华南理工大学
South China University of Technology

《人工智能》 课 程 论 文

(2020-2021 学年第 1 学期)

基于深度神经网络的五子棋引擎设计与现实

学生姓名：常睿

提交日期：　　年　　月　　日　　学生签名：

学　　号	201830680022	座位编号	
学　　院	软件学院	专业班级	软件工程中澳班
课程名称	人工智能	任课教师	应伟勤
教师评语：			
本论文成绩评定：　_____分			

基于深度神经网络的五子棋引擎设计与现实

常睿

摘要：针对如何实现一个具有人工智能的五子棋引擎，结合采用了自我博弈学习，蒙特卡洛树搜索，以及策略价值神经网络三种核心技术，设计开发了一个使用 python 环境的 9*9 的五子棋引擎，实现了下一步走子的概率预测，以及目前棋盘形势的评分功能，经过 10 几个小时的 300 次训练，与完全使用蒙特卡洛树搜索算法的五子棋走子策略进行对比评估，得出一个几乎可以说是百战百胜的引擎，效果较好，这次设计主要是将棋盘的大小进行了升级（9*9），随着训练次数的增加，蒙特卡罗树也会增加模拟的次数，但是整体来说，我的模型也还占优，不足之处在于当蒙克卡罗树模拟次数增加时，我们的模型暂时并未完全胜出，一部分原因取决于电脑的算力以及训练次数不够，另外在 loss 函数的设计，以及更新参数的方法还可进一步调优。参数例如蒙特卡洛树的模拟增加次数也可进行最优化（我设定为 300）。

关键词：自我学习；纯蒙特卡洛树搜索；策略价值网络

2018 年 12 月，Deep Mind 推出的 Alpha Zero 诞生一周年之际，《自然》杂志以其为封面发出了关于它的完整论文。Alpha Zero 是一个极其强大的人工智能，是强化学习的产物，精通三种棋类游戏，而且在顶尖水平，而且其不需要数据集作为支撑，便是一个巨大的创新，围棋，日本将军棋等十分复杂，因次我把对象面向五子棋。

为了实现一个 AI 的五子棋引擎，我使用的主要强化学习方法步骤与逻辑如下：首先通过自我博弈获得训练的数据，模型的好坏由策略决策网络得出的价值结果与真实值之间的损失函数来刻画，当队列中储存的数据量大于我们使用的 mini-batch 梯度下降方法所需的最小的批量数目时，我们便可以进行策略价值网络的参数更新，最终一更新模型，以下时具体的实现细节。

一、强化学习与马尔可夫决策过程

在进行模型的设计与实现之前必须明确强化学习的几个概念。

强化学习采用的是一边获得样例边学习的方式，在获得样例之后更新自己的模型，利用当前的模型来指导下一步的形行动，下一步行动的结果获得回报，此回报在来印象行动，目的是获得最大的回报。而其中又有两个很关键的概念对我们的模型设计有很关键的作用，这两个概念是，探索和开发，探索是指选择之前未使用过的行动，从而探索更多的可能，开发则是选择已经执行过的行动，对模型进行完善。强化学习作为一个序列决策问题，它需要连续选择一些行为，从这些行为完成后得到最大的收益作为最好的结果。它在没有任何标注的情况下告诉算法应该怎么做的情况下，通过先尝试做出一些行为——然后得到一个结果，通过判断这个结果是对还是错来对之前的行为进行反馈。

在强化学习中，马尔科夫决策过程（Markov decision process, MDP）是对完全可观测的环境进行描述的，也就是说观测到的状态内容完整地决定了决策的需要的特征。几乎所有的强化学习问题都可以转化为 MDP[3].马尔可夫决策过程通过以下几种方法去决定价值函数。穷举法：把所有可能的路径和状况都试一遍计算，直接比较最后哪条路的总价值最大。动态规划（DP）：将一个问题拆成几个子问题，分别求解这些子问题，反向推断出大问题的解。即要求最大的价值，那么根据递推关系，根据上一循环得到的价值函数来更新当前循环。但是它需要知道具体环境的转换模型，计算出实际的价值函数。相比穷举法，动态规划即考虑到了所有可能，但不完全走完。蒙特卡洛（MC）：采样计算。即通过对采样的多个完整的回合（如玩多次游戏直到游戏结束），在回合结束后来完成参数的计算，求平均收获的期望并对状态对动作的重复出现进行计算，最后再进行更新[4]。

在这个框架下我们便可以逐步确定我们的五子棋引擎，首先对于状态搜索，刻

画价值函数（这里我们的价值便是博弈的胜利的概率）我们在这里不进行启发式的设计，采用最基础的蒙特卡洛算法，然后下一步根据上一步作出决定，得到最终的最优策略。我在此次实践中，专注于修改棋盘的规格，以及调整了超参数每次与完全蒙特卡罗搜索算法博弈成功后，蒙特卡罗增加的模拟次数为 300。

二、自我博弈

在我的模型中，我选择从零开始训练，一开始由于没有类似于棋谱之类的数据集，所以我们使用自我博弈的技术来收集数据，这也避免了需要棋谱的大量数据，更加直接快捷。自我博弈显然需要使所进行的博弈面临各种各样的情况，保证期盼的不同的都有落子，为了保持这样的多样性我们必须收集大量的数据，蒙特卡罗搜索不失为一种好的推演每个位置的算法，为了求解问题，首先建立一个概率模型或随机过程，使它的参数或数字特征等于问题的解：然后通过对模型或过程的观察或抽样试验来计算这些参数或数字特征，最后给出所求解的近似值。解的精确度用估计值的标准误差来表示。蒙特卡洛方法的主要理论基础是概率统计理论，主要手段是随机抽样[1]。每一手的走子，都由蒙特卡罗搜索计算出每个位置的概率，并且更具我们的赢的规则去选择出最终能使之成为赢家的位置[2]。

通过这种方法，我们结合强化学习的传统思想，状态，动作，概率，回报（在这里我们指的是是否取胜）来构建一个存储数据的结构：我使用了 policy_value_fn()这个函数来进行记录，这个输入棋盘的状态，然后会输出得出关于动作，概率，得分（位于-1，1 之间）的元组列表，根据规则选出最终的走子。我们不妨把一个棋盘的状态用 S（t）来表示，那么我们收集到的数据便是

S_1，S_2，S_3…S_n, 以便我们在策略价值神经网络中进行训练，而这里也正是模型的奇妙之处，经过训练后的模型，又作为最新的最优的模型[2]去帮助蒙特卡洛树作出决策，一开始蒙特卡罗搜索作出的决定是均匀的，慢慢地随着训练次数增加，最终会达到收敛。

三、策略价值网络

策略价值网络，一个典型的深度学习网络，为了减少训练时间，模型深度并不是太深，最深深度仅有 7 层，首先是三层公共的卷积神经网络层，接着分为两部分，左边的是走子概率输出部分，128*9*9 先经过一个的卷积神经网络层，再经过一个 4*9*9 的全连接层，最后输出走子的概率 P，而向右则是局面胜出率部分，首先是一个 128*9*9 卷积神经网络，在经过两个全连接层，第一个是 2*9*9 的输入，经过激活函数 Relu 之后到达全连接层 2，在经过 tanh 激活函数得到最终的局面胜利概率。输入也不是很大的矩阵，是足够描述我们棋盘状态的 4*9*9 矩阵。具体的流程见 Figure1。

我们的棋盘大小是 9*9 所以只需要 9*9 的二维矩阵便可以表述棋盘，4 个平面中，2 个分别是对手的棋子位置与玩家的棋子位置，1 为此位置有落子，0 为空，因为下棋往往需要记录对手上一步所落子的位置，所以第三个平面则是对手的最后一步落子状态，只有落子的那一个位置为 1，第四个平面则表示当前的玩家是不是先下棋的人，如果是先手则全为 1，不是则全为 0。

在公共的卷积神经网络的部分，每一份分别使用 32，64，和 128 个 3*3 的卷积核，然后经过 Relu 激活函数分成走子概率分支与局面胜出概率，在走子概率分支部分使用 4 个 1*1 的卷积核，而在局面胜出概率的分支先使用 2*1*1 的卷积核，最终连接两个全连接层的到胜出概率。

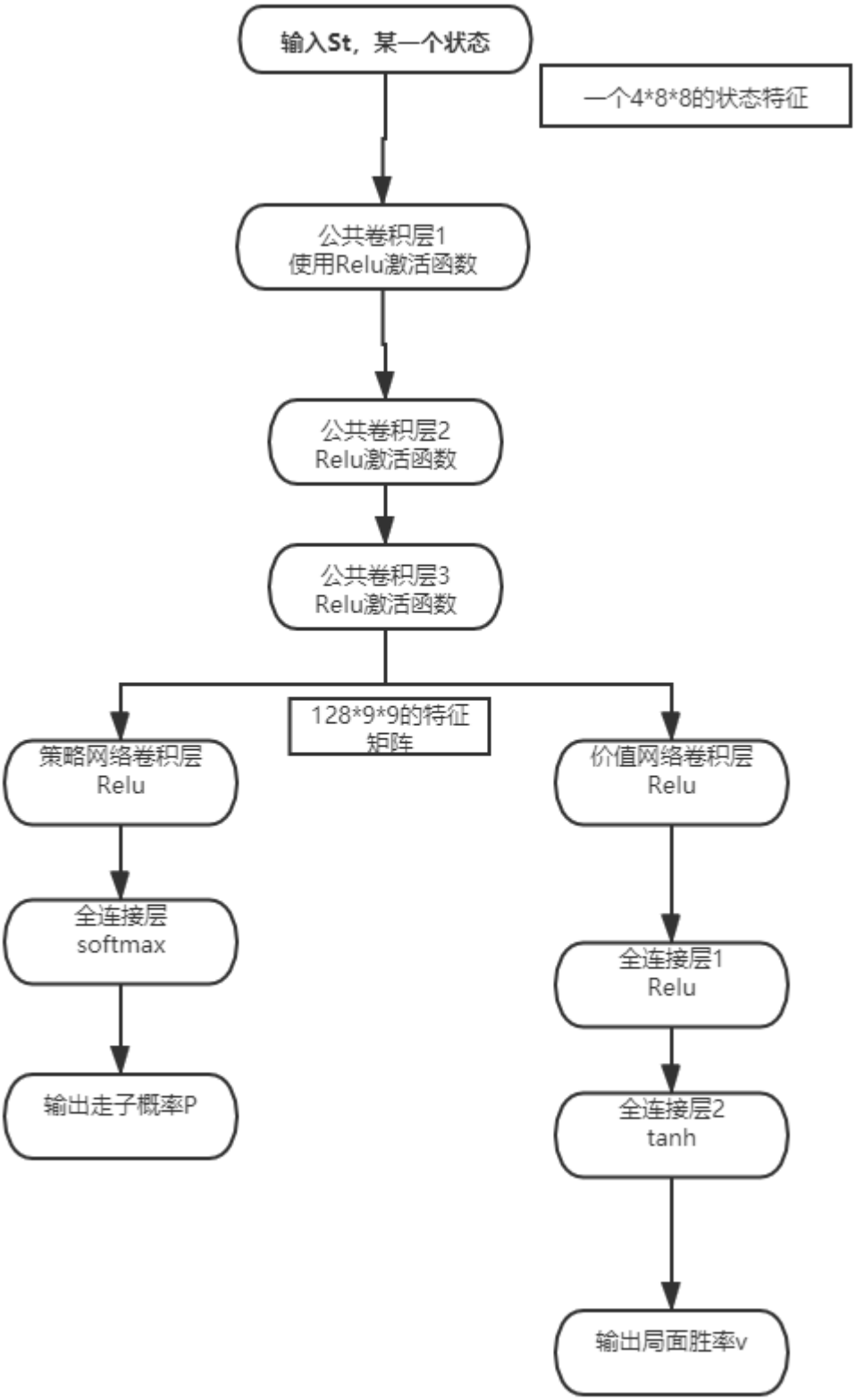

Figure 1 **策略深度神经网络模式图**

整个网络自上往下看就是一个典型的前馈神经网络，我们采用 mini-batch 梯度下降的方法来对神经网络进行训练，Loss 函数则是为了使策略神经网络的输出值与使用

蒙特卡罗搜索树确定的走子概率的差异性最小化，并且使局面的胜出概率与对弈的真实结果差值最小。通过不断的迭代，此实验中我进行了 300 次迭代，损失函数如下：

$$\mathrm{L} = (z - v)^2 - \pi^T \log P + c \| \theta \|^2$$

此公式 中的最后一项为正则化参数，为了防止过拟合问题的出现[2]。

实验结果：

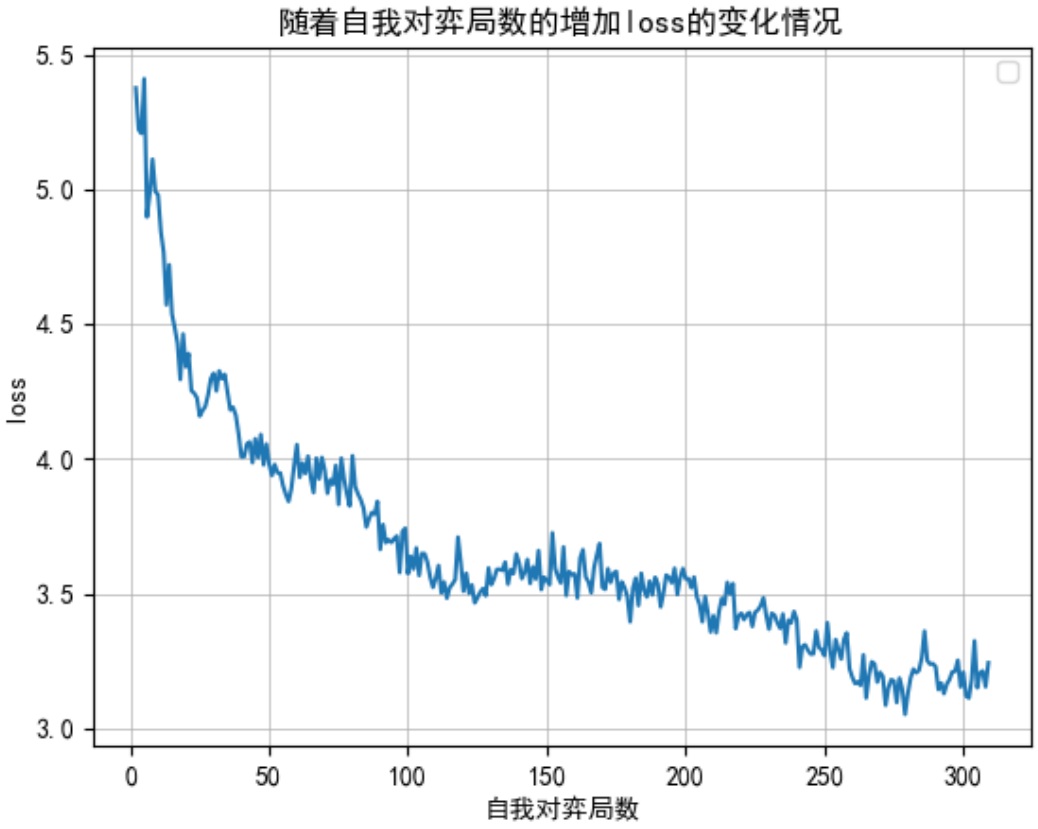

Figure 2 **损失函数随训练次数的变化**

如图二便是我训练的随着自我博弈次数增加的 loss 的变化情况图，从中可以看出 275 次之后 loss 函数已经开始逐渐趋近稳定，在 3 附近一直震荡，预计在进行 3000 次的训练后，此模型的 loss 函数应该收敛于 2 左右，所以目前的模型整体效果是不错的，损失函数随着自我对弈局数的增加而减小，而且逐渐开始收敛。这表明我们的策略神经网

络的走子概率分支已经逐渐毕竟真实的蒙特卡罗树的走子概率，已经可以刻画五子棋引擎的走子策略智能了。

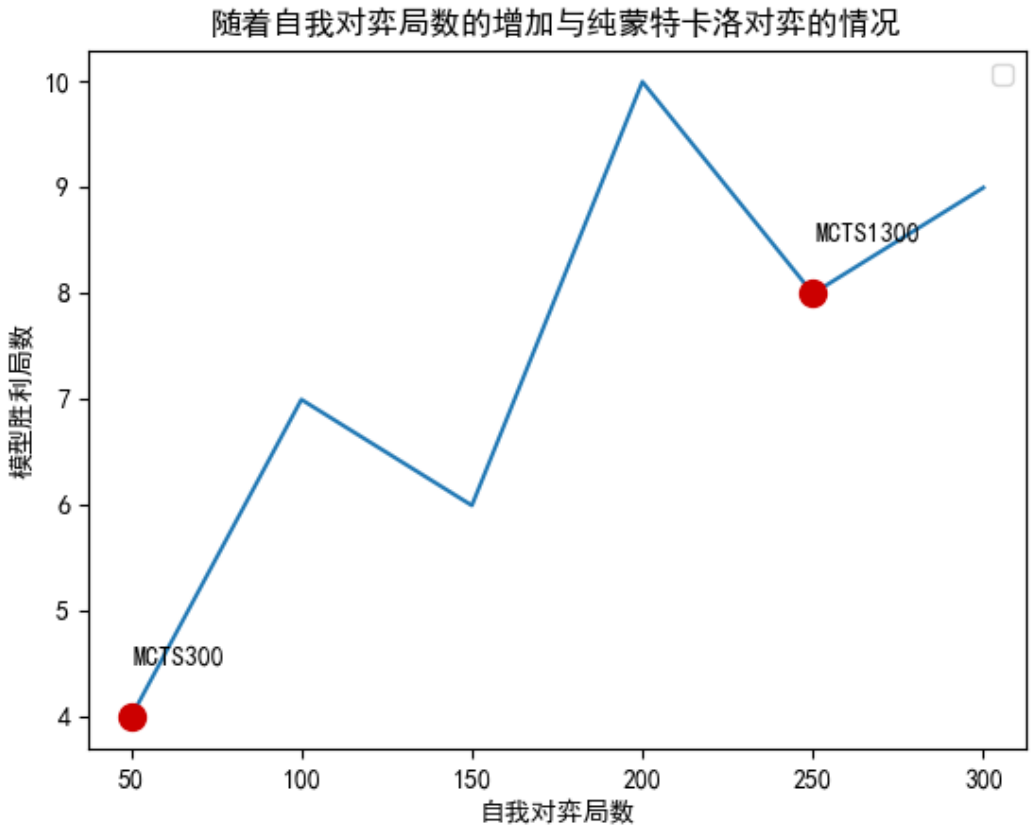

Figure 3 **随着自我对弈局数的增加模型与蒙特卡罗树博弈结果**

如图三，我们可以看出，随着自我对弈局数的增加模型与蒙特卡罗树的对弈结果是已经最高达到 10：0，但最终整体在 8-9 之间，随着对弈次数的增加，整体的对弈结果毫无疑问一定是 10：0 的。显然 300 次的训练次数是不够的，如果能进行到 3000 次的训练结果一定很优美，是一条 y=10 的常数函数，或者说一直在 10 左右向右边延伸。

结论：经过三百次的训练也花了近一天的时间，得到的模型效果还好能令人满意，整体水平上赢的概率 9：1，效果不错，是一个挺智能的 AI 五子棋引擎，达到了我预期的标准，这次工作主要在于对棋盘的大小进行了修改，从最初的 8*8 的棋盘变成现在的 9*9 的棋盘训练的速度变慢是我明显感觉到的，毕竟虽然棋盘规模仅仅变大了一点点，但是

其蒙特卡罗树的搜索分支，以及各种走子的组合确实指数级别的上升，难免会导致我的训练很慢，另外我还对蒙特卡罗树每 50 次对战，如果输了增加 1000 次蒙特卡罗搜索进行了修改，我改成了 300 次，这意味着蒙特卡罗搜索的次数变少了，可能覆盖的情况变少了，但是有利于我模型的训练，不然模型可能相同的时间仅仅只能迭代 100 次。这是我参数的一个调整，但整体对训练效果的影响不大，所以这个超参数或许是可以有一个更加合适的数值的。遇到被战胜对比标准的 Alpha Zero,我发现其实还是有很多地方就可以进行修改的，比如说，我的卷积核的参数，以及我的网络的深度，或者是刻画棋盘的特征矩阵的数目可以多很多，这样肯定会更加准确，而且随着蒙特卡罗树搜索的次数上升，我相信也不会对我的模型有太大影响，当然如果有更简单的方法，例如启发式的搜索，进行 α 或 β 剪枝，去替换蒙特卡罗树搜索，说不定可以取得更加好的训练效率。DeepMind 的科学家已经在围棋和象棋上进行了与 α 和 β 剪枝的模型[5]比较，以后我会尝试更多的实验，希望也可以让我的五子棋引擎也能尝试剪纸。以更多的超参数组合去训练，在更好的算力支持下争取可以在一个 loss 函数的最低设定值下训练知道让其自动停止。在整个设计的过程中，我研读了 Alpha Zero 已经构建好的框架，进而学习强化学习的相关知识，对其进行的符合我的情况的简化与延伸，其主要是针对围棋，国际象棋，日本将军棋等，而我将它延伸到五子棋对战当中，完成此次实践的两个关键部分还是蒙特卡洛搜索以及策略神经网络。通过这次实践，我对强化学习的框架更加深入地了解了，每个过程真的就是一个马尔可夫过程，每次做决定都是为了使回报函数最大，这也很符合我们人做事情的思维,而且我感觉我对半监督学习的认识也加强了,因为 Alpha Zero 并不需要自己去寻找数据集，而是通过自己的对抗作为数据源，没有标记出类似于棋谱的东西，直至最后的胜出有一个评判标准，我觉得与 GAN 有一点点的相似。整个过程之中，有一段时间训练的结果一直在震荡，当我看见在第 250 次的时候，输出了一个 New best policy 的时候，心里非常激动，这就是机器学习的魅力吧。

参考文献

[1]https://baike.baidu.com/item/%E8%92%99%E7%89%B9%E5%8D%A1%E7%BD%97%E6%B3%95/1225057?fr=aladdin

[2] David Silver1 *, Julian Schrittwieser1 *, Karen Simonyan1 *, Ioannis Antonoglou1 , Aja Huang1 , Arthur Guez1 , Thomas Hubert1 , Lucas Baker1 , Matthew Lai1 , Adrian Bolton1 , Yutian Chen1 , Timothy Lillicrap1 , Fan Hui1 , Laurent Sifre1 , George van den Driessche1 , Thore Graepel1 & Demis Hassabis1 Mastering the game of Go without human knowledge ，NATURE， 19 October 2017 | VOL 550 | NATURE | 355

[3] https://zhuanlan.zhihu.com/p/28084942

[4] https://blog.csdn.net/qq_39388410/article/details/88795124

[5]David Silver,1* Thomas Hubert,1* Julian Schrittwieser,1* Ioannis Antonoglou,1 Matthew Lai,1 Arthur Guez,1 Marc Lanctot,1 Laurent Sifre,1 Dharshan Kumaran,1 Thore Graepel,1 Timothy Lillicrap,1 Karen Simonyan,1 Demis Hassabis, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, DeepMind, 6 Pancras Square, London N1C 4AG