

AlphaZero

1. 概述

国际象棋是人工智能历史上研究最广泛的领域之一。最强大的智能程序需要结合复杂的搜索技术、特定的领域知识以及人工评估等方法，同时还需要人类专家数十年的改进。AlphaGo [1] 以及 AlphaGo Zero [2] 通过自我博弈和强化学习的方式，在围棋领域便实现了对人类玩家的超越。在本文中，将重点介绍 AlphaZero [3]，并尝试将其方法应用于众多具有挑战性的领域中。AlphaZero 不依赖于除游戏规则之外的任何先验知识，从随机策略出发收集数据。在蒙特卡罗树搜索（MCTS）和强化学习技术的协同助力下，经过仅24小时的训练，AlphaZero 便能在国际象棋、将棋（日本象棋）以及围棋等游戏中达到超越人类的水平。

2. 研究背景与相关工作

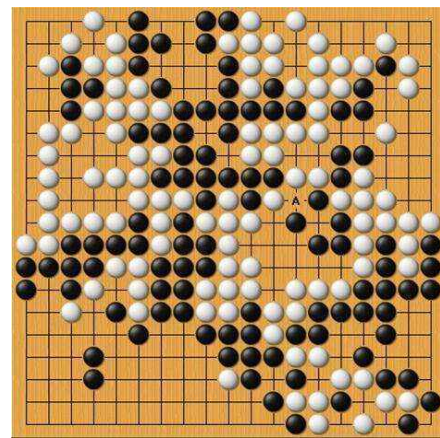
在计算机科学领域，对国际象棋的研究可以追溯至该学科的起源。巴贝奇、图灵、香农和冯诺依曼等科学家致力于设计硬件、算法和理论，旨在分析棋局策略。随后，国际象棋成为一代人工智能研究者面临的重要挑战，最终催生出能以超人水平执行任务的高性能计算机国际象棋程序。然而，这些程序的适用范围仅限于特定领域，若要将其推广至其他问题领域，则需要人类付出大量努力。

人工智能一直以来的目标是创建可以从第一性原理（First Principles）[4] 中自行学习的程序。2017年，AlphaGo Zero 算法运用深度卷积神经网络表示围棋知识，仅通过自我对弈的强化学习进行训练，在围棋游戏中实现了超越人类的表现。本文中，作者将一种类似但完全通用的算法 AlphaZero 应用于国际象棋、将棋以及围棋等游戏。在不依赖于除游戏规则之外的任何先验知识的前提下，该研究验证了通用强化学习算法在多个具有挑战性的领域均能达到超人的水平。

2.1 棋类游戏研究现状

国际象棋

1997年，国际象棋程序深蓝（Deep Blue）[5] 在击败人类世界冠军后，人工智能领域实现了具有里程碑意义的突破。在随后的二十年里，国际象棋计算机程序持续超越人类水平。这些程序采用手工特征和精细调整的权重来评估棋局，利用众多巧妙的启发式方法和特定领域规则来构建搜索树，同时结合高效的 Alpha-Beta 搜索技术[6]。该方法通过“剪枝”技术消除了某些明显劣于已探索分支的分支。在后续实验中，AlphaZero 着重研究了2016年顶级国际象棋引擎锦标赛（TCEC）上的世界冠军 Stockfish [7]，以及其他强大的国际象棋程序，如 Deep Blue 等。这些程序在架构上具有极高的相似性。



(图1: 从左到右: 国际象棋、将棋、围棋。)

将棋

将棋 (Shogi, 也称日本象棋) 是一种源自日本的传统棋类游戏, 拥有丰富的策略和技巧。就计算复杂性而言, 与国际象棋相比, 将棋是一种更难的游戏。将棋在9x9的棋盘上进行, 共有40个棋子, 分为两方, 每方各20个。棋子包括不同种类的武将, 如国王 (玉将/王将)、金将、银将、飞车、角行、桂马、香车和步兵等。每种棋子都有其特定的移动规则, 使得游戏策略极为丰富。将棋的一个独特之处在于其"打ち歩" (Drop) 规则。"打ち歩" (Drop) 规则允许玩家在捕获对手棋子后, 将其翻转并作为己方棋子重新放回棋盘上。这一规则为游戏增添了更多战略层次和复杂性。最强大的将棋程序如计算机将棋协会 (CSA) 的世界冠军 Elmo [8], 击败了人类冠军悬殊。这些程序使用与国际象棋程序类似的算法, 同样使用了许多特定领域知识以及高度优化的 Alpha-Beta 搜索引擎。

围棋

围棋 (Go), 起源于古代中国, 是一种具有悠久历史和丰富文化底蕴的棋类游戏。围棋在全球范围内广泛流行, 特别是在东亚地区, 如中国、日本和韩国等国家。围棋以其简单的游戏规则、高度的策略性和深邃的哲学内涵而闻名。围棋在19x19的棋盘上进行, 棋子分为黑白两色, 双方轮流落子。游戏的目标是通过占领棋盘上的领土和捕获对手棋子来争取最高分。与国际象棋和将棋相比, 围棋的基本规则相对简单: 每次落子只需放置在棋盘上的交叉点, 并在必要时执行“提子”动作, 即移除被包围的对手棋子。尽管围棋的规则简单, 但其策略复杂度极高。棋局的可能性几乎无穷无尽, 使得围棋成为人工智能领域的重要挑战。传统围棋的研究方法主要依赖于启发式搜索和专家知识。然而, 这些方法在处理围棋复杂性方面受到了很大的限制。

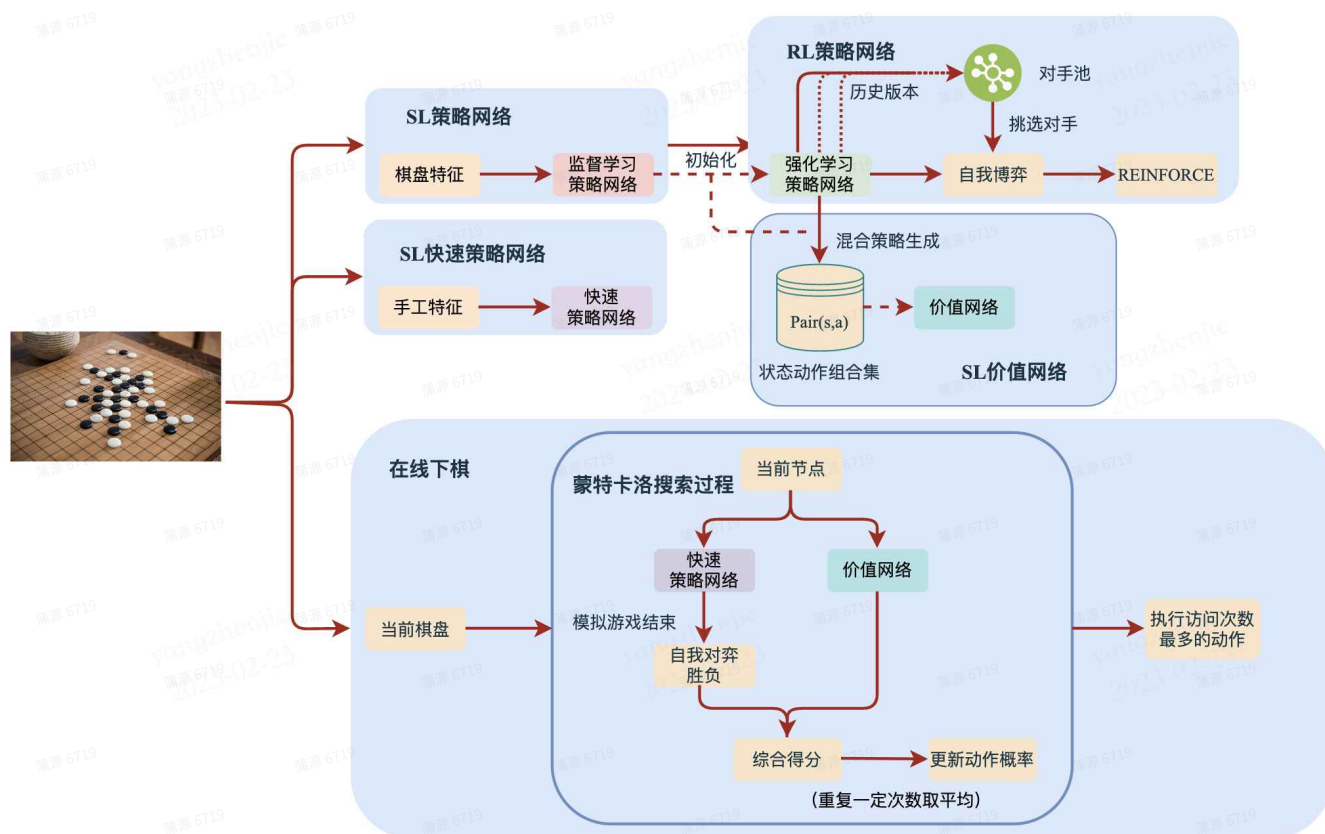
DeepMind 推出的 AlphaGo、AlphaGo Zero 和 AlphaZero 一系列算法, 才成功地将深度学习和强化学习相结合, 以解决围棋中的挑战。AlphaGo 中使用的神经网络架构是针对围棋特殊设计的, 因为游戏规则是平移不变的, 正好符合卷积网络的权重共享性质。卷积网络的局部结构也对应于棋盘上各点之间的连接, 并且具有旋转不变性和对称性质, 适合进一步去做数据增强。此外, 围棋动作空间简单, 并且游戏结果只有输和赢, 没有平局。这两者都可能有助于神经网络训练。国际象棋和将棋不太适合 AlphaGo 的神经网络架构, 因为国际象棋和将棋的规则是位置相关的并且不具有对称性质, 以及包括远程交互等。国际象棋的动作空间包括棋盘上所有玩家棋子的所有符合规则的棋

盘位置，将棋还允许将捕获的棋子放回棋盘上。国际象棋和将棋除了输赢之外，还可能出现平局。为了解决上述问题，DeepMind提出通用方法AlphaZero，在不依赖于除游戏规则之外的任何先验知识下，结合强化学习、自我博弈以及蒙特卡洛搜索树，在国际象棋、将棋和围棋上都超越了人类玩家水平。

2.2 相关工作

2.2.1 AlphaGo

2016年 AlphaGo 4:1 战胜世界冠军李世石，是人工智能领域里程碑式的事件。围棋是一个具备完全信息的游戏，通常具备完全信息的游戏都会有一个最优的价值函数，它能够通过穷举法在任何的状态(棋面)下知道游戏的最终胜负。但是，对围棋穷举的复杂度非常的高，假设落子的位置数为 w ，游戏的长度为 d ，复杂度则是 w^d 。如图2所示为 AlphaGo 整体框架图。



(图2: AlphaGo 整体框架图: 包括监督学习策略网络、监督学习快速策略网络、监督学习价值网络、强化学习策略网络以及在线下棋的流程。)

AlphaGo 的训练步骤

- 基于监督学习的策略网络，收集16万局人类玩家的棋谱组成29.4M状态动作对 (state-action pair) 来进行训练。
- 基于监督学习的快速策略网络，同样利用人类玩家的棋谱进行监督学习，训练简化版的快速策略，用于蒙特卡洛搜索树。快速策略网络相比于策略网络使用了更简单的棋类特征以及网络结构。

- 基于强化学习的策略网络，在监督学习训练得到的策略网络基础上，将当前网络与之前的任一网络随机进行对打，利用强化学习中的 REINFORCE 算法进行训练，最大化获胜的奖励期望，并不断地将中间训练结果加入到对手池中。
- 基于强化学习的价值网络，利用监督学习的策略网络和强化学习的策略网络收集状态动作对，训练价值网络，优化目标为最小化当前状态的价值与最终胜负。

○ AlphaGo 的下棋步骤（结合快速策略网络、价值网络和 MCTS 算法）

- 在某一状态建立 MCTS 搜索树。
- 利用策略网络计算该状态下落子概率权重。
- 利用价值网络对状态进行评估，同时利用快速策略网络从当前状态出发模拟至游戏结束，得到最终结果。结合价值网络的评估值和最终胜负计算出最终评估值。
- 利用最终评估值反向传播更新节点的价值。
- 重复上述操作，当某个节点访问次数超过某个阈值，即拓展下一个节点。

○ AlphaGo 中的 MCTS 与传统 MCTS 的区别

- AlphaGo 使用动作价值函数，而传统的 MCTS 使用状态价值函数。
- 二者在动作选择时计算的 UCB 也不同

● 传统MCTS中UCB公式

$$UCB = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$$

w_i 表示第 i 个节点的获胜次数（继续模拟到游戏结束的获胜次数）， n_i 表示第 i 个节点的访问次数， N_i 表示在第 i 个节点的父亲节点的访问次数， c 为探索系数，默认为 $\sqrt{2}$ ，在实践中通常根据经验进行选择。UCB 分数有两部分组成，左边表示利用（exploitation），是该节点的平均收益值（越高表示这个节点期望收益高，更偏向于选择；右边表示探索（exploration），用父节点的总访问次数除以子节点的访问次数，如果子节点访问次数越少则值越大，越偏向选择，因此 UCB 公式是可以兼顾探索和利用的。

● AlphaGo 中 UCB 计算公式为

$$UCB(s_t, a) = Q(s, a) + cP(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}$$

左边同样表示利用（exploitation），利用价值网络计算当前 (s, a) 下的估值。右边表示探索（exploration），其中分子 $P(s, a)$ 表示在状态 s ，执行动作 a 的概率，其由监督训练的策略网络计算得出。而在传统的 MCTS 方法中，分子只是单纯的访问次数，与 (s, a) 无关。因此，AlphaGo 在计算 UCB 时考虑到了不同的 (s, a) 在人类经验中出现的概率（即先验概率）可能是不同的。

2.2.2 AlphaGo Zero

- AlphaGo Zero 与 AlphaGo 的区别
 - AlphaGo Zero 不需要除规则外的人类玩家棋谱等先验知识，只需从自我博弈中进行学习。
 - 编码信息中无手动编码的特征，只保留棋盘信息。
 - 策略网络和价值网络合二为一，并且结构采用带有残差连接的 ResNet。网络最后设计为多头模式，分别为策略预测网络和价值预测网络。
 - AlphaGo Zero 不需要快速策略网络进行 Rollout，用价值网络预测的 \bar{v} 来计算这个叶子节点的状态价值。
- AlphaGo Zero 的训练步骤
 - 利用当前的神经网络来指导进行蒙特卡洛搜索，收集自我博弈棋局数据。
 - 使用自我博弈棋局数据，训练策略网络和价值网络，训练的目标函数是对于每个输入 s ，神经网络输出的 p, v 和蒙特卡洛搜索的结果中的 π, z 差距尽可能的接近（即让策略预测网络和价值预测网络输出的值去预测 MCTS 的搜索结果）。
 - 经过训练的神经网络继续参与自我博弈，产生更高质量的数据，循环上面的步骤。

2.2.3 相关工作对比表

	AlphaGo	AlphaGo Zero	AlphaZero
发表时间	2016 Nature	2017 Nature	2018 Science
训练数据	16万局人类专家数据	自我博弈	自我博弈
网络架构	单独的策略网络和价值网络	策略网络价值网络共享主干网络，输出层分为Policy Head 和Value Head。	策略网络价值网络共享主干网络，输出层分为Policy Head 和Value Head。
棋类	围棋	围棋	围棋，国际象棋，日本将棋
性能	击败围棋世界冠军李世石	击败AlphaGo	在围棋，国际象棋，日本将棋超越人类水平

（表1：AlphaGo，AlphaGo Zero 和 AlphaZero 的对比表。）

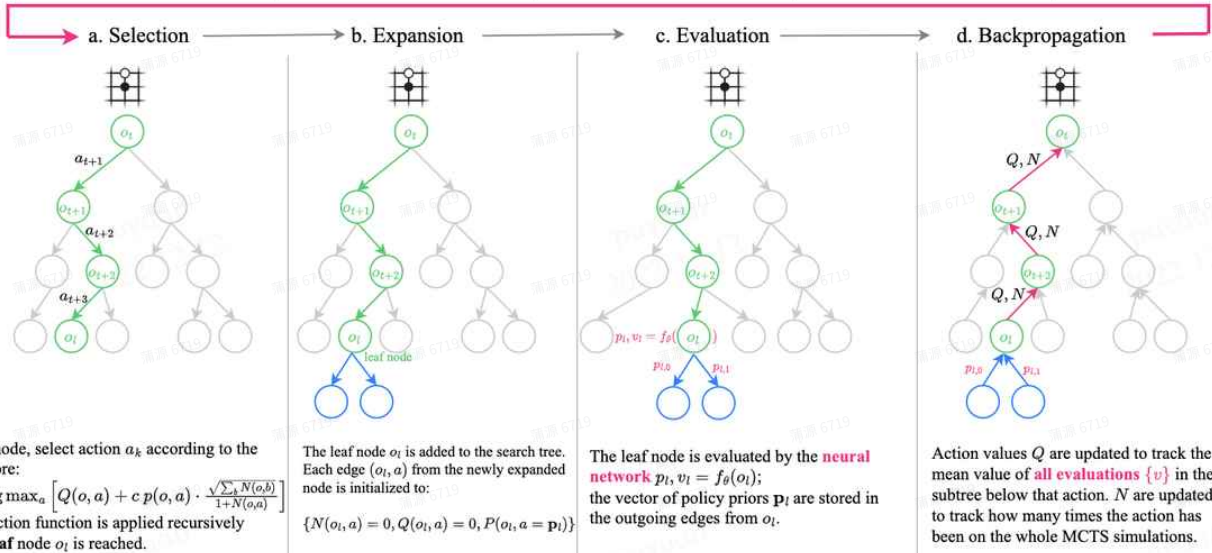
我们在表1中对比了 AlphaGo、AlphaGo Zero、AlphaZero 的发表时间、训练数据、网络架构、适应的棋类游戏以及性能的异同。

3. 算法

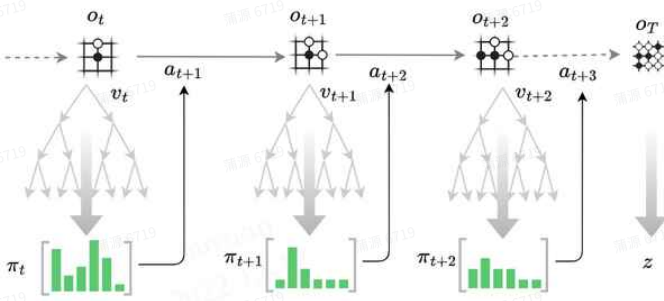
3.1 概览图

AlphaZero: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

A. MCTS in AlphaZero



B. Acting (Self-play)



C. Training

$$l_t(\theta) = l^v(z_t, v_t) + l^p(\pi_t, \mathbf{p}_t) + c \|\theta\|^2$$

where, z_t is the game reward from the **perspective of the current player** and π_t is the MCTS searched policy at timestep t .
 v_t and \mathbf{p}_t is the predicted value and policy from the neural network f_θ .
 l^v is MSE loss, l^p is cross-entropy loss.

(图3: AlphaZero 算法概览图。A: 利用 MCTS 进行搜索的过程, 包括选择、扩展、评估和回溯四步。B: 对 MCTS 返回的根节点的 visit count 的分布进行采样, 获得最终的输出动作, 并通过 self-play 收集样本数据。C: 损失函数包括, MCTS 得到的动作分布和策略网络的动作分布的交叉熵损失, 环境真实 return 和价值网络输出的预测值之间的均方差损失。即让预测的 value, policy 朝着 MCTS 搜索的方向和环境真实 return 逼近。)

3.2 AlphaZero 概述

AlphaZero 算法是 AlphaGo Zero 算法的更通用版本。它使用深度神经网络、通用的强化学习算法和通用的树搜索算法取代了传统游戏程序中使用的含有人类先验的手工特征和增强方式。不同于手工制作的评估函数和启发式的滑动排序, AlphaZero 采用参数为 θ 的深度神经网络 $(p, v) = f_\theta(s)$ 。该神经网络将棋盘位置 s 作为输入, 输出为每个动作 a 的概率 $p_a = P_r(a|s)$; 同时输出一个标量值 v , 用于预测位置 s 上的期望收益 z , $v \approx \mathbb{E}[z|x]$ 。AlphaZero 完全从自我博弈的对局中学习这些动作的概率和状态的价值估计, 然后再利用动作概率和状态估计进一步指导搜索。

AlphaZero 没有使用具有特定领域先验知识的 Alpha-Beta 搜索，而是使用通用的蒙特卡洛树搜索 (MCTS) 算法。每次搜索(Each Search)都包含一系列模拟的自我博弈游戏，相当于从根状态到叶子节点遍历一整棵树，并根据当前神经网络 $f(\theta)$ ，每次模拟 (Each simulation) 通过在状态 s 中选择一个具有低访问次数（即之前没有充分探索过的）、高移动概率和高价值 (averaged over the leaf states of simulations that selected a from s) 的动作 a 来进行。搜索返回一个向量 π 表示动作概率分布， $\pi_a = P_r(a|s_{root})$ 。

AlphaZero 中深度神经网络的参数 θ 是从自我博弈的对局中强化学习训练得到的，从随机初始化的参数 θ 开始。每次对局，都是在当前的棋面 $s_{root} = s_t$ 执行 MCTS 得到动作概率分布 $\pi_t = P_r(a|s_t)$ ，然后选择动作 $a_t \sim \pi_t$ 来玩游戏（与根状态的访问计数成比例（用于探索）或贪婪地（用于开发）的选择）。在游戏结束时，根据游戏规则进行评分，计算出游戏结果 z ：输 -1，平 0，赢 +1。更新神经网络参数 θ 以最小化预测结果 v_t 和游戏结果 z 之间的误差，并最大化策略网络输出的概率 p_t 与搜索概率 π_t 的相似性。具体公式可参考 3.4 小节。

3.3 AlphaZero 与 AlphaGo Zero 的区别

AlphaZero 使用同一种算法学习三种不同的棋类，并都取得了超人的水平。本文中描述的 AlphaZero 算法在以下几个方面不同于原始的 AlphaGo Zero 算法：

- AlphaGo Zero 假设只有输赢的结果，并估计和优化获胜的概率。AlphaZero 会估计和优化预期结果，同时考虑平局或其他潜在结果。
- 围棋的规则对于旋转和反射是不变的。AlphaGo 和 AlphaGo Zero 都利用了这一性质。首先，通过为每个盘面生成 8 个对称的盘面来扩充训练数据。其次，在 MCTS 期间，在由神经网络评估之前，使用随机选择的旋转或反射来转换棋盘位置，使得蒙特卡洛评估是在不同 biases 上的平均结果。为了适应更广泛的游戏类别，AlphaZero 没有使用对称性的假设；国际象棋和将棋的规则是不对称的（例如，兵只能向前移动，kingside 和 queenside 的易位是不同的）。AlphaZero 没有增强训练数据，也没有在 MCTS 期间变换棋盘位置。



- 在 AlphaGo Zero 中，自我博弈的数据是通过先前迭代中最佳智能体相互对打产生的。新智能体的表现都会与最佳智能体进行比较；如果它以 55% 的优势获胜，那么它会取代最好的智能体，然后由这个新智能体进行自我博弈。相比之下，AlphaZero 只是维护一个持续更新的神经网络，自我博弈是通过使用该神经网络的最新参数生成的，省略了评估步骤和最佳智能体的选择。

- AlphaGo Zero 使用贝叶斯优化搜索得到的最佳超参数。在 AlphaZero 中，我们为所有游戏使用相同的超参数、算法设置和网络架构，无需针对游戏进行特定调整。唯一的例外是策略中确保探索的噪声和学习率的设置，这个需根据不同游戏种类额外设置。
- 与 AlphaGo Zero 一样，根据每种游戏的基本规则，棋盘状态通过空间平面进行编码。同样地，动作也仅基于每种游戏的基本规则，由空间平面或平面向量进行编码。

3.3 AlphaZero 中的 MCTS 流程

假设此时 AlphaZero 的策略网络和价值网络已经训练好。每条边保存的信息为：

$N(s, a), P(s, a), Q(s, a), R(s, a), T$ ，MCTS 在模拟游戏中搜索的过程可以分为以下4个阶段：



1. 选择 (selection)

- UCB，结合价值函数+策略函数，选择 score 最高的 action

$$a^k = \arg \max_a \{Q(s, a) + cP(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1+N(s, a)}\}$$

- 访问次数 N 、平均价值 Q 、策略 P 、奖励 R 和真实状态转移 T 。
- 通过UCB公式选择得分最高的action，利用该公式一直搜索到叶子结点。

2. 扩展 (expansion)

- 根据上面选择方式继续搜索，直到遇到叶子节点（即没有孩子节点的节点）。
- 将叶子节点加入到搜索树中，由环境得到当前状态下的可选action
- 利用这些可选action进行孩子节点的拓展，同时初始化其孩子节点信息 $\{N(s^i, a) = 0, Q(s^i, a) = 0, P(s^i, a)\}$ 。

3. 评估 (evaluation)

- 通过价值网络和价值网络计算输出动作概率和状态价值 p^l, v^l ，并将动作概率保存到边上。

4. 回溯 (backup)

- 更新所在支路的 V 和 N ：

$$V(s^k) := \frac{V(s^{k-1})}{N(s^{k-1})+1},$$

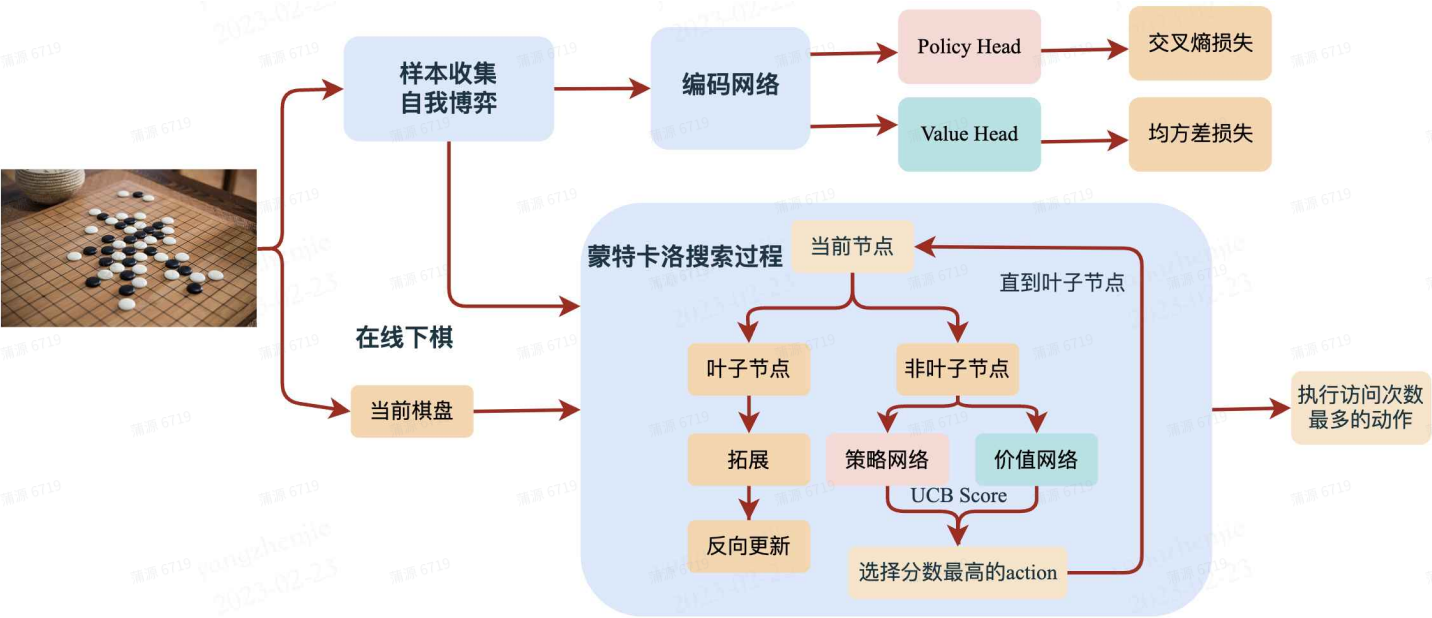
$$N(s^k) := N(s^{k-1}) + 1.$$

在 MCTS 搜索完成后，在根节点返回 visit counts 集合 $\{N(s, a)\}$ ，从这个集合中选取访问次数最多的动作，即得到最终的执行动作。

3.4 训练与评估

在 AlphaGo 中，模型训练的样本有一部分来源于人类专家数据，但通过对 AlphaGo 输给李世石的一局中分析，状态价值在第87手后值网络输出的对当前棋局的评估逐渐变差。价值网络只能评估当前的局势，然而，棋局形势的演化通常需要多步棋的连贯铺垫才能逐渐形成。AlphaGo 使用监督学习的策略拓展搜索树的节点，如果职业玩家下出“神之一手”，由于人类专家数据的局限性，AlphaGo 无法正确地作出应对。为了解决这个问题，在 AlphaGo Zero 以及 AlphaZero 训练中，放弃使用人类专家数据，所有训练样本都是通过自我博弈的形式产生的，在训练过程中，策略函数和价值函数不断优化，并且指导 MCTS 进行搜索，同时利用 MCTS 搜索结果与环境交互产生训练样本，进而再优化目标函数。通过这种策略提升的方法，不断地获得更高质量的训练数据。


训练流程



(图4: AlphaZero 训练流程图：上半部分为训练流程，下半部分为在线对弈时的流程。)

如图4所示，在训练阶段，首先通过自我博弈获得数据样本，在自我博弈时，利用当前状态建立蒙特卡洛搜索树根节点，通过策略网络和价值网络计算 UCB 得分，并选择得分最高的 action，直到遇到叶子节点。将叶子节点进行拓展，根据价值网络的输出，反向更新该支路的节点信息并模拟多次。模拟多次后，从 visit count 分布中进行采样，从而作为当前状态下的动作。

当游戏的完整一局结束后，将样本数据存放在经验回放当中，用于训练强化学习网络。强化学习网络包括策略网络和价值网络，策略网络通过交叉熵损失进行优化，价值网络通过 MSE 损失进行优化。随着策略和价值函数的不断提升，训练样本也在不断地变好，从而训练出较强的智能体。损失函数为



$$l = l_v(z_t, v_t) + l_p(\pi_t, p_t) + c||\theta||^2$$
$$l_v = (z - v)^2$$
$$lp(\pi_t, p_t) = -\pi_t^T \log p_t$$

其中 l_v 为均方差损失， l_v 为交叉熵损失， z_t 为游戏的收益， v_t 为价值网络的输出， π_t 为 MCTS 的搜索策略， p_t 为策略网络的输出。

评估流程

在评估阶段，AlphaZero 在执行 action 时，利用当前状态建立蒙特卡洛搜索树根节点，并开始模拟。在模拟过程中利用策略网络和价值网络计算 UCB 得分，并选择得分最高的action，直到探索到叶子节点。然后将叶子节点进行拓展，并根据价值网络的输出反向更新该支路的节点信息。多次模拟后，选择执行次数最多的动作为当前状态下的最佳动作。

3.5 实践细节

输入输出的表征

在本节中，我们将描述 AlphaZero 中神经网络使用的棋盘输入表征和动作输出表征。

Go		Chess		Shogi	
Feature	Planes	Feature	Planes	Feature	Planes
P1 stone	1	P1 piece	6	P1 piece	14
P2 stone	1	P2 piece	6	P2 piece	14
		Repetitions	2	Repetitions	3
				P1 prisoner count	7
				P2 prisoner count	7
Colour	1	Colour	1	Colour	1
		Total move count	1	Total move count	1
		P1 castling	2		
		P2 castling	2		
		No-progress count	1		
Total	17	Total	119	Total	362

(表2：分别表示AlphaZero在 Go、Chess 和Shogi 使用的输入特征。对 T=8 步历史中的每个位置重复第一组特征。计数由单个实值输入表示；其他输入特征由使用指定数量的二进制独热编码表示。当前玩家用 P1 表示，对手用 P2 表示。)

Chess		Shogi	
Feature	Planes	Feature	Planes
Queen moves	56	Queen moves	64
Knight moves	8	Knight moves	2
Underpromotions	9	Promoting queen moves	64
		Promoting knight moves	2
		Drop	7
Total	73	Total	139

(表 3：AlphaZero 分别在国际象棋和将棋中使用的动作表示。由一维特征表示，这些特征编码了合法移动的概率分布。)

网络结构

AlphaZero 与 AlphaGo Zero 网络架构相同。由一个编码器 (Encoder) 和后面的策略预测网络 (Policy Head) 和价值预测网络 (Value Head) 组成。编码器由卷积层和残差块 (ResNet) 组成。卷积层由256 个卷积核大小 (kernel size) 为 3x3 且步幅 (stride) 为 1 的卷积组成。策略预测网络由卷积层与线性层组成，线性层的大小为棋类可执行的动作数目。价值预测网络由卷积层和一个输出大小为 1 的 tanh 线性层组成。

4. 实验

4.1 实验设置

	Chess	Shogi	Go
Mini-batches	700k	700k	700k
Training Time	9h	12h	34h
Training Games	44 million	24 million	21 million
Thinking Time	800 sims 40 ms	800 sims 80 ms	800 sims 200 ms

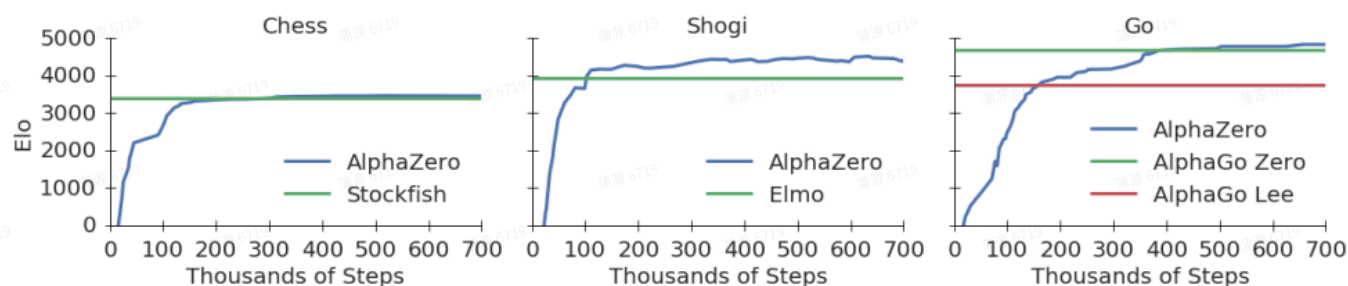
(表4: AlphaZero 在各个棋类游戏上的 mini-batch 数量, 训练时间, 训练游戏局数和思考时间。)

每种比赛训练了一个单独的 AlphaZero 实例。训练从随机初始化的参数开始, 以 batch size 为 4096 进行 700k step 训练。其中 5000 个第一代 TPU 用于生成 self-play 游戏, 16 个第二代 TPU 用于训练神经网络。在训练期间, 每个 MCTS 使用 800 次模拟。每种游戏的训练时长, 思考时间等信息总结在表 4 中。每一局的学习率初始设定为 0.2, 随后下降三次 (分别到 0.2, 0.02, 0.002)。落子动作的选择和 MCTS 中根节点的访问次数成正比。根节点的先验概率中添加了狄利克雷噪声 $Dir(\alpha)$, 与某个位置中的合法棋步的近似数量成反比。在国际象棋、将棋和围棋中分别设置为 $\alpha = \{0.3, 0.15, 0.03\}$ 。除了特殊说明的情况外, AlphaZero 的训练和搜索算法与策略都与 AlphaGo Zero 一致。

4.2 实验结果

4.2.1 AlphaZero 的 Elo 性能以及与其他棋类算法的对比

图 5 展示了 AlphaZero 在 self-play RL 中的性能与训练步数的关系, 这里性能以 Elo[9] 为评估单位。在国际象棋中, AlphaZero 仅训练了 4 个小时 (300k steps) 就超过了 Stockfish。在将棋中, AlphaZero 训练不到 2 小时 (110k steps) 就超过了 Elmo。在围棋中, AlphaZero 在训练了 8 小时 (165k steps) 后就超过了 AlphaGo Lee。



(图5: AlphaZero 训练 700k step 后与基线算法在各个棋类上的 Elo 比较。左图: AlphaZero 在国际象棋上, 和 2016 年 TCEC 世界锦标赛的冠军程序 Stockfish 的性能比较。中图: AlphaZero 在将棋上, 和 2017 年 CSA 世界锦标赛的冠军程序 Elmo 的性能比较。右图: AlphaZero 在围棋上, 和 AlphaGo Lee 以及 AlphaGo Zero (20 个残差网络, 训练 3 天) 的性能比较。)

4.2.2 AlphaZero 与其他棋类算法的胜负比较

表5展示了比赛规则中1分钟思考时间的限制下，AlphaZero 在各个棋类上与其他算法100局的胜负情况。其中 AlphaZero 和 AlphaGo Zero 都使用一台4个 TPU 的机器。而 Stockfish 和 Elmo 都以最高等级进行比赛，使用 64个线程和1 GB 的 hash size。很显然，AlphaZero 战胜了所有对手，在与 Stockfish 的对战中无一败绩，而在与 Elmo 的对战中 只输了8局。同样地，AlphaZero 也战胜了 AlphaGo Zero。

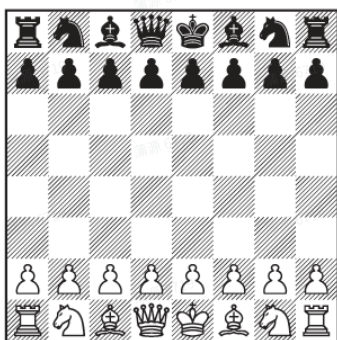
Game	White	Black	Win	Draw	Loss
Chess	<i>AlphaZero</i>	<i>Stockfish</i>	25	25	0
	<i>Stockfish</i>	<i>AlphaZero</i>	3	47	0
Shogi	<i>AlphaZero</i>	<i>Elmo</i>	43	2	5
	<i>Elmo</i>	<i>AlphaZero</i>	47	0	3
Go	<i>AlphaZero</i>	<i>AG0 3-day</i>	31	—	19
	<i>AG0 3-day</i>	<i>AlphaZero</i>	29	—	21

(表5：各个棋类每100局，AlphaZero 的胜负场次，对手算法分别为 Stockfish，Elmo 和训练3天的AlphaGo Zero，每个程序有1分钟的思考时间。)

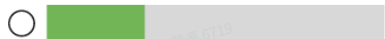
A

Chess

AlphaZero vs. Stockfish



W: 29.0% D: 70.6% L: 0.4%



W: 2.0% D: 97.2% L: 0.8%

Shogi

AlphaZero vs. Elmo



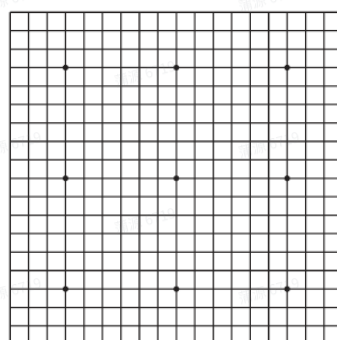
W: 84.2% D: 2.2% L: 13.6%



W: 98.2% D: 0.0% L: 1.8%

Go

AlphaZero vs. AG0



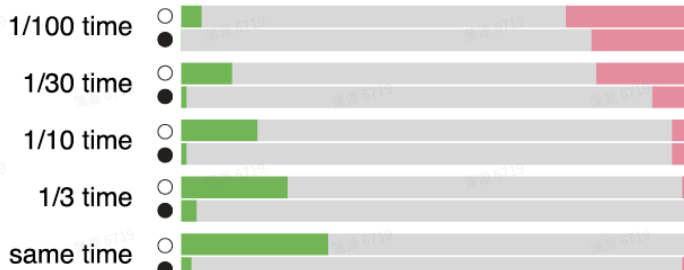
W: 68.9% L: 31.1%



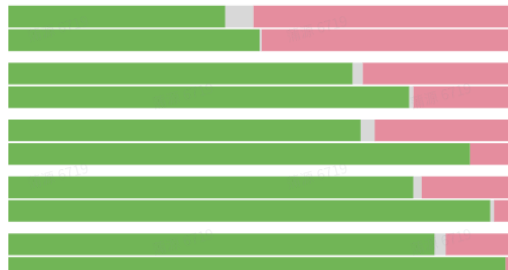
W: 53.7% L: 46.3%

B

Chess



Shogi



C

Latest Stockfish



Aperyphapaq



Opening Book



CSA time control



D

Human openings



TCEC openings



AlphaZero wins

AlphaZero draws

AlphaZero loses

AlphaZero white

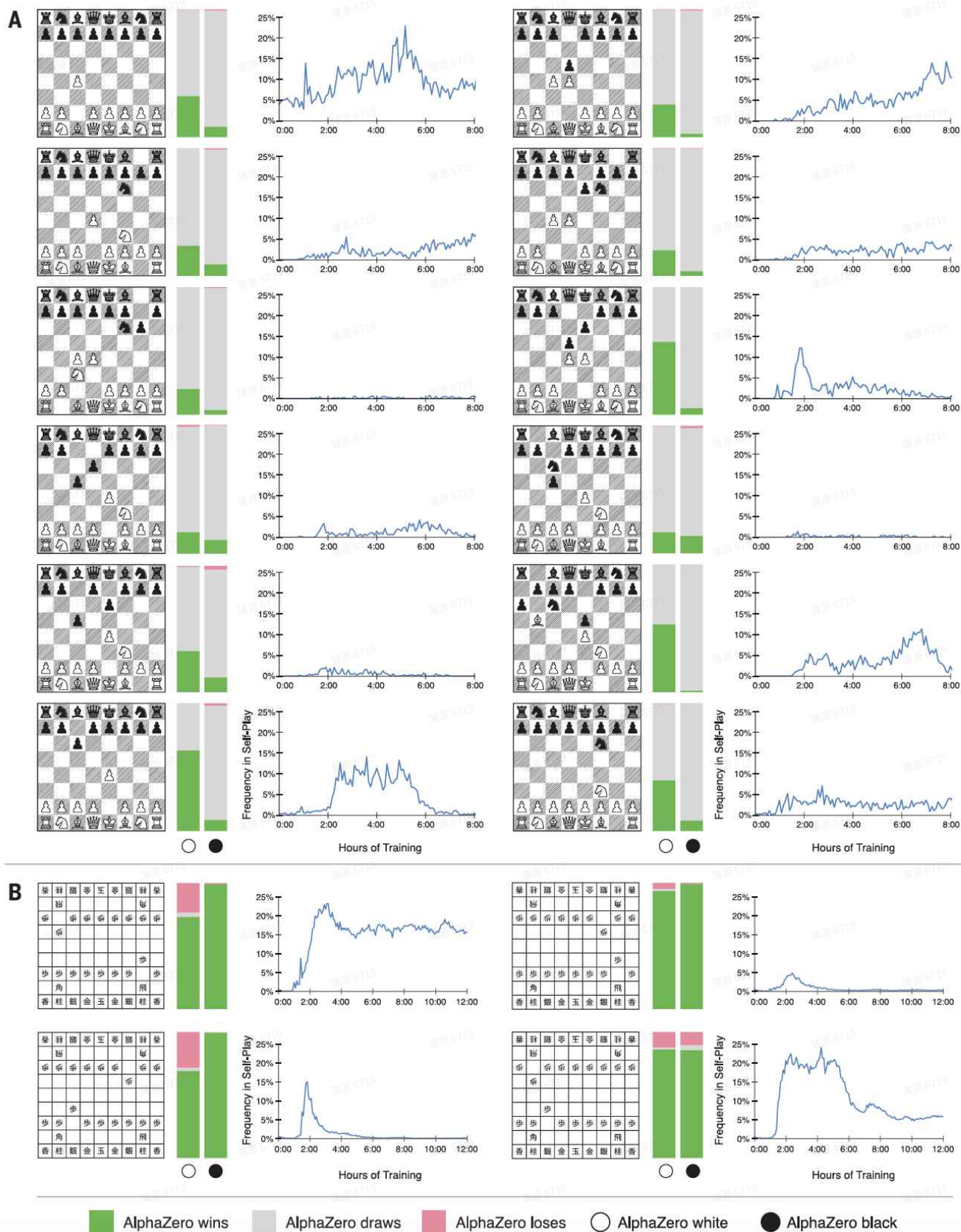
AlphaZero black

(图 6: 与现有的棋类程序对比。(A) AlphaZero 在国际象棋、将棋和围棋中分别与 Stockfish、Elmo 和之前发布的经过 3 天训练的 AlphaGo Zero (AG0) 版本进行比赛评估。在下面条形图中, 上方为 AlphaZero 执白棋; 下方为 AlphaZero 执黑棋。比赛结果都是从 AlphaZero 的角度表示: 获胜 (W: 绿色)、平局 (D: 灰色) 或失败 (L: 红色)。(B) 与 Stockfish 和 Elmo 相比, AlphaZero 的可扩展性和思考时间。Stockfish 和 Elmo 所用时间固定为每场比赛 3 小时以及每次移动限制时间为 15 秒。AlphaZero 随着时间的增长, 取胜和平局的概率增大。(C) AlphaZero 在国际象棋中的额外评估: 对战(在撰写本文时)最新版本的 Stockfish 以及对战具有强大 opening book 的 Stockfish。将棋中的额外评估: 在全时控制 (full time controls) 下对战另一个强大的将棋

程序 Aperyqhapaq，在 2017 年 CSA 世界锦标赛时间控制（每场比赛 10 分钟，每次动作 10 秒）下对阵 Elmo。(D) 在国际象棋不同的开局下 AlphaZero 的表现。)

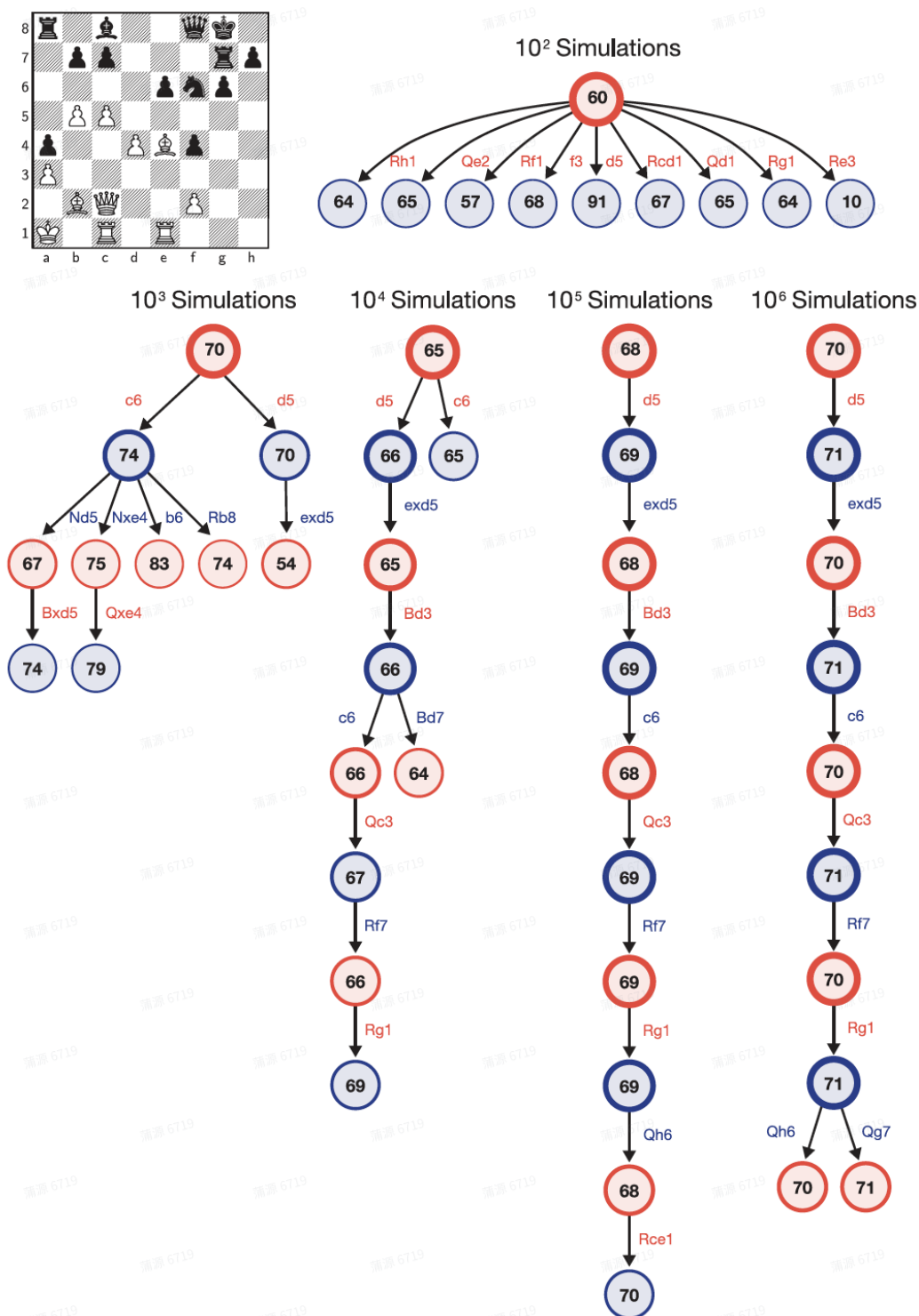
4.2.3 AlphaZero 对棋类知识的掌握

作者分析了 AlphaZero 发现的一些棋类知识。图7展示了国际象棋最常见的12种人类开局和将棋最常见的4种人类开局，每种开局都是由 AlphaZero 单独发现，并在 self-play 中常常使用的。同时，从每个开局开始，AlphaZero 都显然战胜了 Stockfish 和 Elmo，这表明 AlphaZero 确实掌握了国际象棋和将棋中广泛的棋路。



(图7: AlphaZero 在国际象棋中与 Stockfish 对战以及在将棋中与 Elmo 对战。在左栏中, AlphaZero 从给定的位置开始执白棋; 在右栏中, AlphaZero 执黑棋。每个条形图表示胜负结果, 都是从 AlphaZero 的角来评定胜负: 其中**获胜** (绿色)、**平局** (灰色) 或**失败** (红色)。折线图表示 AlphaZero 选择此开局进行自我博弈的百分比频率与训练持续时间 (以小时为单位) 的关系。)

4.2.4 AlphaZero 的 MCTS 搜索过程

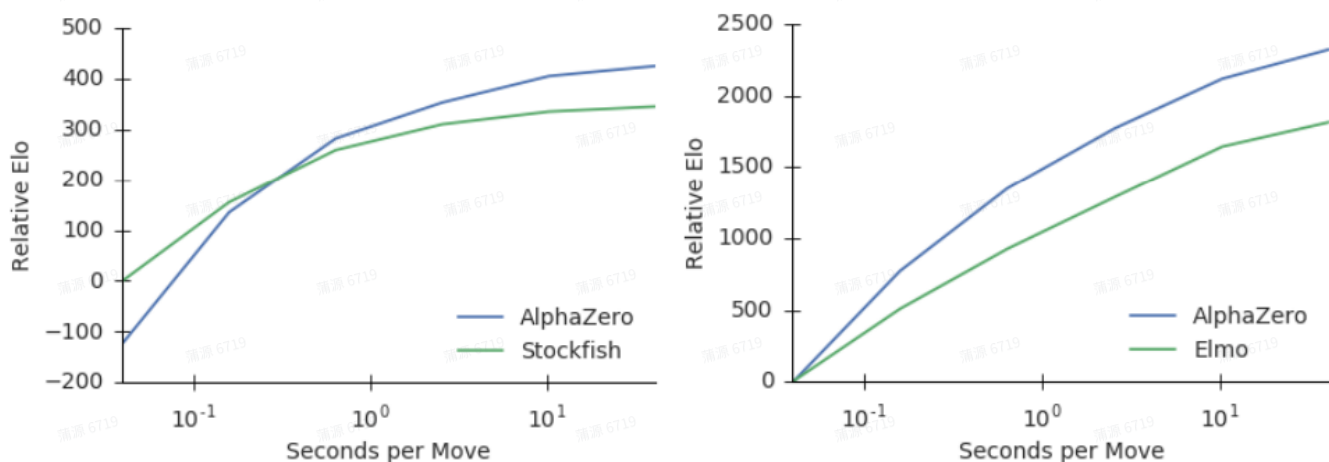


(图 8: AlphaZero 的搜索过程。在 AlphaZero (白方) 和 Stockfish (黑方) 对战的一局游戏 (整局游戏过程参考原论文表 S6) 执行 29 步 Qf8 之后的状态, 如左上角的插图所示。在这个状态上分别经过 10^2 , ..., 10^6 次模拟后, AlphaZero 的 MCTS 的内部状态 (internal state, 用圆圈表示) 结果。每个结果显示了前 10 个访问量最大的 internal state, (从白方角度的) 估计值显示在每个状态对应的圆圈中, 统一缩放到了 $[0, 100]$ 。每个内部状态的访问计数, (对于该树的根状态) 与圆圈边界的厚度成正比。AlphaZero 在开始模拟时考虑了动作 30.c6 但最终选择了动作 30.d5。)

4.2.5 AlphaZero 的性能扩展能力

图9展示了每种算法的性能在思考时间上的可扩展性, 评估标准为 Elo。相对于思考时间为40毫秒的 Stockfish 和 Elmo, AlphaZero 的 MCTS 随着思考时间增长的性能提升比 Stockfish 和 Elmo 更

为显著，这与人们的普遍认知 Alpha-Beta 搜索在这方面更有优势的结论相反。Alpha-Beta 剪枝搜索之所以厉害，是因为它在保证找到最优解的前提下，通过剪枝策略显著减少了搜索空间，提高了搜索效率。相比之下 AlphaZero 使用蒙特卡洛树搜索（MCTS）作为其主要搜索策略，MCTS 通过对不确定性的处理和有效的资源分配，实现了更高效的搜索。结合神经网络的评估函数，AlphaZero 可以更聚焦于有更好前景的走子选项，从而在更少的搜索步骤内找到优秀的解决方案。且由于其从零开始，仅通过自我对弈来探索棋盘空间、学习策略和提升棋艺。这使得 AlphaZero 能够找到人类可能忽略的策略，并在棋局中获得竞争优势。



（图9：AlphaZero 的性能（以 Elo 为评估标准）随着每步思考时间的扩展能力。左图：AlphaZero 和 Stockfish 在国际象棋上，性能与每步思考时间关系的对比。右图：AlphaZero 和 Elmo 在国际象棋上，性能与每步思考时间关系的对比。）

5. 总结与展望

AlphaZero 是一种结合深度强化学习和 MCTS 的人工智能算法，可以在除游戏规则外没有其他先验知识的情况下不断地自我学习，并在许多场景取得超越人类玩家的成果。

以下是对 AlphaZero 的总结与展望：

总结：

- 自我学习能力强：AlphaZero 能够在没有先验知识的情况下自我学习，只需要提供游戏规则和奖励信号，就能逐步提高。
- 多种场景通用：AlphaZero 不仅适用于围棋等棋类游戏，还可以应用于国际象棋、将棋等棋类游戏，展示了广泛的适用性。
- 算法优化：AlphaZero 优化了深度强化学习算法中的多个部分，包括神经网络设计、搜索算法、训练方法等，提高了算法的效率和性能。

展望：

- 更广泛的应用：未来可以探索如何将 AlphaZero 扩展到更广泛的应用领域，例如自动驾驶、医疗保健、金融等。

- 算法进一步优化：虽然 AlphaZero 在深度强化学习领域取得了重大进展，但其算法仍然有改进空间，例如如何更好地应对稀疏奖励、如何更好地处理多智能体环境等等。
- 算法解释性：由于深度强化学习算法具有很强的复杂性，其结果和决策过程往往难以解释和理。未来可以探索如何提高算法的可解释性，使得其结果更容易被人理解和接受。

总的来说，AlphaZero 是深度强化学习领域的一次重大突破，其成功也展示了深度学习和人工智能在游戏和其他领域的广泛应用前景。随着算法的不断改进和拓展，AlphaZero 将在未来继续发挥重要作用，并为人类带来更多的启发。

6. 参考文献

- [1] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *nature* 529.7587 (2016): 484-489.
- [2] Silver, David, et al. "Mastering the game of go without human knowledge." *nature* 550.7676 (2017): 354-359.
- [3] Silver, David, et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play." *Science* 362.6419 (2018): 1140-1144.
- [4] A. L. Samuel. Some studies in machine learning using the game of checkers II - recent progress. IBM Journal of Research and Development, 11(6):601-617, 1967.
- [5] M.Campbell, A.J. Hoane, and F. Hsu. DeepBlue. Artificial Intelligence, 134:57-83, 2002.
- [6] D. E. Knuth and R. W Moore. An analysis of alpha-beta pruning. Artificial Intelligence, 6(4):293-326, 1975.
- [7] Stockfish: Strong open source chess engine; <https://stockfishchess.org/> [accessed 29 November 2017].
- [8] Computer Shogi Association, Results of the 27th world computer shogi championship; www2.computer-shogi.org/wcsc27/index_e.html [accessed 29 November 2017].
- [9] Arpad E. Elo. The Rating of Chessplayers, Past and Present. Arco Publishing, New York, 1978.