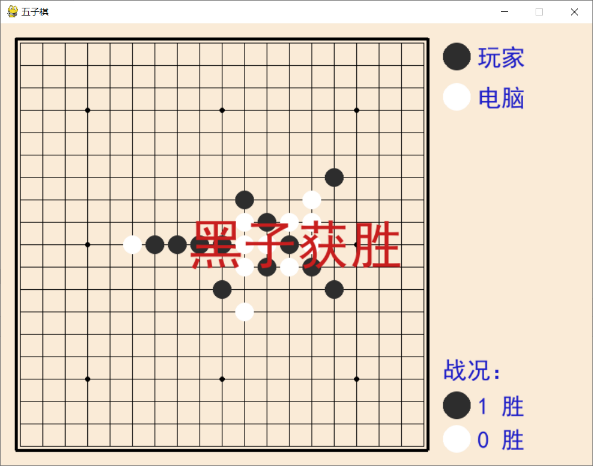
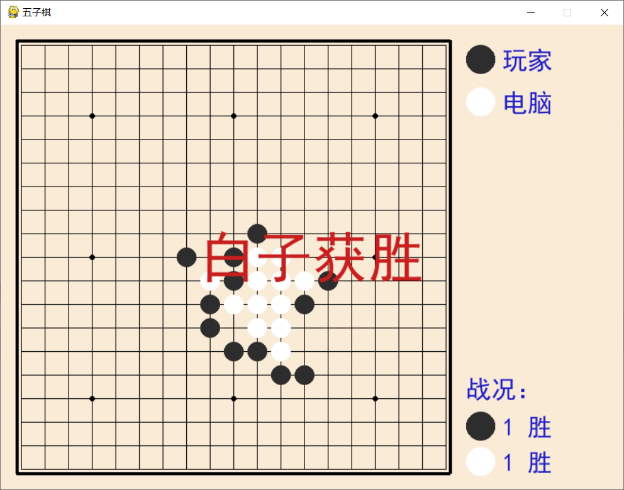
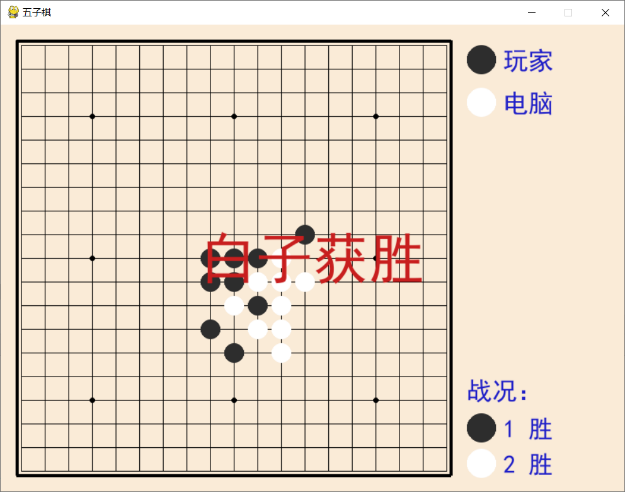


图2.3.12 进化60代后（3） 图2.3.13 进化60代后（4）



进化60代后，算法变得很智能，我在观察进化的时候发现，白棋好像出子变得越来越套路化，每一代进化很快，而且黑子从来赢不了。于是就停止了进化。

我和电脑下了几盘，发现电脑的下子非常套路化，另外进攻性非常强。我推测出现这种现象的原因是起初几代白字获得了优势，于是这种优势越积攒越大，白子发现这种策略能一直赢黑子，就一直向这种方向进化，慢慢地形成了它的策略，由于黑子一直处于弱势，以至于白子可以慢慢放弃防御性，转而追求极致的进攻性。

5.运行60多代之中，白色棋子偶然一次获得了特别好的变异，于是胜率飙升，慢慢地这个算法产生的后代也越来越好，针对对方算法有很强的竞争优势，逐渐黑棋不再能够获得胜利，算法运行越来越快，白棋算法在和黑棋中效果越来越好，但是同时越来越僵化，进攻性越来越强，和人对战的时候显得尤为清楚。

6. 在发现上述问题之后，于是我改变了策略，既然黑子的策略已经跟不上白子了，不妨不让黑子作为白子的老师，转而让前面的搜索树、ANN作为对手，让白子的策略继续进化。

果然在开始的阶段，由于进攻性放弃了部分防御的白子吃了瘪。这样的思想固然可以，但是我发现好像有些拔苗助长了。进化的效果反而变得有些差了。我推测出现这种问题的原因是博弈树搜索效果太好了，对战的时候进化算法中白棋的策略每次都打不过，白棋基本不能获得胜利，算法只能盲目进化，随机进化所以进化之中很多次都是无意义的变异。我后来改变了进化方法，原来进化的时候一个个体只有一个系数变异，改为了全部系数都会变异。

7.目前为止，最好的进化计算已经可以说是我的最好效果之一了。目前对战电脑学会了从开局就布局，整体能力非常强，老谋深算。我已经赢不了了，只能在它的手里坚持一段时间。

## 四、强化学习

**实验结果：**

1. 第一次用的DQN效果很差，基本上是随机下棋，不会堵人。起初几次训练没有什么效果。
2. 第一代DQN训练后可以看到进步，但是依旧不是很好。

3.改进后的第二代DQN效果开始效果很差，训练后效果也不是很好。

**实验结果分析：**

首先第一代DQN开始效果很差，训练一段时间效果也没有特别好。我认为主要原因在于特征的选择上。

图2.4.1 开始的第一代DQN

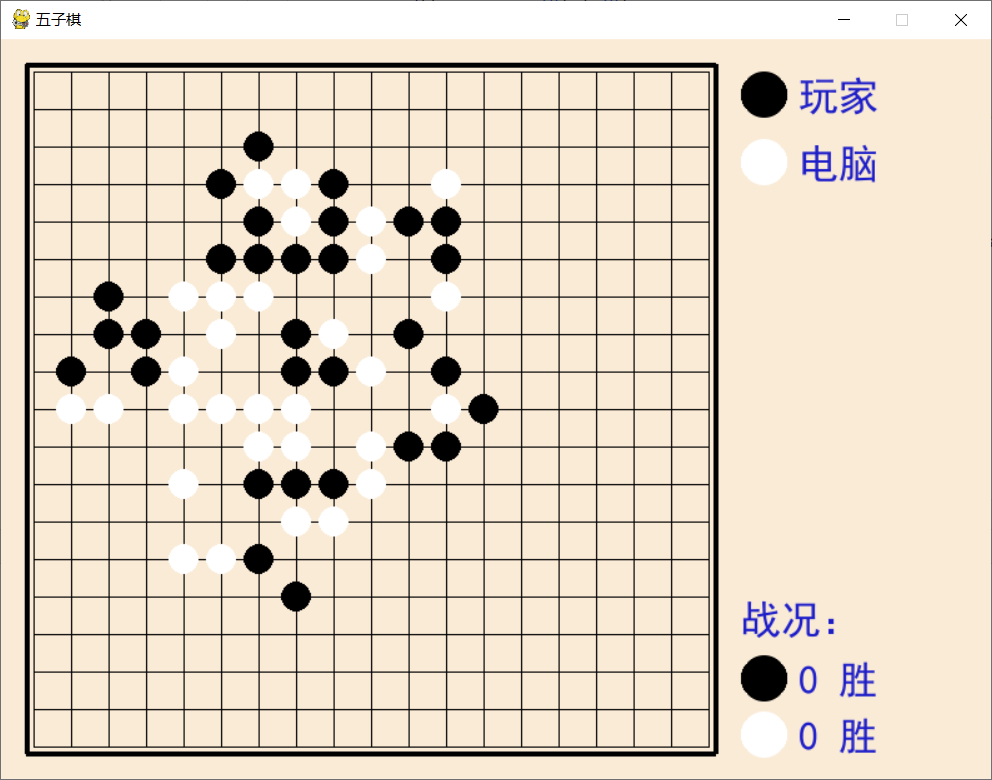
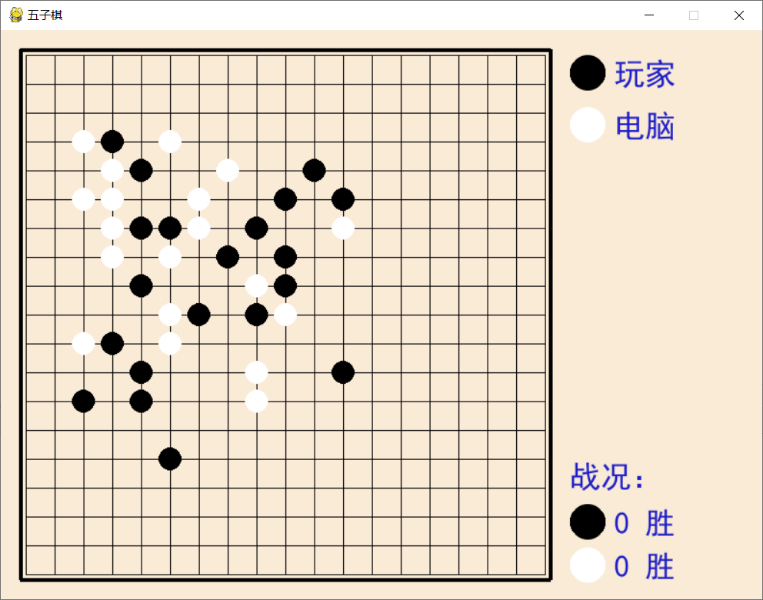
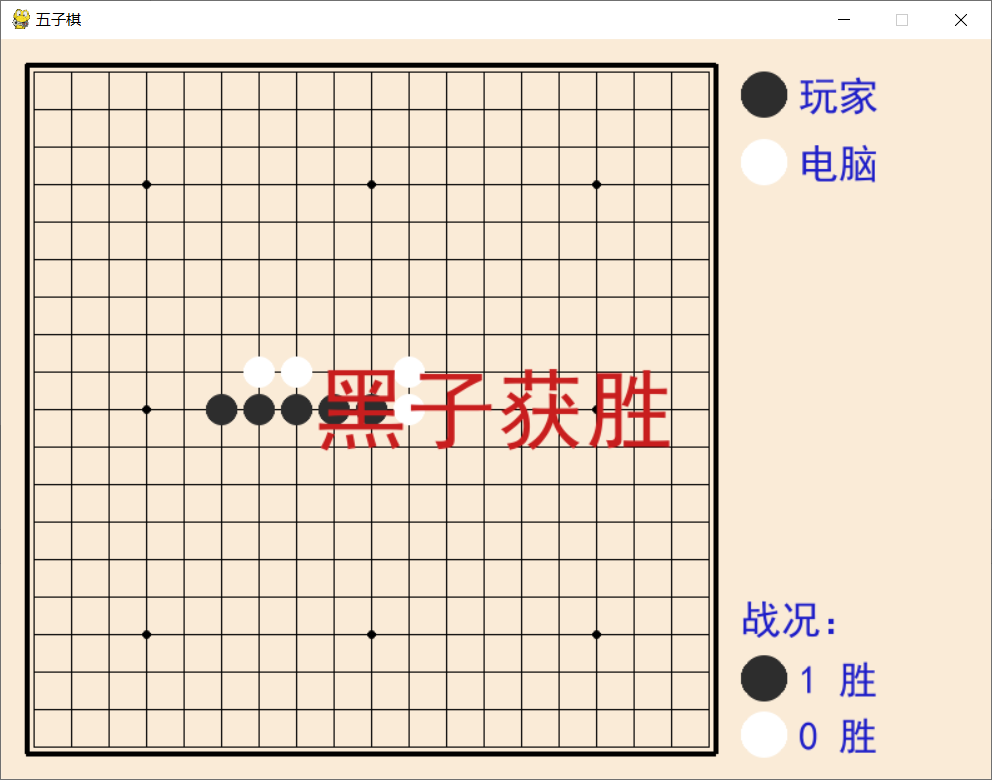


图2.4.2 训练一段时间的DQN



从一开始的训练时的杂乱到后来的逐渐有序，虽然训练后与人对战都没有什么还手之力，但是还是可以看出来效果的提升的。主要问题我认为有两个，一个是特征选择，真个棋盘作为特征，没有代表性，像是随机在下。而且游戏输赢更多的在于局势、不同的棋形。整个棋盘输入进去的话，可能需要大量的训练才可能出效果。另外就是可能网络有一些深，需要很多次迭代才能有效果，起初的几次强化没有什么效果。

图2.4.3 第二代DQN训练前几代的人机对战



改进后的第二代DQN效果开始效果很差，训练后效果也不是很好。一开始的效果很差，但是要比第一代的要好一些主要的原因我认为是特征的选择上，这里是将黑白色棋子分开了，重新组成了一个数组作为特征。

之后训练了10w步左右，效果确实有提升，但是不是很多的提升，效果变好了，但是没有特别好。主要原因我认为和之前相同，应该还是特征的选择上和训练的难度上。特征并不是很清晰，如果把棋形作为特征，可能效果会好得多。另外就是深度过深、参数过多以及特征选择不好，导致得训练步数需要很多很多才能取得一个好的结果。10w步的训练还是效果不明显。

## 五、机械臂

实验结果

最终能够夹取五子棋放在指定的位置上。

实验结果分析

机械臂控制部分是根据机械臂源代码进行改写的，能够正常运行并不意外。此处分析一些代码。

图2.5.1 机械臂（1）



先把五子棋放在指定位置上，然后对摄像头进行调试。

利用摄像头获得图片，对图像和掩模进行位运算，分别进行开运算和闭运算，找出最大轮廓，获得五子棋中心的坐标。如果两次坐标差距很小，认为没有移动，没有移动的时候进行下一步操作。

图2.5.2 机械臂（2）

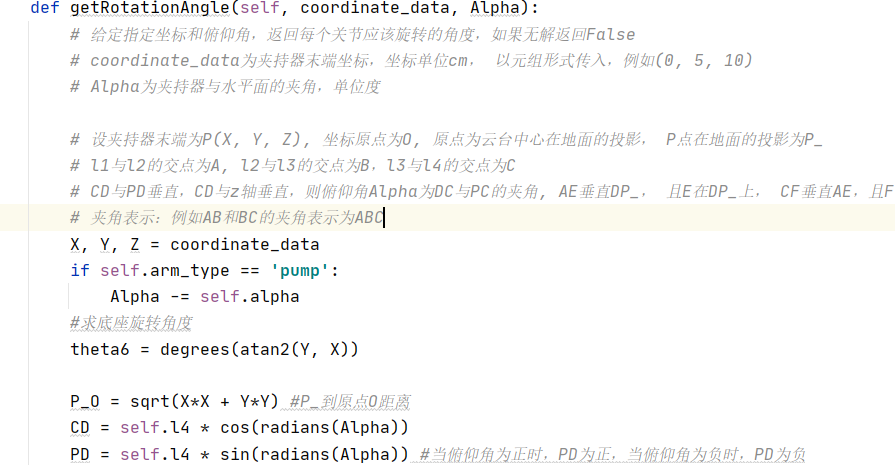
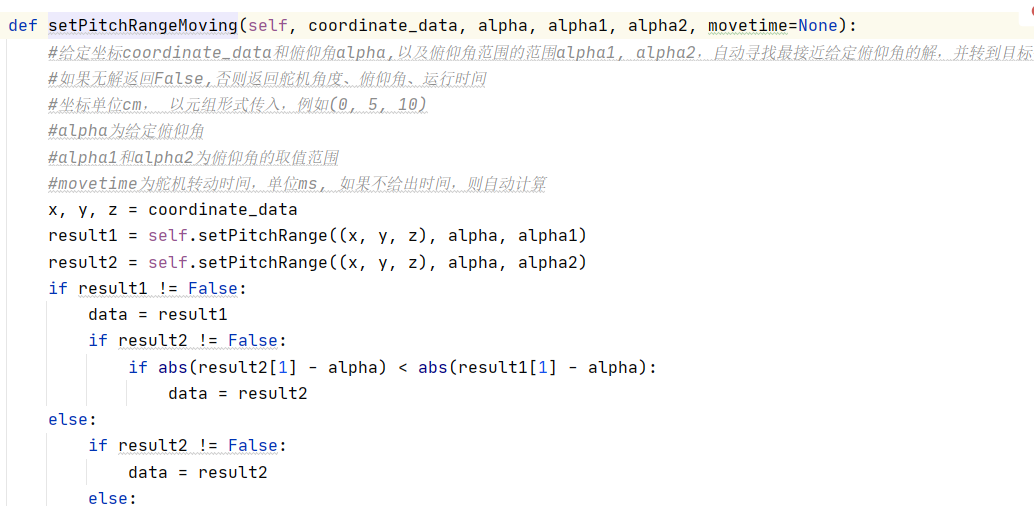
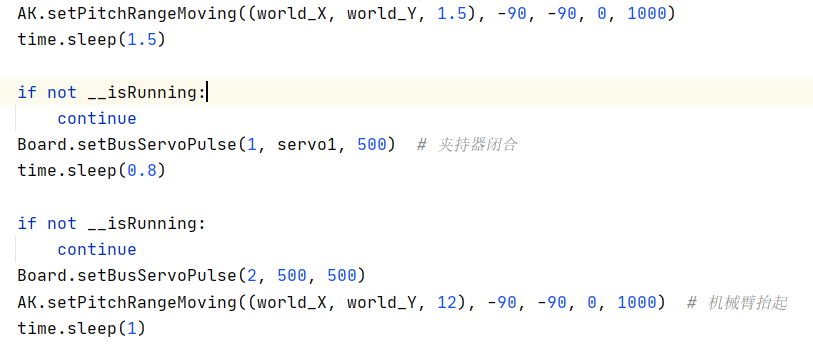


图2.5.3 机械臂（3）



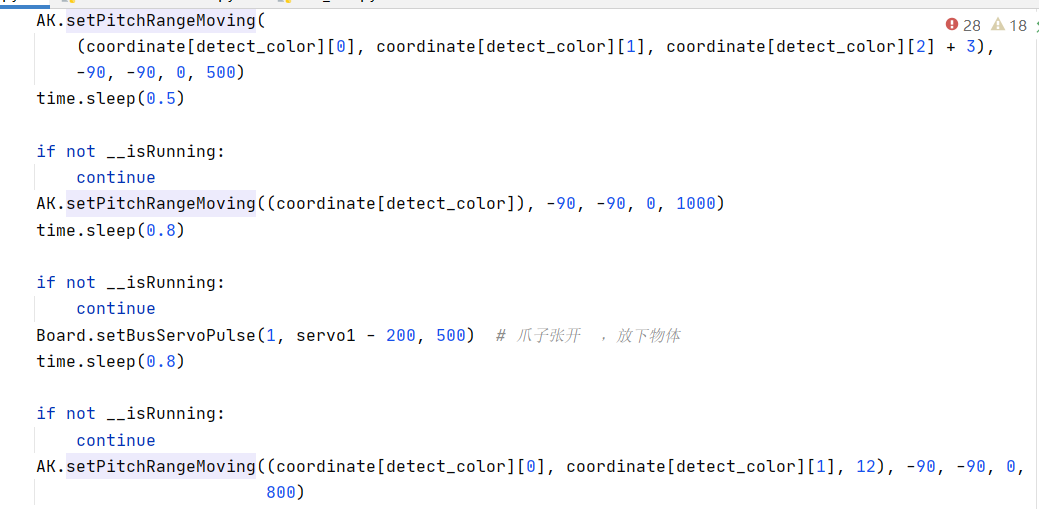
利用夹持器的位置、舵机之间的距离，五子棋坐标等计算能否在可允许移动的范围内找到舵机的移动的俯仰角、计算夹持器需要旋转的角度并且移动过去。

图2.5.4 机械臂（4）



将机械臂移动到五子棋上，然后夹取五子棋，然后将机械臂抬起。

图2.5.5 机械臂（5）



再将机械臂移动到指定位置的上方，将机械臂放下，张开爪子，放下物体，最后还原即可。

其中指定位置是通过文件获得，在电脑计算出放置的坐标后，将其写入文件。然后机械臂移动之前读取文件，然后进行坐标和真实世界的位置的转换。

另外就是机械臂中遇到的一些问题了。

首先开始连接机械臂的时候就发生了一些问题。一开始看老师给的方法没有特别清楚。后来了解到其他同学是采用了外接显示器的方法控制机械臂。经过我的了解，最后我采用的方法是连接热点，然后程序控制。

后来在机械臂控制方面遇到了很多问题。最大的问题就是毫无基础，涉及到了很多的基础方面的知识，而这些都是没有涉及到的。后来的解决方法就是学习他的源代码。

最后的问题就是五子棋的位置。五子棋的夹取高度需要调试，另外由于机械臂的限制，棋盘不能用特别大。我采用的19\*19的棋盘

就太大了，机械臂放不过去。这个问题没办法解决，只能靠近机械臂下棋。

# 后记

后记其实写不写都可以，不过感触很多，还是想写下几句话。其实一开始做的时候，感觉志在必行，做着、做着发现实现起来各个方面还是有难度的。于是开始了各种学习和摸索，每天想的除了大作业就是五子棋要么就是机械臂，头昏脑胀了好几周。其实我的全部任务大概一周之前，6月8号左右全部的算法就完成的差不多了。用了一周的时间进行了后续的机械臂调试和大作业报告的撰写，以及后期的补充实验。感觉收获了很多，重新燃起了对我们专业的热情。

老师有一句话很戳我，不会为了让大家都选他的课，就把课程弄得很水，他一定会认认真真得对待这件事。确实，每个人的心中都会自己的坚持，日后我可能也会成为一名教师，希望能够做一位有坚持、有信仰、有人格魅力的好老师！