设计研究与应用

# 基于 MCTS 和卷积神经网络的五子棋策略研究

欧俊臣，沙 玲，杨淞文

（上海工程技术大学 机械与汽车工程学院，上海 201620）

**摘 要：** 随着 AlphaGo 的诞生，人机对弈和人工智能再次成为研究热点。传统的 MCTS（蒙特卡洛树搜索）虽然在迭代搜索方面已有良好的成效，但由于五子棋搜索空间较大，算法极易陷入局部最优化问题，且耗时严重。我们用 MCTS 和卷积神经网络上设计的策略系统，让其与 MCTS 进行训练（self-play），使五子棋的策略系统能在一定时间内对自身进行升级，然后又回来继续训练自身，这样得到的五子棋策略系统不仅比传统的 MCTS 更具有即时性，棋力也更强。

**关键词：** 五子棋；卷积神经网络；MCTS

**中图分类号：** TP183 **文献标识码：** A **DOI：** 10.3969/j.issn.1003-6970.2020.04.034

**本文著录格式：** 欧俊臣，沙玲，杨淞文. 基于 MCTS 和卷积神经网络的五子棋策略研究[J]. 软件，2020，41（04）：160–164

## Research on Gobang Strategy Based on MCTS and Convolutional Neural Networks

OU Jun-chen, SHA Ling, YANG Song-wen

(*College of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai* 201620, *China*)

**【Abstract】:** With the birth of AlphaGo, man-machine game and artificial intelligence have once again become research hotspots. Although the traditional MCTS (Monte Carlo Tree Search) has a good effect in iterative search, because the search space of Gomoku is large, the algorithm is easy to fall into the local optimization problem, and it takes time. We use MCTS and the convolutional neural network to design a strategy system to let it train with the MCTS (self-play), so that the Gomoku's strategy system can upgrade itself within a certain period of time, and then come back to continue training itself. The Gobang strategy system is not only more immediacy than the traditional MCTS, but also much stronger.

**【Key words】:** Gobang; Convolutional neural networks; MCTS

## 0 引言

随着计算机的发展，基于“人工智能”而生的产物越来越多，在人机博弈领域内更是如此。无论是 IBM 设计的“Deep Blue”还是 Google 开发的 AlphaGo Zero，都代表了人工智能在人机博弈所能达到的最高水平。五子棋，又称为五子连珠，英文名 Gobang 或者 Gomoku，发源于古代中国，后流传到日本，如今在世界范围内广受人们的喜爱。五子棋的规则很简单，对局双方分别持黑白棋子，持黑棋者为先手、白棋者为后手，双方先后落子，谁先在横、竖、斜任一方向形成五颗棋子相连，便取得比赛的胜利[1]。

五子棋中游戏中，由于棋盘点数有限，假设对局双方棋力相当，当持黑棋者以某些定式开局，理想情况下双方无限制地对局，胜利总是出现在执先手的一方，故在职业比赛中常常会有禁手规则，即禁止持黑棋者以一定定式开局[2]。本论文环境下不考虑禁手规则。

## 1 背景介绍

### 1.1 蒙特卡洛树搜索（Monte-Carlo Tree Search，MCTS）

MCTS 是一种通过在决策空间中获取随机样本并根据结果构建搜索树来在给定域中查找最优决策的方法。它已经对人工智能（AI）方法产生了深远

**作者简介：** 欧俊臣(1994–)，男，研究生，主要研究方向：机器视觉；沙玲(1970–)，女，副教授，主要研究方向：机械CAD/CAM技术及应用，机器视觉；杨淞文(1994–)，男，研究生，主要研究方向：机器视觉。

的影响，这些方法可以表示为连续决策树，特别是游戏和规划问题。在搜索空间狭小的情况下存在盲目搜索。盲目搜索的意思就是，以当前局势为根节点，按照博弈树展开的形式，直到遍历完整个搜索空间。这样的遍历方式有两种，如图 1 所示，分别是深度优先搜索（Deepth-first search，DFS）和广度优先搜索（Breadth-first search，BFS）[3-4]。

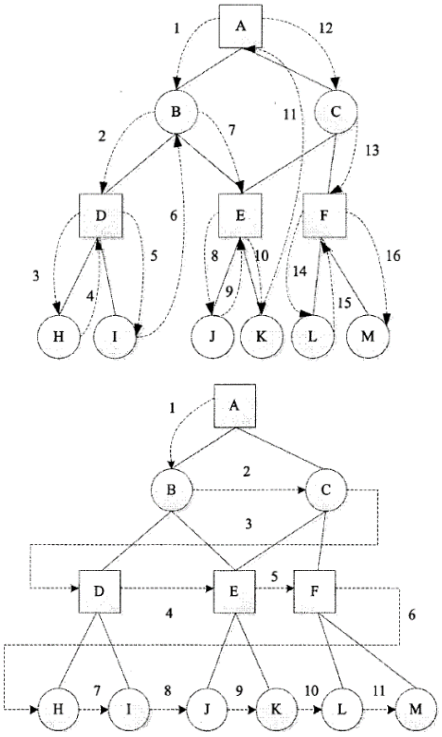



**图 1　DFS 和 BFS 搜索示意图**
**Fig.1　DFS and BFS search diagram**

MCTS 的主要概念是搜索，即沿着博弈树向下的一组遍历过程。单次遍历的路径会从根节点（当前博弈状态）延伸到没有完全展开的节点，未完全展开的节点表示其子节点至少有一个未访问到。遇到未完全展开的节点时，它的一个未访问子节点将会作为单次模拟的根节点，随后模拟的结果将会反向传播回当前树的根节点并更新博弈树的节点统计数据。一旦搜索受限于时间或计算力而终止，下一步行动将基于收集到的统计数据进行决策[5]。MCTS 分为四个步骤：

（1）选择（Selection）：在初始状态下，棋盘状态为空，以当前状态 为根节点随机选择一个未被展开过的节点进行访问。

（2）扩展（Expansion）在此过程中，将一个新的子节点添加到树中，并将其添加到在选择过程中最优到达的节点。

（3）模拟（Simulation）在这个过程中，通过选择动作或策略进行模拟，直到达到结果或预定义的状态。

（4）反向传播（Backpropagation）在确定新添加节点的值之后，必须更新剩余的树。因此，执行反向传播过程，从新节点反向传播到根节点。在此过程中，每个节点中存储的模拟数量将增加。此外，如果新节点的模拟结果是 win，那么 win 的数量也会增加[6]。MCTS 流程图如图 2 所示。

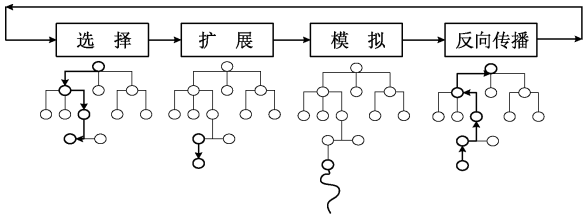



**图 2　MCTS 搜索流程图**
**Fig.2　MCTS search flowchart**

### 1.2　卷积神经网络（Convolutional Neural Network，CNN）

卷积神经网络是近年发展起来的，并引起广泛重视的一种高效识别方法，20 世纪 60 年代，Hubel 和 Wiesel 在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络。现在，CNN 已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用[7]。

## 2　训练框架搭建

我们本次使用的硬件为联想 Thinkpad Edge E431，采用 Intel 酷睿 i5 3210M 处理器，CPU 主频为 2.5 GHz。软件环境为 Python3.7.2 和 TensorFlow1.13。由于 CPU 算力有限，我们以 8x8 的棋盘作为研究。我们首先以 MCTS 为基础设计五子棋的搜索算法，再利用该算法与 CNN 的算法进行对弈，将对弈结果反馈给模型，使该模型能够实时更新最新数据，不断提升模型的棋力。以下将该模型简称为 self-play 模型，如图 3 所示。

在 self-play 中，相应的 $s$ 为当前状态，$z$ 是 self-play 的结果，在描述时二者都基于当前游戏者视角进行，一般通过两个二值矩阵来对当前两个游戏者棋子的位置进行描述。不过在实际的对局中，

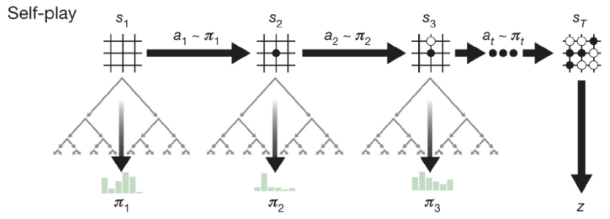



**图 3　Self-play 过程示意图**
**Fig.3　Self-play process diagram**

一局中每一个中的 $z$ 需要在对局结束后才可确定出，如判断发现最后胜者是 $s$ 局面对应的当前游戏者，则 $z$=1，而若败者是，则 $z$=–1，在平局情况下，$z$=0。

我们首先将棋盘信息定义为矩阵 T，分别用 0、1、–1 来填充。其中 0 代表空位，即未有落子的点位，令 1 代表黑棋落子点，–1 代表白棋落子点。再将矩阵 T 当作 CNN 的训练数据。

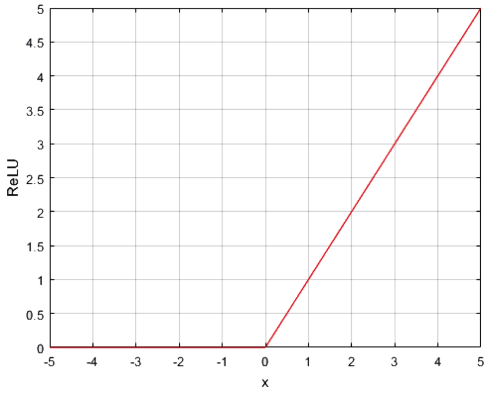
利用 TensorFlow 的 API 函数：tf.layers.conv2d 创建每一个二维卷积层，将输入进行卷积来输出一个 tensor。首先定义一个策略网络类（class PolicyValueNet），在该类下我们可以自由定义 PolicyValueNet 的各种属性。我们首先将矩阵 T 定义为输入层，然后定义三个公共的全卷积网络，其 filters 参数分别为 32、64、128，kernal_size 为 3×3，padding=“same”，data_format=“channels_last”，激活函数为 ReLU 函数，其他参数设置为默认值。ReLU 函数图像如图 4(a)所示。

接着以上述第三个二维卷积层为输入层定义一个激活网络，其参数为：filters=4，kernal_size=[3,3]，padding=“same”，data_format=“channels_last”，激活函数同样为 ReLU 激活，其他参数设置为默认值。接着进一步分成 policy 和 value 两个输出。在 policy 端，先通过 4 个滤波器进行降维，接着通过 tf.layers.dense ()定义全连接层，filters=2，units=board_height * board_width，activation=tf.nn.log_softmax，使用 softmax 非线性函数直接输出棋盘上每个位置的落子概率。其中参数 units 表示输出的维度大小，改变 inputs 的最后一维。log_softmax 激活函数为：

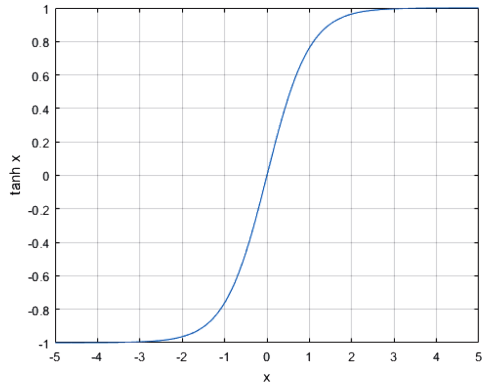
$$\log\_soft\max = \log its - \log(reduce\_sum(\exp(\log its), axis)) \quad (2)$$

其中，logits 表示一个非空的 Tensor，且该 tensor 必须是下列类型之一：half，float32，float64；axis 表示将执行 softmax 的维度，默认值为–1，表示最后一个维度。

在 value 这一端再定义评估网络（Evaluation Networks）。同样利用 tf.layers.conv2d()函数来构建。同样以上述三个公共二维卷积网络为输入，filters=2，kernal_size= [1,1]，padding=“same”，data_format=“channels_last”，activation 为 ReLU 激活函数，其他参数设置为默认值。然后再用 tf.layers.dense ()命令定义两个全连接层，其中前者的输入为以上的卷积层，units = 64，ReLU 为激活函数；后者的输入为第一个全连接层，units=1，激活函数为 tanh，函数图像如图 4(b)所示。在此基础上输出的为一个评估分数，可通过其反映出当前状态，具体表达式如下。




(a)




(b)

**图 4　ReLU 和 tanh 函数图像**
**Fig.4　ReLU and tanh function images**

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (3)$$

根据前面的论述可知，策略价值网络在处理过程中输入为当前局面描述 $s$，当前评分 $\pi$ 则为输出，通过自学习中采集的 $(s,\pi,z)$ 数据来对这种网络进行训练。具体分析以上的训练示意图可知，在此训练过程中，需要控制这种网络的输出概率 $p$ 和 MCTS 的概率 $\pi$ 趋于一致，使得对应的评分结果和真实的

对局结果 $z$ 尽可能的接近，从而实现相应的处理目标[8]。进行优化分析可知，这种操作也就是在 self-play 数据集中通过迭代操作而使损失函数最小：

$$l=(z-v)^2-\pi^T\log P+c\|\theta\|^2 \quad (4)$$

其中第三项主要作用是避免过拟合，为尽可能的降低损失函数，在训练过程中需要不断的使得其值减小。

## 3 数据分析

我们在 8x8 的棋盘上与 MCTS 算法进行五子棋对弈，其每一下一步所进行的搜索信息如图 5 所示。MCTS 会对计算机每一个落子点位所进行的搜索，其中包括每一个点位的概率和对应的胜率。从图中可知，总共进行了 4701 次搜索，最大搜索深度为 41 层，最终得出的坐标为[6,6]。正因为 MCTS 搜索算法需要从当前局面开始往下遍历一定深度，所以耗时时间长。

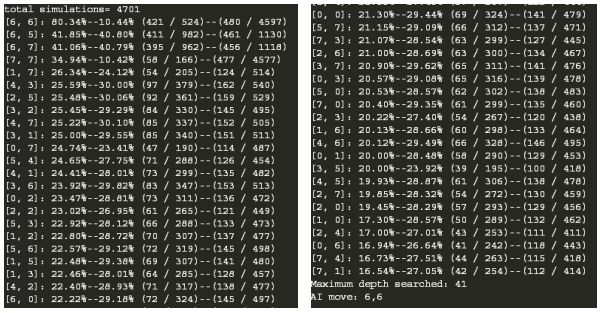



**图 5 MCTS 搜索点位信息图**
**Fig.5 MCTS search point infographic**

当对 MCTS&CNN 的模型进行训练时，相应的函数值和 self-play 局数的相关性如图 6 所示，在此实验过程中开展了 3000 局对局，此函数的值也最终降低到 2.2。

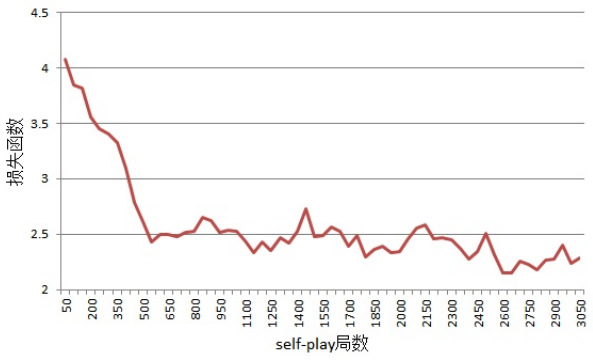



**图 6 损失函数**
**Fig.6 Loss function**

对此模型而言，self-play 数据可基于最新模型确定出，且进行一定更新操作，虽然表面上看基于最优模型确定出的 self-play 数据可满足要求，有较高的收敛性，对提高结果的准确性有重要的意义，本文进行对比分析发现对 6×6 棋盘的 4 子棋而言，通过这种模型进行更新得到的 self-play 数据进行训练，五百局后所得模型就可有效的满足应用要求，对比分析发现基于最优模型对应的 self-play 数据进行训练，在同样的条件下需要 1500 局才可满足要求[9]。

在训练过程中，除了观察到损失函数在慢慢减小，我们一般还会关注策略价值网络输出的策略（输出的落子概率分布）的熵（entropy）的变化情况。正常情况下，最开始的时候，策略网络基本上是均匀的随机输出落子的概率，因此熵会比较大。随着训练过程的慢慢推进，策略网络会慢慢学会在不同的局面下哪些位置应该有更大的落子概率，也就是说落子概率的分布不再均匀，会有比较强的偏向，这样熵就会变小。也正是由于策略网络输出概率的偏向，才能帮助 MCTS 在搜索过程中能够在更有潜力的位置进行更多的模拟，从而在比较少的模拟次数下达到比较好的性能。图 7 展示的是同一次训练过程中观察到的策略网络输出策略的熵的变化情况[10]。可以看到，当训练到 500 局左右时，熵已经降到了 2，但熵并不稳定，在 2 左右来回波动，直到 2300 局之后熵才稳定了下来，一直到 3050 局训练结束。

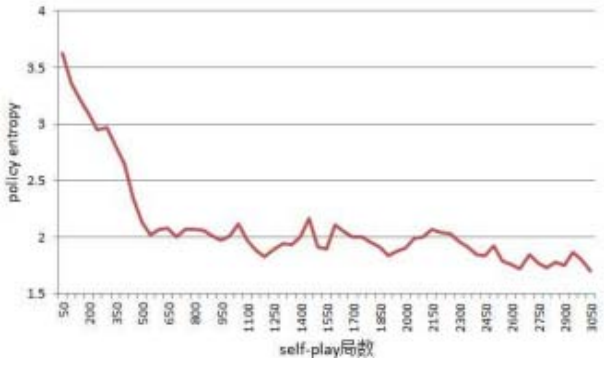



**图 7 熵的变化曲线图**
**Fig.7 Change graph of entropy**

传统的 MCTS 算法需要在每一局面往下搜索很多层，之后还需要将每一个子树的得分往上回溯更新，所以耗时严重，且容易陷入局部最优化的局面。图 8 为一局纯 MCTS 算法的五子棋耗时曲线图，可以看到 MCTS 每一步搜索所用平均时间为 7 s 左右，波动最大接近 1.5 s。图 9 为 self-paly300 局对弈所

用时间，每个点表示一局比赛所用平均时间（平均时间：己方一局对弈总耗时和落子次数的比值）其中红色曲线代表 MCTS 耗时，蓝色曲线表示 MCTS 和卷积神经网络耗时，可以看出后者极大缩短了计算机每一步落子“思考”的时间，且稳定在 1 s 附近。

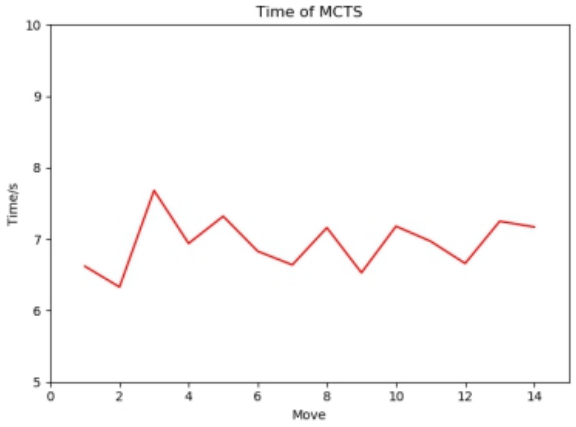



**图 8　一局中 MCTS 搜索时间曲线**
**Fig.8　MCTS search time curve in one round**

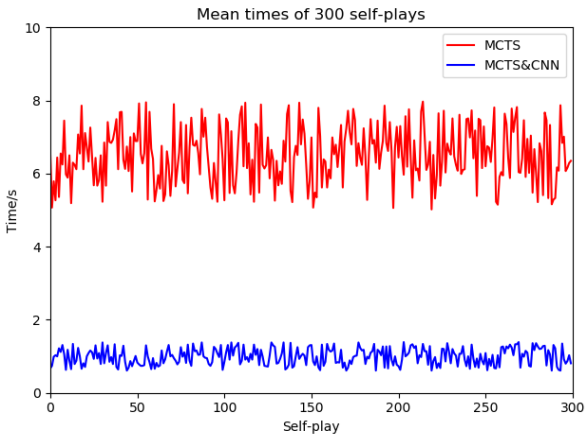



**图 9　300 局 self-play 的时间曲线**
**Fig.9　300 rounds of self-play time curve**

## 4　结果与总结

传统的 MCTS 搜算算法耗时长、且效果一般，五子棋棋力较弱。运用 MCTS 和卷积神经网络训练出的策略价值网络不仅完美的克服了以上问题，且根据验证结果表明建立的这种模型棋力较强，对真实玩家表现出很高的挑战性。这种基于传统搜索算法与神经网络结合的方式，不仅能集成传统算法的优势，更能发挥出人工智能的魅力，使得我们在普通 PC 机上也能进行相关模型的训练。

## 参考文献

[1]　杨向南. 基于卷积神经网络和嵌套网络的目标跟踪算法研究[D]. 华侨大学, 2016.
[2]　高春生. 计算机围棋中落子预测与死活问题的研究[D]. 昆明理工大学, 2017.
[3]　牛恺泽, 邓鑫. 五子棋人工智能研究与实践[J]. 数字通信世界, 2019(01): 32-33.
[4]　于文波. 基于蒙特卡洛树搜索的计算机围棋博弈研究[D]. 大连海事大学, 2015.
[5]　Gaymann Audrey, Montomoli Francesco. Deep Neural Network and Monte Carlo Tree Search applied to Fluid-Structure Topology Optimization.[J]. Scientific reports, 2019, 9(1).
[6]　Teresa Neto, Miguel Constantino, Isabel Martins, João Pedro Pedroso. A multi-objective Monte Carlo tree search for forest harvest scheduling[J]. European Journal of Operational Research, 2019.
[7]　张加佳. 非完备信息机器博弈中风险及对手模型的研究[D]. 哈尔滨工业大学, 2014.
[8]　吴天栋. 非完备信息机器博弈算法及对手模型的研究[D]. 武汉理工大学, 2018.
[9]　郑健磊, 匡芳君. 基于极小极大值搜索和Alpha Beta剪枝算法的五子棋智能博弈算法研究与实现[J]. 温州大学学报(自然科学版), 2019, 40(03): 53-62.
[10]　宋万洋. 基于 α - β 剪枝树算法的安卓五子棋程序设计与实现[J]. 现代信息科技, 2019, 3(11): 92-93+97.