人工智能大作业项目报告

——基于人工智能原理的五子棋及其拓展

陈冠旭 朱奕坤 张方杰 吕泓泰

一、 摘要

该项目主要基于 Python 实现了五子棋对弈以及基于其的功能拓展。其充分应用了人工智能原理课程所学习的知识，使用了 α－β 剪枝算法、DQN 算法、机器学习等多种算法原理。该项目最终完成了五子棋的图形化实现，基于搜索算法与机器学习方法的两个五子棋对弈 AI，基于卷积神经网络的手写识别的功能。其所有源代码可在 GITHUB 上由网址 https://github.com/tommychen123/gobang_for_ai 获取，该项目充分体现了学生对于人工智能基础课程所学内容的应用。

二、 关键词

Python、五子棋 AI、人工智能算法、卷积神经网络

三、 项目内容

（1） 五子棋的图形化实现

五子棋的图形化实现主体是基于 pygame 库实现的。首先是棋盘、操作页面、棋子的绘制，使用 pygame 内置库可以比较轻松的实现，其大部分实现储存于源代码的 GameMap.py 文件中，这里展示对于部分棋子绘制代码。

```
def drawChess(self, screen):  # 绘制棋子
    player_one = (255, 251, 240)
    player_two = (88, 87, 86)
    player_color = [player_one, player_two]

    font = pygame.font.SysFont(None, REC_SIZE*2//3)
    for i in range(len(self.steps)):
        x, y = self.steps[i]
        map_x, map_y, width, height = self.getMapUnitRect(x, y)
        pos, radius = (map_x + width//2, map_y + height//2), CHESS_RADIUS
        turn = self.map[y][x]
        if turn == 1:
            op_turn = 2
        else:
            op_turn = 1
        pygame.draw.circle(screen, player_color[turn-1], pos, radius)

        msg_image = font.render(
            str(i), True, player_color[op_turn-1], player_color[turn-1])
        msg_image_rect = msg_image.get_rect()
        msg_image_rect.center = pos
        screen.blit(msg_image, msg_image_rect)
```

图 1-棋子绘制代码（部分）

之后是游戏整体的构建。这里的最基本的游戏规则采用的是无禁手、白先、15*15 的棋盘上横平竖直斜线的任意五颗相同颜色的棋子连成一条线即为该执颜色棋子的棋手获胜的规则。对于该规则的实现方法如下：构建一个 Map 类，用于存储其中的棋子信息和棋手轮次，在落子的时候进行判断，是否合法，同时每下完一步对棋盘上的棋子进行一次遍历检测，是否达到了棋局结束的条件。这里展示判断胜利的部分

代码。

```
def judge_Win(self):
    x = 0
    y = 0
    cnt = 0
    color = self.EMPTY
    # 遍历四个方向
    for d in range(4):
        color = self.EMPTY
        cnt = 0
        # 遍历9颗连续棋子
        for k in range(-4, 5):
            x = self.premove[0] + self.DIR[d][0] * k
            y = self.premove[1] + self.DIR[d][1] * k
            if self.judge_Legal(x, y):
                if self.board[x][y] == self.EMPTY:
                    color = self.EMPTY
                    cnt = 0
                else:
                    if self.board[x][y] == color:
                        cnt += 1
                    else:
                        color = self.board[x][y]
                        cnt = 1
            else:
                if k > 0:
                    break
            if cnt == 5:
                self.winner = color
                return color
    return self.EMPTY
```

图 2-判断胜利代码（部分）

最后是人机交互的设计，要实现“下棋行为”和“功能选择”必须需要设计人机交互界面。这里使用 pygame 的 button 控件去对各种人机行为进行分类统筹、调用不同的函数实现具体的功能。为了达成更好的人机交互效果程序使用了 tkinter 库中的 messagebox 从而弹出提示框来辅助提醒。下面展示一个按钮控件部分的实现代码。

```python
class MultiStartButton(Button):  # 开始按钮（多人游戏）
    def __init__(self, screen, text, x, y):  # 构造函数
        super().__init__(screen, text, x, y, [
            (230, 67, 64), (236, 139, 137)], True)

    def click(self, game):  # 点击，pygame内置方法
        if self.enable:  # 启动游戏并初始化，变换按钮颜色
            game.start()
            game.winner = None
            game.multiple = True
            game.mode = 2
            self.msg_image = self.font.render(
                self.text, True, self.text_color, self.button_color[1])
            self.enable = False
            return True
        return False

    def unclick(self):  # 取消点击
        if not self.enable:
            self.msg_image = self.font.render(
                self.text, True, self.text_color, self.button_color[0])
            self.enable = True
```

图 3-BUTTON 控件代码（部分）

最终程序实现的图形化界面如下图所示。

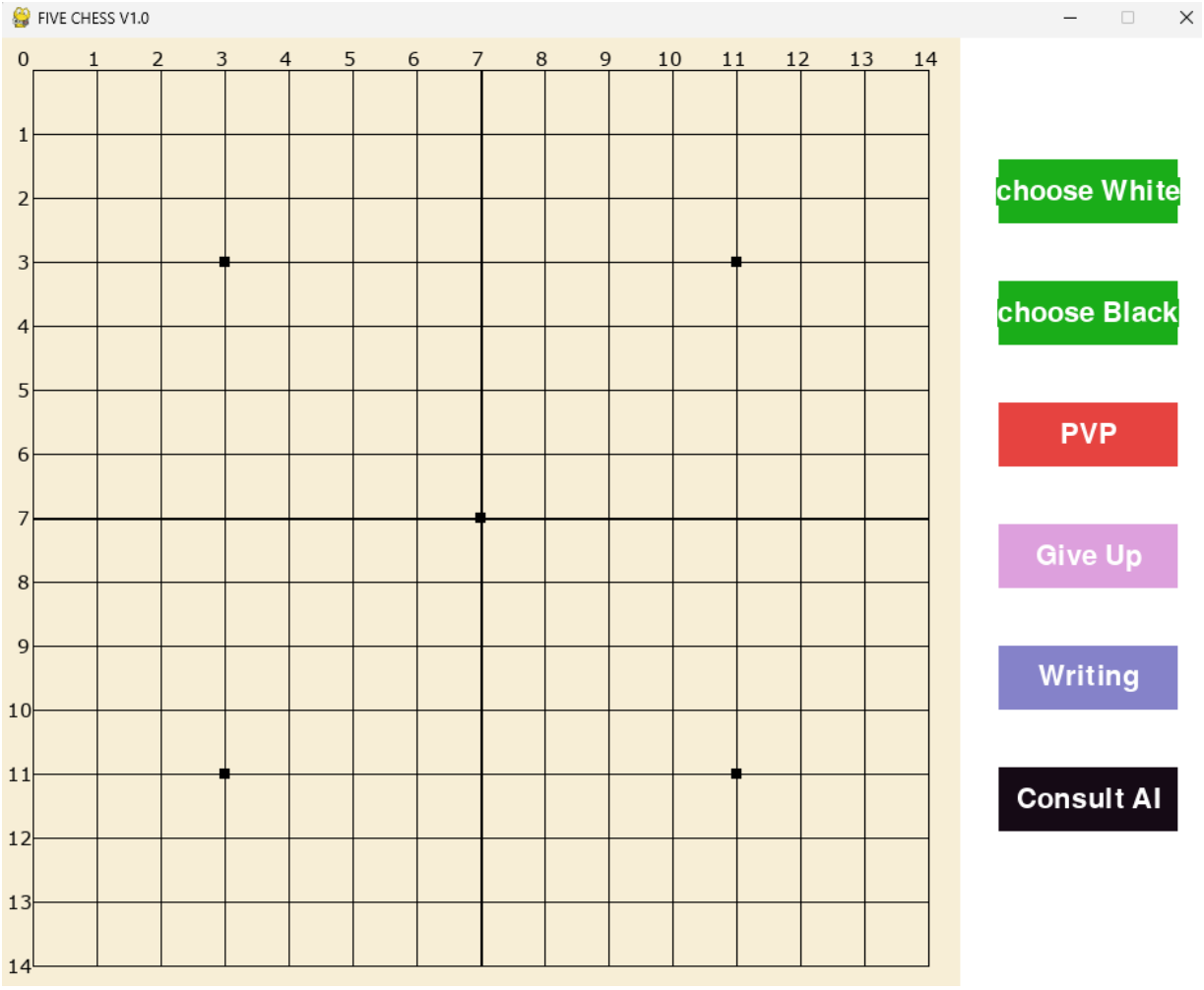


图 4-最终图形化界面

通过按相应的按钮去选择各种预置好的功能，在棋盘上下棋会有落子音效，胜利会有胜利判断等图形功能。具体可以见项目演示的详细演示。

（2） 五子棋 AI 的α－β剪枝实现

五子棋本质上是一个博弈类游戏。那我们可以基于该原理将 AI 和玩家构建一个博弈树。假设 AI 方为 MAX 点，我方则为 MIN 点。如果当层的节点为奇数时那么就为 MAX 层，同样为偶数时就为 MIN 层。当在 MAX 层时，该层的值就应该为下一个 MIN 层中的最大一个的值。当在 MIN 层是，该层的值就应该为它子层 MAX 的最小的一个。通俗的说就是当轮到我方时，我们就应该选择一个最有利于我们的点，预测对方可能下的最有利他方的点(相对我方来说就是最坏的点)。这样反复计算下去就能够得到根节点的最大值，这个点也就是我们最佳下棋点。在计算这个点时可以很明显的看出这是一个不断递归的过程，到达叶子节点时根据相关的计算规则算出该值然后向上一层不断的返回。这样就可以获得一颗博弈树。

而基于构建的博弈树我们继而采用 α－β 剪枝法去实现。其基本思想或算法是，边生成博弈树边计算评估各节点的倒推值，并且根据评估出的倒推值范围，及时停止扩展那些已无必要再扩展的子节点，即相当于剪去了博弈树上的一些分枝，从而节约了机器开销，提高了搜索效率。具体的剪枝方法如下:

对于一个与节点 MIN，若能估计出其倒推值的上确界 β，并且这个 β 值不大于 MIN 的父节点(一定是或节点)的估计倒推值的下确界 a，即 a≥β，则就不必再扩展该 MIN 节点的其余子节点了(因为这些节点的估值对 MIN 父节点的倒推值已无任何影响了)。 这一过程称为 a 剪枝。

对于一个或节点 MAX,若能估计出其倒推值的下确界 a，并且这个 a 值不小于 MAX 的父节点的估计倒推值的上确界 β，即 a≥β，则就不必再扩展该 MAX 节点的其余子节点了(因为这些节点的估值对 MAX 父节点的倒推值已无任何影响了)。 这一过程称为 β 剪枝。

为了便于理解，此处使用了 GitMind 工作软件对剪枝流程进行了流程图绘制，具体过程便不再文字赘述。

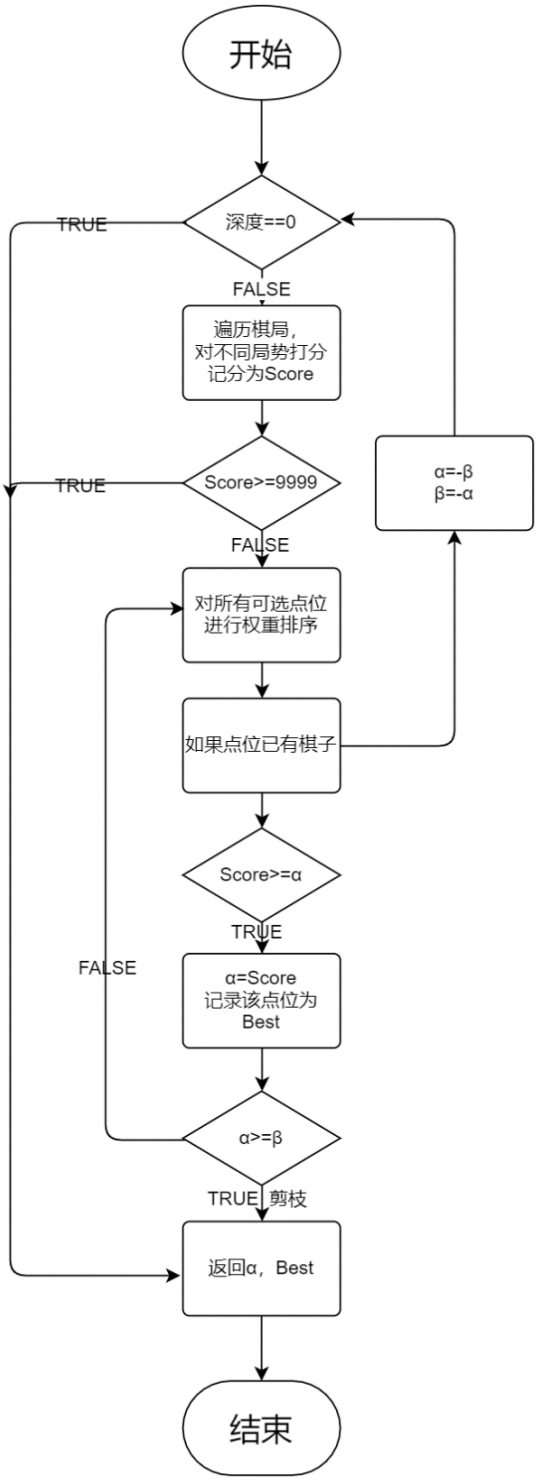


图 5-搜索算法逻辑图

如果想要更多的了解具体代码实现，该部分算法程序在 AlphaBeta.py 文件中有比较详细的叙述，配备有注释，可以方便理解。下面展示部分剪枝代码。

```
def __search(self, board, turn, depth, alpha=SCORE_MIN, beta=SCORE_MAX):
    score = self.evaluate(board, turn)
    if depth <= 0 or abs(score) >= SCORE_FIVE:
        return score

    moves = self.genmove(board, turn)
    bestmove = None
    self.alpha += len(moves)

    # 没有move则返回score
    if len(moves) == 0:
        return score

    for _, x, y in moves:
        board[y][x] = turn

        if turn == MAP_ENTRY_TYPE.MAP_PLAYER_ONE:
            op_turn = MAP_ENTRY_TYPE.MAP_PLAYER_TWO
        else:
            op_turn = MAP_ENTRY_TYPE.MAP_PLAYER_ONE

        score = - self.__search(board, op_turn, depth - 1, -beta, -alpha)

        board[y][x] = 0
        self.belta += 1

        # alpha/beta 剪枝
        if score > alpha:
            alpha = score
            bestmove = (x, y)
            if alpha >= beta:
                break
```

图 6-剪枝算法代码（部分）

（3） 五子棋坐标的输入手写识别

手写识别主要有两个任务点一个是在电脑上使用画板进行手写，第二个是对识别的图片进行数字识别。画板的设计使用的是 PyQt 库的相关控件去实现，其关键在于记录鼠标按住到松下这段过程中的轨迹，并将其描绘，具体实现在 MyMnistWindow.py 之中。下面展示部分画板代码。

```
if len(self.pos_xy) > 1:
    point_start = self.pos_xy[0]
    for pos_tmp in self.pos_xy:
        point_end = pos_tmp

        if point_end == (-1, -1):
            point_start = (-1, -1)
            continue
        if point_start == (-1, -1):
            point_start = point_end
            continue

        painter.drawLine(
            point_start[0], point_start[1], point_end[0], point_end[1])
        point_start = point_end
painter.end()
```

图 7-画板算法代码（部分）

之后是对画上的图片进行截图然后交给模型去识别。这里的模型使用的是基于 CNN 即卷积神经网络设计的图像数字识别模型。简要阐述一下其原理。卷积神经网络是一种多层、前馈型神经网络。从功能上来说，可以分为两个阶段，特征提取阶段和分类识别阶段。特征提取阶段能够自动提取输入数据中的特征作为分类的依据，它由多个特征层堆叠而成，每个特征层又由卷积层和池化层组成。处在前面的特征层捕获图像中局部细节的信息，而后面的特征层能够捕获到图像中更加高层、抽象的信息。再具体而言，其包含卷积核、池化、标准化、感受野等过程。

而在使用 CNN 实现时候，我们可以直接使用 MNIST 的数据集去进行训练。MNIST 数据集是美国国家标准与技术研究院收集整理的大型手写数字数据库,包含 60,000 个示例的训练集以及 10,000 个示例的测试集。其中的图像的尺寸为 28*28。这里我们直接对从网络上找好的 MNIST 训练模型直接运用，省去了模型训练的复杂过程。这里展示部分识别数字的代码。

```
def recognize_img(self, img):
    myimage = img.convert('L')  # 转换成灰度图
    tv = list(myimage.getdata())  # 获取图片像素值
    tva = [(255 - x) * 1.0 / 255.0 for x in tv]  # 转换像素范围到[0 1], 0是纯白 1是纯

    init = tf.global_variables_initializer()
    saver = tf.train.Saver

    with tf.Session() as sess:
        sess.run(init)
        saver = tf.train.import_meta_graph(
            './HandWriting/minst_cnn_model.ckpt.meta')  # 载入模型结构
        saver.restore(sess, './HandWriting/minst_cnn_model.ckpt')  # 载入模型参数

        graph = tf.get_default_graph()  # 加载计算图
        x = graph.get_tensor_by_name("x:0")  # 从模型中读取占位符变量
        keep_prob = graph.get_tensor_by_name("keep_prob:0")
        y_conv = graph.get_tensor_by_name("y_conv:0")  # 关键的一句  从模型中读取占位

        prediction = tf.argmax(y_conv, 1)
        # feed_dict输入数据给placeholder占位符
        predint = prediction.eval(
            feed_dict={x: [tva], keep_prob: 1.0}, session=sess)
        print(predint[0])
    return predint[0]
```

图 8-手写识别代码（部分）

（4） 基于 DQN 的五子棋 AI 实现

在原有简单 AI 的基础上，我们试图去尝试去采用强化学习算法对神经网络模型进行学习。具体而言，使用强化学习 DQN 算法训练神经网络。同样使用卷积神经网络作为估值函数。通过引入随机过程和经验池，可以一定程度上解决问题三中出现的陷入局部最优解的问题。通过使用主网络和目标网络两个网络交替训练的方式，完成网络的训练。而对网络参数的更新，则以棋盘-Q 值数据对对网络进行反向传播与梯度下降，达到训练的效果。

对于网络，执行若干次迭代。每次迭代进行一局棋局的对战，在每一局的对战过程中，将每一步的落子所产生的棋盘及下一状态，作为一个状态节点投入经验池进行保存。当经验池中的数据量达到一定程度后，便在每次落子后抽取 BATCH 组数据，对网络进行训练。而每训练 100 轮次，将主网络参数拷贝至目标网络，进行参数的更新。

将网络参数存储下来，后续通过函数调用，来对当前棋盘的棋局进行评估。不过由于训练模型的复杂，以及 15*15 棋盘的高复杂度，在经过数千次训练后，黑白双方仍未能很好的相互博弈。目前能做到有四子连珠的时候去完成任务等最基础功能。我们相信经过不断的训练，日久天长，该模型的思维能匹

配人类。

下面展示部分创建 Q 值数据代码，这部分代码主要在 DQN_AI.py 之中。

```python
def create_Q(self):
    # 网络权值
    W1 = self.weight_variable([5, 5, 1, 16])
    b1 = self.bias_variable([16])  # 5*5*16
    W2 = self.weight_variable([5*5*16+1, 225])
    b2 = self.bias_variable([1, 225])

    # 输入层
    self.state_input = tf.placeholder("float", [None, self.state_dim])
    self.turn = tf.placeholder("float", [None, 1])

    y0 = tf.reshape(self.state_input, [-1, 15, 15, 1])
    # 第一卷积层
    h1 = tf.nn.relu(self.conv2d(y0, W1) + b1)
    y1 = self.max_pool_3_3(h1)  # 5*5*16

    # 第二全连接层tf.concat([t1, t2], 0)
    h2 = tf.concat([tf.reshape(y1, [-1, 5 * 5 * 16]), self.turn], 1)
    self.Q_value = tf.matmul(h2, W2)+b2
    # 保存权重
    self.Q_weihgts = [W1, b1, W2, b2]
```

图 9-DQN 算法代码（部分）

（5） 代码合作与版本管理

在整体代码编译中我们小组采用了 Git 仓库管理的方法。熟练运用 git 进行四个人合作，对每次提交进行了相关的配合，以下展示我们 Git 部分的提交记录。并在出现错误的时候即使进行了回档。目前该项目已经上传到 GitHub 上 https://github.com/tommychen123/gobang_for_ai 可以通过该链接克隆代码查看。

```
commit 24ef5cbe32e07885ff7ccc338ae165d4ac55a621 (HEAD -> main, origin/main, origin/HEAD)
Author: “pluto” <“1341460360@qq.com”>
Date:   Sat Dec 16 16:46:32 2023 +0800

    规范了注释和代码1.0

commit f629f54e22678c56f09b0b6af7df229752690158
Author: tommychen123 <cgx2003@qq.com>
Date:   Fri Dec 15 21:14:41 2023 +0800

    新增手写识别和AI咨询

commit f7031a4213e5b44cd6802e300d205a117185b1c6
Merge: d33f554 22482e4
Author: tommychen123 <cgx2003@qq.com>
Date:   Fri Dec 15 21:12:49 2023 +0800

    Merge branch 'main' of github.com:tommychen123/gobang_for_ai

commit d33f554f7202ea31dd22ab63ba04cb0c91038d2f
Author: tommychen123 <cgx2003@qq.com>
Date:   Fri Dec 15 21:09:49 2023 +0800

    新增DQN和手写识别
```

图 10-项目的 Git log

我们对整个项目进行了模块化分工，将 DQN_AI.py,MyMnistWindow.py,GameMap.py 等上述介绍的功

能性代码进行分工协作，最后再统一整理，极大地提高了工作效率，充分地发挥了每个成员的作用。

四、 项目演示

（1） 项目环境配置

这是本次项目最终测试使用的电脑配置。

设备名称　CGX-ALIENWARE-X17R2

处理器　12th Gen Intel(R) Core(TM) i7-12700H　2.30 GHz

机带 RAM　32.0 GB (31.7 GB 可用)

系统类型　64 位操作系统，基于 x64 的处理器

通过测试显卡　GTX1660Ti/RTX3060/RTX3070/RTX3080Ti

所使用的 python 及其依赖库环境。

Python 版本 Python 3.10.7　具体库可以见 requirements.txt

numpy==1.26.2

Pillow==10.1.0

pygame==2.5.2

PyQt5==5.15.10

PyQt5_sip==12.13.0

tensorflow==2.10.0

tensorflow_gpu==2.10.0

tensorflow_intel==2.10.0

值得一提的是本项目使用的 tensorflow V1 版本的相关函数，而 2.10 版本的 tensorflow 会有许多不兼容现象出现。所以需要加入在引用的时候 import tensorflow.compat.v1 as tf 以及 tf.compat.v1.disable_eager_execution()这两句语句去采用 tensorflow1.x 版本的相关函数。

另外模型训练时候的 GPU，请根据 tensorflow 版本，采取合理的 Cuda 版本以及 Cudnn 版本，挑选合适的显卡运行，此处不再赘述。

（2） 五子棋界面与操作演示

接下来对操作进行相应的演示。在克隆的项目中运行 python main.py 进入程序页面。选择执白或者执黑开始在棋盘中下棋，此时会进行阿尔法贝塔剪枝的 AI 进行对局。每下一步棋后 AI 会自动下下一步棋。我们可以调整 AlphaBeta.py 中的搜索深度去改变 AI 的强度。默认搜索深度为 2，可以调整为 4.PVP 模式时，不会有 AI 对弈，黑白都需要玩家去点击操控。游戏运行时会有相关的背景音乐和落子音效，当棋局终止时后有提示框和文字提醒。

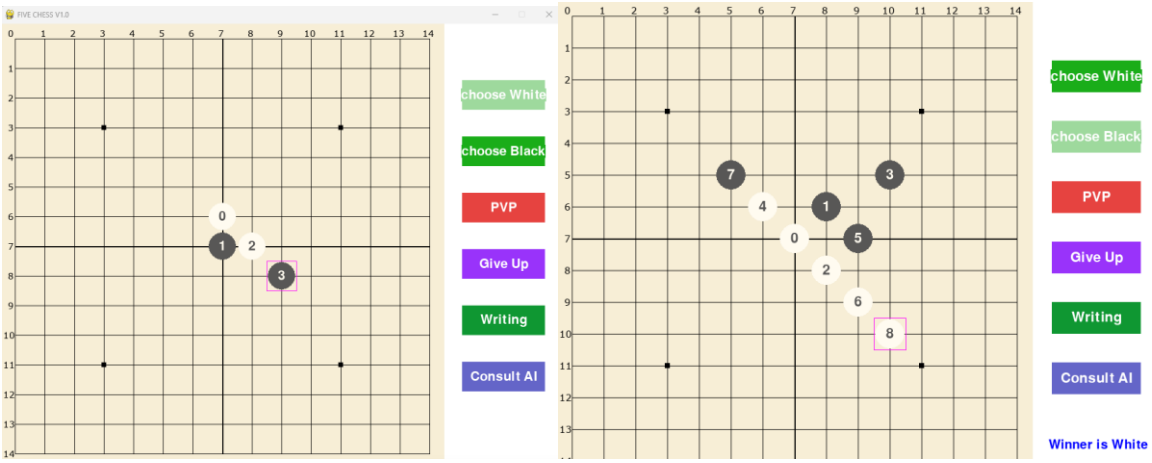


图 11-界面展示

当我们选用 Writing 模块的时候，会弹出如下界面。

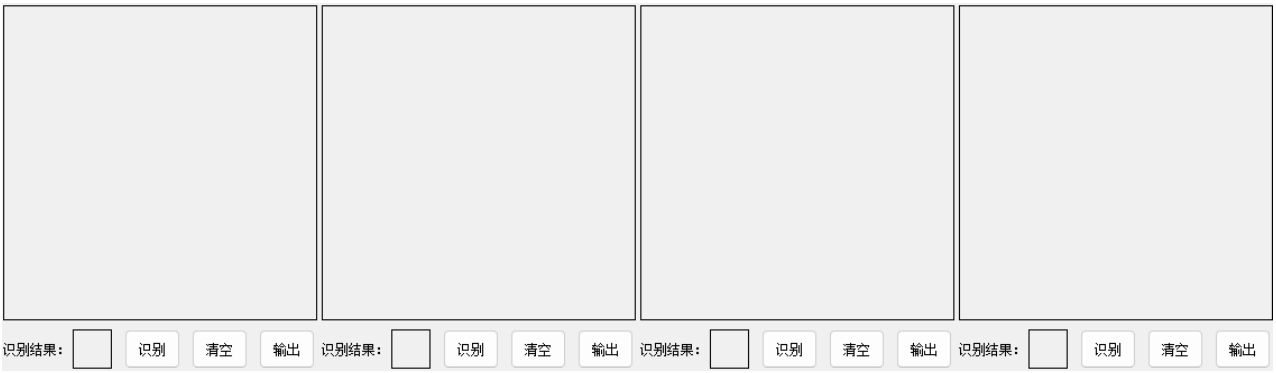


图 12-手写板初始态

对每个框内填写相应的坐标的数字信息（分别对应的是横坐标的第一、二位，纵坐标的第一、二位）。填写后点击识别，即可分析出相应的手写数字，点击输出即可将该位数字输出到棋盘之上，等待进一步的操作。需要注意的是如果不填写，则默认该位数字是零。同时输出的时候会对位置坐标的合法性进行相关的检查，如果有误，则会报错。

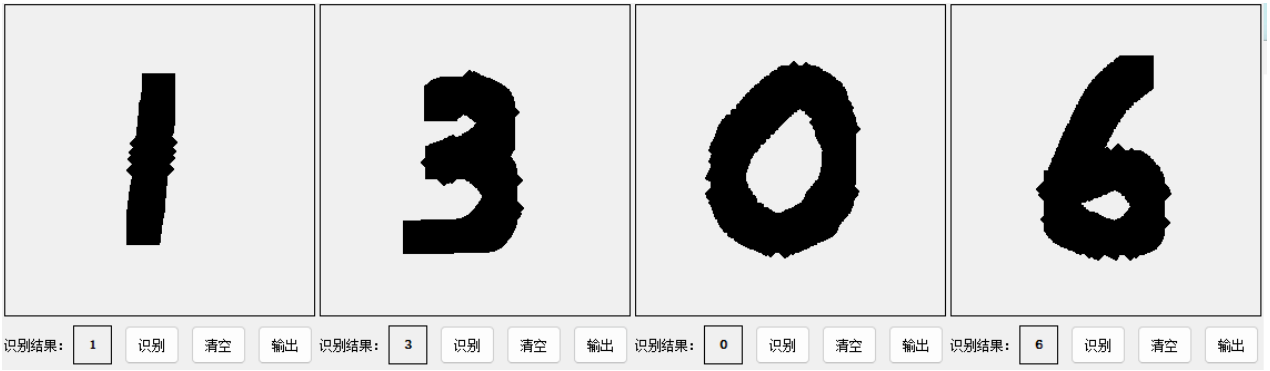


图 13-手写板识别态

点击 Consult AI 模块时，会基于当前场面状况给与一个 AI 的反馈提示。在 main.py 中，将 use_AI_model 置为 True，即可使用有 DQN 训练的五子棋 AI 给出的场况提示。否则是由 EasyAI.py 中的 AI 给出提示。这里由于目前 DQNAI 的模型尚不健全，于是默认是置为 False 的。

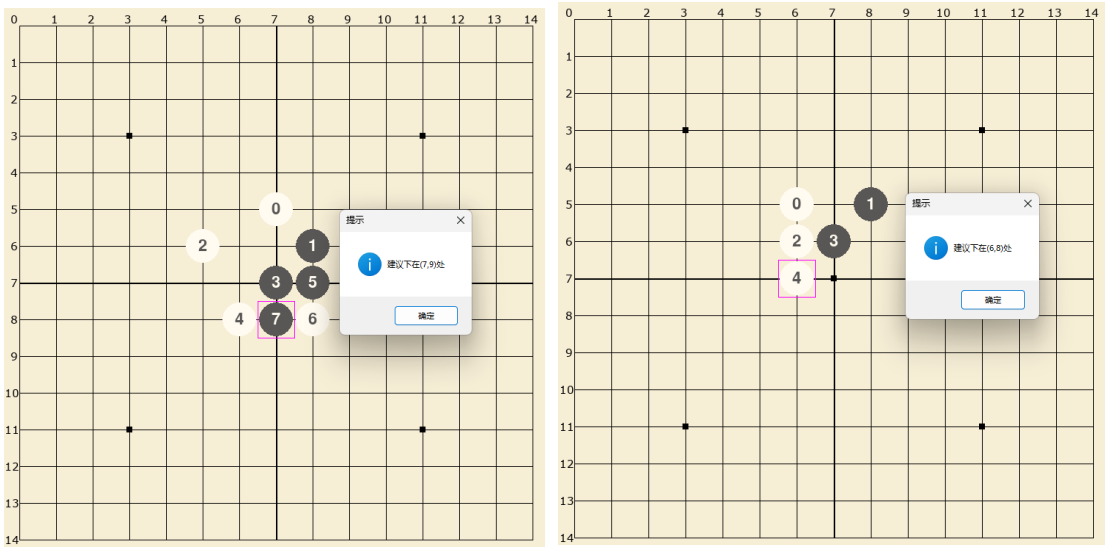


图 14-AI 提示

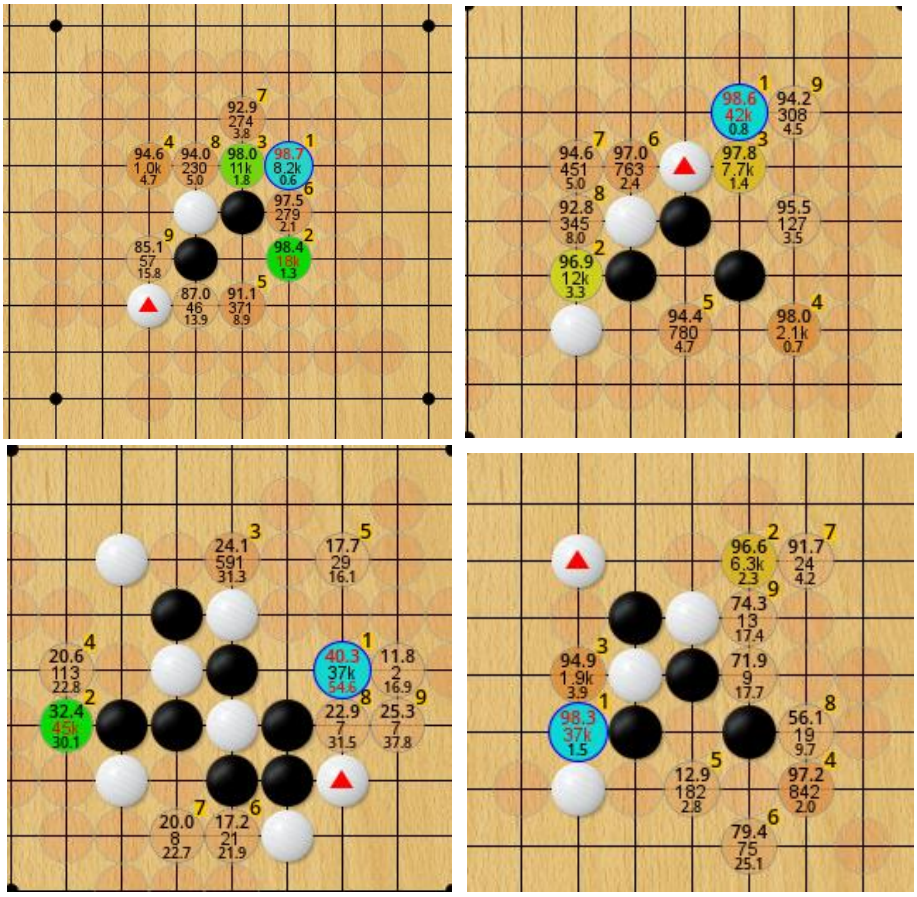
总体游戏进程默认是循环的，所有功能按键都能反复使用。最后点击关闭按钮即可退出 Pygame 界

面，即退出该项目。

（3） 五子棋 AI 对比总结

这里我们选取由 Katago 围棋 AI 修改而成的目前实力极强的 Gomoku AI。其项目代码见 https://github.com/hzyhhzy/KataGo/tree/gomoku。使用其无禁 freestyle：15 路无禁(15x15 freestyle)引擎来和我们搜索深度为 4 的 α－β 剪枝算法构造的 AI 进行对弈分析。

以以下棋局为例子。



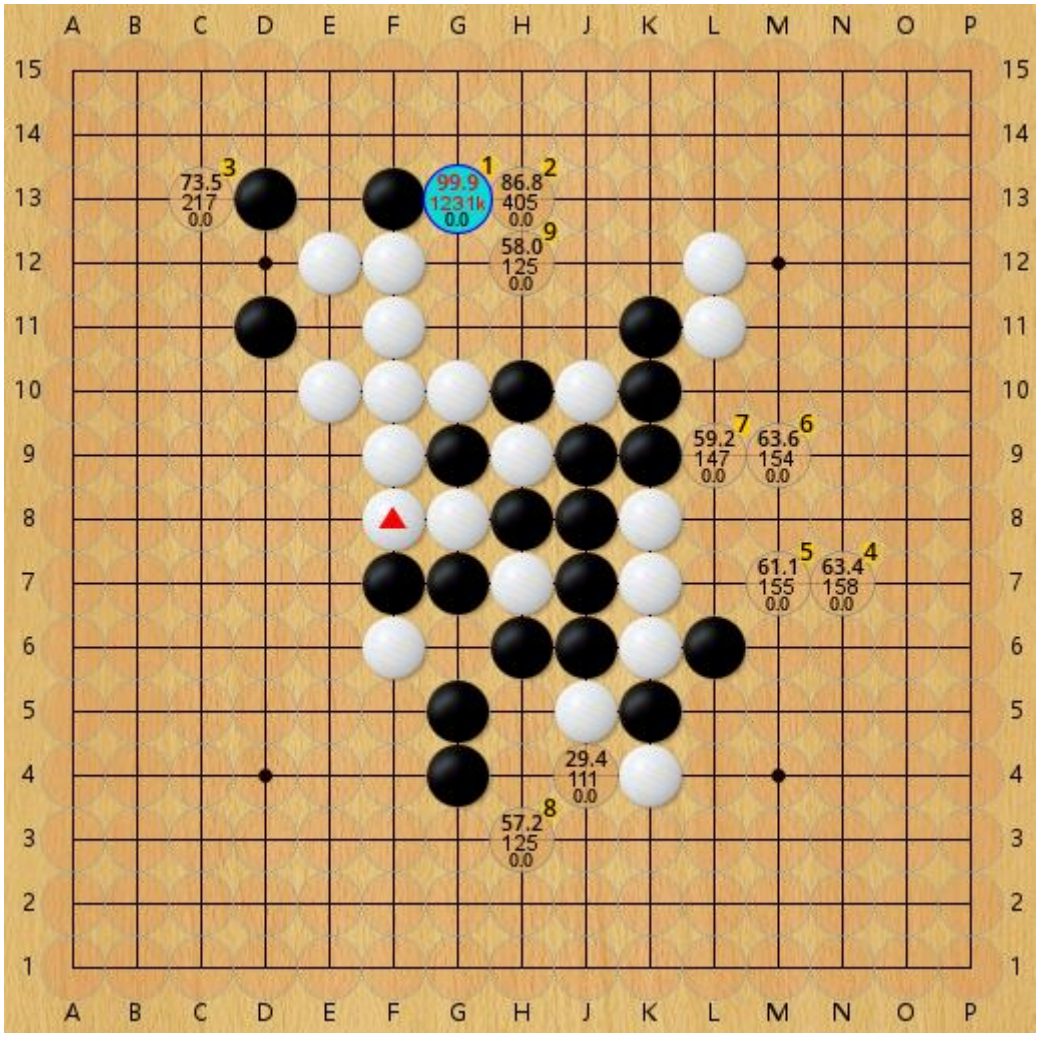


图 15-AI 对弈棋局展示

通过分析得知，我们的搜索算法在前期能基本上做到每一步的一选、二选，在棋局复杂的过程中，每一步逐渐降低成五、六选，并逐渐被更强的 AI 搬回局面然后打败。总体而言，在面临更复杂的情况下，搜索算法还做不到最优。但面对针对通常情况已经非常强大了。

五、 项目总结

（1） 项目存在的不足与改进之处

目前来看我们的项目已经充分应用了人工智能原理所学习到的知识，并且成功完成了既定目标。不过总体看来还有几处不足之处。一是 AI 模型生成的不稳定性，由于 15*15 的变化量太多，目前训练的模型没法达到黑白子博弈的基础程度，应当加入一些算法预训练，使其能够更快的进化。二是 α－β 搜索算法在高搜索深度时运算速度会大幅度降低，受算力限制大，如果能加上 GPU 进行并行运算能提高速度。三是手写识别功能运用的时候还较为复杂，模块嫁接的时候兼容性有待提升。需要后续针对该棋盘做进一步的适配化处理。

道阻且长，让我们不断努力，一步一步攻克难关。

（2） 项目分工

成员姓名	主要完成任务	工作量占比
陈冠旭	协调分工、代码模块组合、功能拓展、项目报告撰写	30%
张方杰	手写识别模块设计、模型收集、代码整理	25%

朱奕坤	基础图形模块设计、阿尔法贝塔剪枝算法设计、调试程序	25%
吕泓泰	DQN 算法模型收集、PPT 制作、答辩提纲撰写	20%

（3） 心得体会

关于这次的人工智能基础大作业，我们首先面临的就是选择一个合适的项目。在选择时，我们考虑了自身的兴趣、专业背景以及当前的热门领域。最终选择了智能 AI 五子棋这一个既能激发我们的兴趣，又有着适当的难易程度和可行性的主题。在完成这个项目的过程中，我们学会了如何用 Python 来实现五子棋 UI 界面、神经网络和一些智能算法，也学会了如何搜索和使用一些常用的数据集，如 MNIST、CIFAR、 IMDB 等等。综上而言，这次的大作业虽然花费了不少的心血和时间，但是当最终的成品完成时，我们都深切地体会到了人工智能的先进性和研究的快乐，感到非常的兴奋和满足。我相信，在今后的学习和工作中，这些技能都能极大地帮助我们，这次经历将成为我们坚强的后盾和宝贵的财富。

六、 参考文献

[1] 雨寒 sgg. Tensorflow（3）：创建画板，实时在线手写体识别--终极篇（PyQt5）[EB/OL]. https://blog.csdn.net/u011389706/article/details/81460820,2018-08-06.

[2] 徐志平. Python 五子棋 AI 实现(1):界面实现[EB/OL]. https://blog.csdn.net/marble_xu/article/details/90340389?spm=1001.2014.3001.5502

[3] 王文敏.人工智能原理[M].北京:高等教育出版社,出版年:2019