AlphaZero

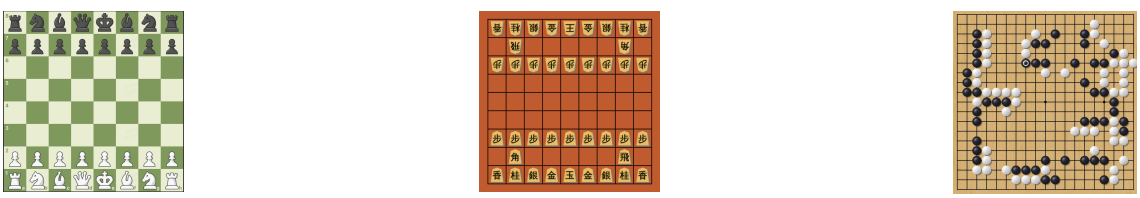
1. 概述

- 当 AlphaGo [1] 和 AlphaGo Zero [2] ——两款由 DeepMind 开发并通过蒙特卡洛树搜索和强化学习技术在围棋领域超越人类的人工智能——问世时，我们对人工智能的可能性有了全新的认识。AlphaZero [3] 算法作为它们的简化增强版本，不仅突破了早期算法的局限，还将自我博弈和强化学习得到的策略性能推向了一个新的高度。AlphaZero 不同于传统的人工智能程序，它不依赖于任何先验知识，除了游戏规则之外。从一个随机策略开始，AlphaZero 通过自我博弈和强化训练，短短24小时内便在国际象棋、将棋（即日本象棋）和围棋等游戏中达到了超人的水平。
- 本文将详细解析 AlphaZero 的工作原理，并探讨它是如何通过自我博弈和强化学习在各种领域取得突破的。无论你是 AI 研究者，还是对人工智能怀有深厚兴趣的读者，我们均相信，这篇文章将为你带来新的启示和灵感。让我们一同深入 AlphaZero 的神奇世界，共同感受这一创新技术如何改变我们对人工智能可能性的理解。

2. 研究背景与相关工作

- 在计算机科学的历史长河中，对国际象棋的研究可以追溯到这个学科的初始阶段。巴贝奇、图灵、香农以及冯诺依曼等科学巨人，都曾专注于硬件设计、算法发展和理论分析，以期深入探索棋局策略。随后的一代人工智能研究者们将国际象棋视为重要的挑战，最终诞生了一些能以超越人类专家水平玩国际象棋的高性能计算机程序。然而，这些程序的应用范围通常局限在特定领域，如果要推广到其他问题或领域，经常需要人类进行大量的调整和优化。
- 自人工智能领域诞生以来，其一直追求的目标就是创造出能从第一性原理（First Principles）[4] 自我学习的程序。2017年，AlphaGo Zero 算法就通过深度卷积神经网络来呈现围棋知识，并通过自我博弈的方式进行强化学习训练，在围棋游戏中实现了超越人类的表现。本文将介绍一种类似但更为通用的算法——AlphaZero，它被应用于国际象棋、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，本研究验证了通用强化学习算法在多个具有挑战性的领域都能达到超越人类的水平。

2.1 棋类游戏研究现状



（图1：从左到右：国际象棋（Chess）、将棋（Shogi）、围棋（Go）

国际象棋

- 1997年，国际象棋（Chess）程序深蓝（Deep Blue）击败人类世界冠军，该项突破被视为具有里程碑意义。在该重大事件后的二十年间，国际象棋计算机程序的水平持续超越人类。这些程序利用人工设定的特征和精心调整的权重来评估棋局，运用多种巧妙的启发式方法和特定领域知识构建搜索树，同时配合了高效的 Alpha-Beta 搜索技术 [6]。这种方法通过“剪枝”技术消除了明显劣于已探索分支的其他分支。在后续的实验中，AlphaZero 着重研究了2016年顶级国际象棋引擎锦标赛（TCEC）的世界冠军 Stockfish [7]，以及深蓝等其他强大的国际象棋程序。国际象棋在线玩的网站。

将棋

- 将棋（Shogi），也称为日本象棋，是一种充满策略和技巧的传统日本棋类游戏。从计算复杂性的角度来看，将棋比国际象棋更为复杂。将棋在9x9的棋盘上进行游戏，共有40个棋子，分为两方，每方各有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，每种棋子都有其特别的移动规则，使得游戏策略丰富多变。将棋的特别之处在于其"打ち歩"（Drop）规则，这个规则允许玩家捕获对方的棋子后，将其翻转并作为己方棋子重新放回棋盘上，这为游戏增加了更深的战略层次和复杂性。计算机将棋协会（CSA）的世界冠军 Elmo [8] 是最强大的将棋程序之一，它已经击败了人类的世界冠军。这些程序使用与国际象棋程序类似的算法，同样采用了大量的特定领域知识以及高度优化的 Alpha-Beta 搜索引擎。

围棋

- 围棋（Go），源于古代中国，是一种历史悠久且富含文化内涵的棋类游戏。其在全球范围内广受欢迎，特别是在中国、日本、韩国等东亚地区。围棋以其简洁的规则、高度的策略性以及深邃的哲学涵义而闻名。在19x19的棋盘上进行的围棋，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领土和捕获对方棋子来争取最高分。相比国际象棋和将棋，围棋的基本规则更为简单：每次只需将棋子放置在棋盘交叉点上，必要时执行“提子”动作，即移除被完全包围的对手棋子。然而，尽管规则简单，围棋所涉及的策略复杂度却极高，因此，围棋一直是人工智能领域的重大挑战。传统的围棋研究方法主要依赖于启发式搜索和专家知识，但这些方法在处理围棋的复杂性时受到了很大的限制。
- DeepMind 推出的 AlphaGo、AlphaGo Zero 和 AlphaZero 一系列算法成功地结合了深度学习和强化学习，从而有效地解决了围棋中的挑战。AlphaGo 所使用的神经网络架构是专为围棋设计的，这得益于围棋规则的平移不变性，这种性质恰好符合卷积网络的权重共享特性。卷积网络的局部结构也对应于棋盘上各点之间的连接，具有旋转和对称性质，适合进行数据增强。此外，围棋的动作空间简单，且游戏只有胜败，没有平局，这有助于神经网络的训练。

国际象棋和将棋的规则与围棋有所不同，它们的规则与位置相关，不具有对称性，还包括远程交互等因素。国际象棋和将棋的动作空间包括棋盘上所有棋子的所有合法移动位置，而将棋还允许将捕获的棋子放回棋盘上。此外，国际象棋和将棋可能出现平局。为了解决这些问题，DeepMind 提出了通用

方法 AlphaZero，它不依赖于任何先验知识，仅依赖游戏规则，通过结合强化学习、自我对弈和蒙特卡洛搜索树，在国际象棋、将棋和围棋上均超越了人类的水平。

2.2 相关工作

2.2.1 相关工作对比表

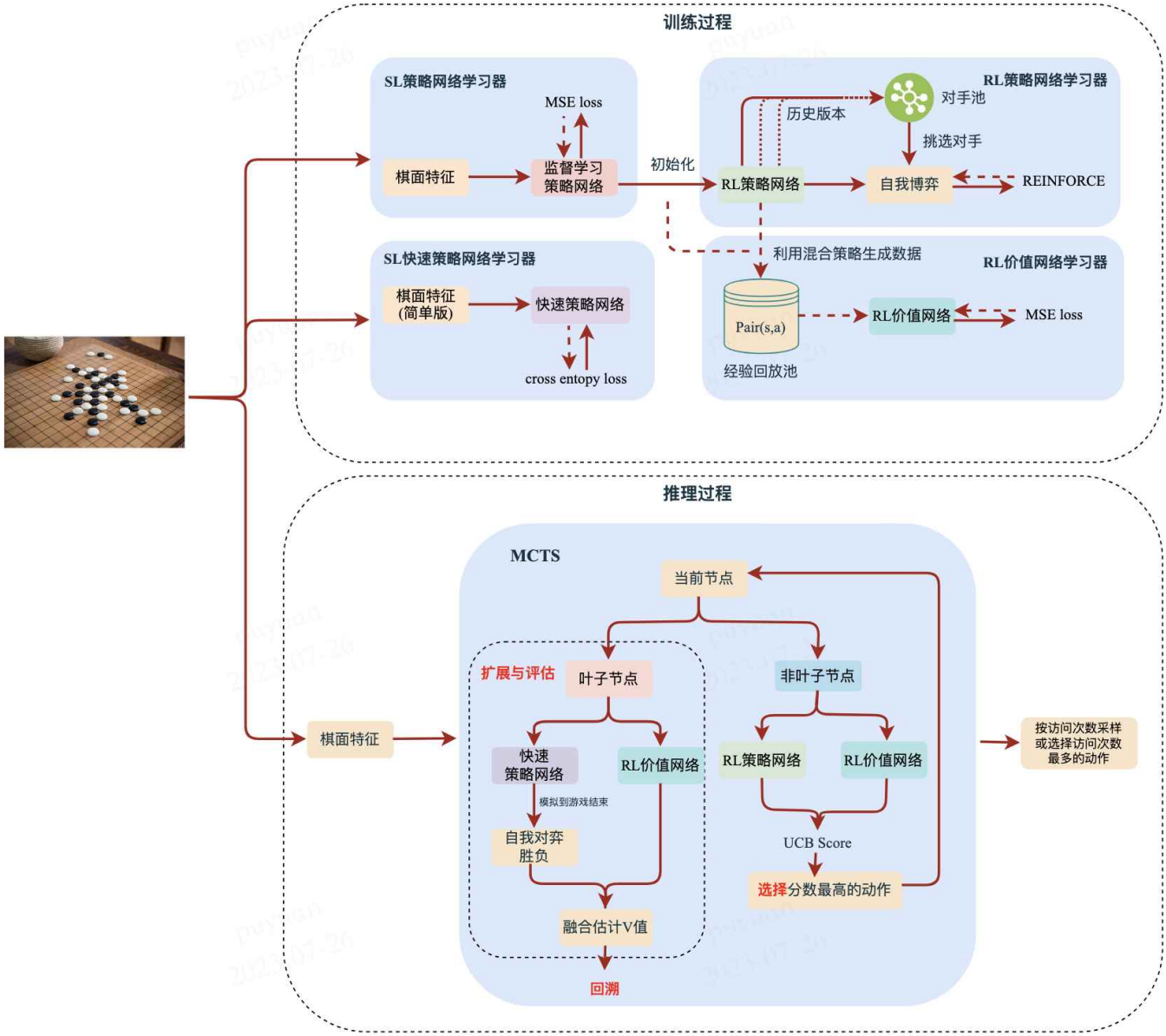
	AlphaGo	AlphaGo Zero	AlphaZero
发表时间	2016 Nature	2017 Nature	2018 Science
训练数据	16万局人类专家数据	自我博弈	自我博弈
网络架构	单独的策略网络和价值网络	策略网络价值网络共享主干网络，输出层分为 Policy Head和 Value Head。	策略网络价值网络共享主干网络，输出层分为 Policy Head和 Value Head。
棋类	围棋	围棋	围棋，国际象棋，日本将棋
性能	击败围棋世界冠军李世石	击败 AlphaGo	在围棋，国际象棋，日本将棋超越人类水平

（表1：AlphaGo，AlphaGo Zero 和 AlphaZero 的简要对比表。）

为方便大家区分，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。

2.2.2 AlphaGo

围棋作为一种完全信息游戏，理论上存在一个最优的价值函数。这个价值函数通过穷举法，能够在任何棋面状态下预判游戏的最终胜负。然而，围棋的穷举复杂度极高。假定落子位置数为 w ，游戏长度为 d ，则复杂度就达到了 w^d 。这种高度的复杂性使得围棋长期以来都是人工智能领域的一大挑战。然而，2016年，这个挑战得到了突破。AlphaGo 以4:1的比分战胜了世界冠军李世石，这个事件成为了人工智能领域的里程碑。AlphaGo 的整体框架如图2所示，它的成功不仅彰显了人工智能的强大实力，也为围棋这种传统游戏赋予了新的科技内涵。



（图2：AlphaGo 训练概览图。上半部分：AlphaGo 的训练过程。包括基于监督学习的策略网络（SL策略网络）、基于监督学习的快速策略网络（SL快速策略网络）、基于强化学习的策略网络（RL策略网络）、基于强化学习的价值网络（RL价值网络）的学习过程。下半部分：AlphaGo 下棋时给出动作的推理过程。）

- AlphaGo 的训练过程
  - 基于监督学习的策略网络（SL策略网络）的训练。它从16万局人类玩家的棋谱中收集了29.4M状态动作对 (state-action pair)，并通过监督学习进行训练。
  - 基于监督学习的快速策略网络（SL快速策略网络）的训练。它同样利用人类玩家的棋谱进行监督学习，但是训练的是一种简化版的快速策略。该策略主要用于蒙特卡洛搜索树。相比于基于监督学习的策略网络，SL快速策略网络使用了更简单的类特征以及网络结构。
  - 基于强化学习的策略网络（RL策略网络）：它在监督学习训练得到的SL策略网络基础上，让当前网络与之前的任一网络进行随机对打。然后，利用强化学习中的 REINFORCE 算法进行训练，以最大化期望奖励。并且，它会不断地将中间训练结果（RL策略网络）加入到对手池中。

- 基于强化学习的价值网络（RL价值网络）：它利用SL策略网络和RL策略网络收集状态动作对，用于训练价值网络，优化目标为最小化当前状态的价值与最终胜负的 MSE。

- AlphaGo 的推理过程
  - AlphaGo 在下棋时，会将RL策略网络、RL价值网络、SL快速策略网络和MCTS算法结合起来。
  a. 在某一状态下，AlphaGo 会建立一个MCTS搜索树。
  b. 利用RL策略网络计算该状态下的落子概率权重。
  c. 接着，AlphaGo 会利用RL价值网络对当前状态进行评估。同时，使用SL快速策略网络从当前状态模拟至游戏结束，得到最终结果。然后，结合RL价值网络的评估值和模拟的最终胜负计算出最终评估值。
  d. 之后，AlphaGo 会利用计算出的最终评估值反向传播，以更新节点的价值。
  e. 最后，AlphaGo 会重复以上操作，当某个节点被访问的次数超过某个阈值时，就会拓展下一个节点。

- AlphaGo 中的 MCTS 与传统 MCTS 的区别
  - 传统 MCTS 中 UCB 公式为

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

    - 其中 w_i 表示在本次模拟时第 i 个节点的获胜次数，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分数由两部分组成，第1项表示利用（exploitation）项，是该节点的平均收益值（状态值的估计）；第2项表示探索（exploration）项，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该项越大。因此 UCB 公式是可以兼顾探索和利用的。
  - AlphaGo 中 UCB 计算公式为

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

    - 第1项同样表示利用项，是RL价值网络和SL快速策略网络进行融合估计的结果，具体地，对于叶子节点的评估是利用RL价值网络估计当前 s_L 的状态值估计 $v_\theta(s_L)$ 与利用SL快速策略网络从该叶子节点rollout到终局得到的真实回报 z_L 的线性混合；第2项表示探索项，其中分子中的 $P(s, a)$ 表示在状态 s 执行动作 a 的概率，由 RL策略网络计算得出，$N(s, a)$ 表示在状态 s 选择动作 a 的次数。
  - 小结：
    - 在传统的蒙特卡洛搜索树（MCTS）方法中，利用项是基于完整模拟得到的估计值。然而，AlphaGo创新地采用了 RL价值网络和SL快速策略网络进行融合估计，这种方法能够减少搜索的深度，大大提高了估计的效率。

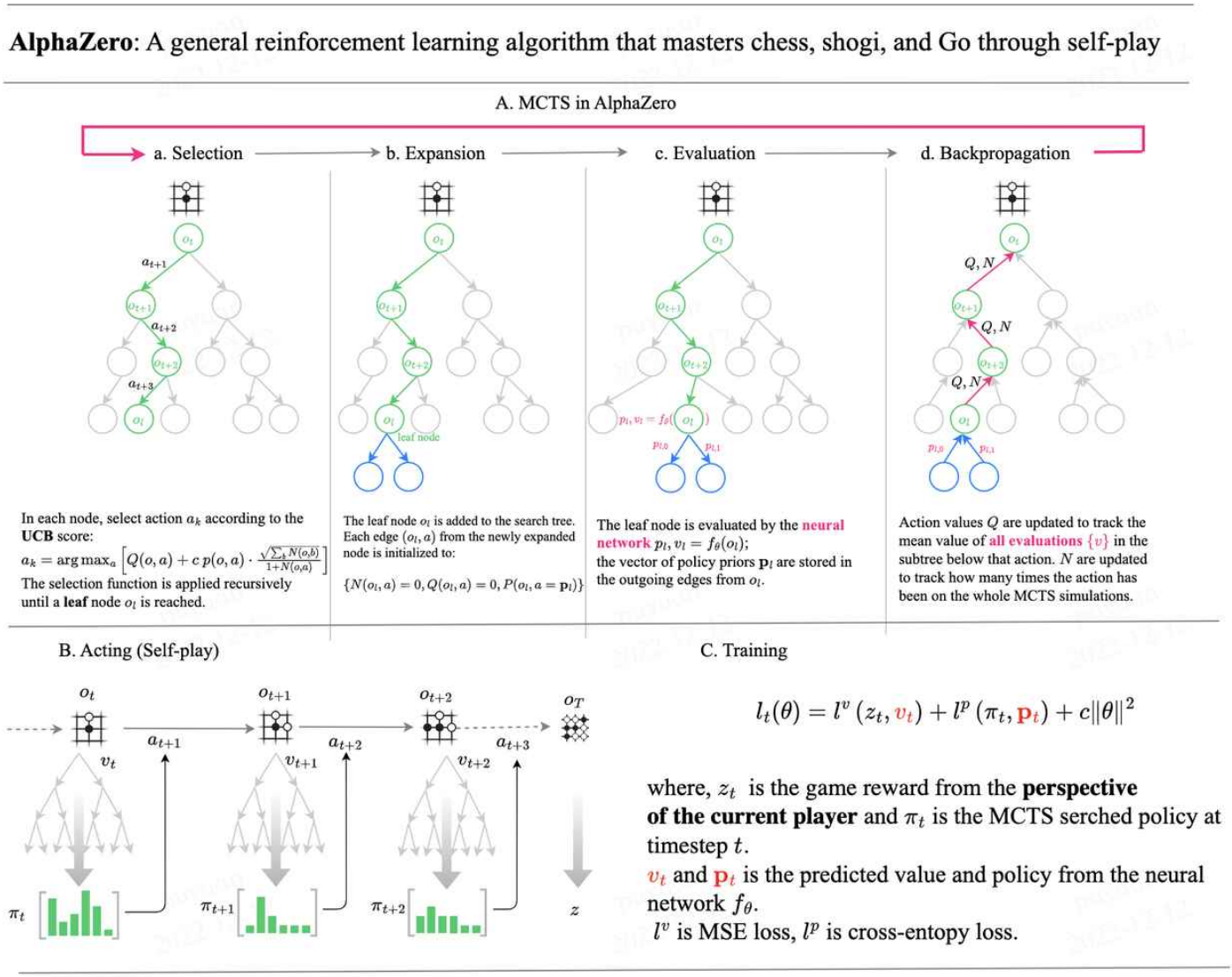
        - 另外，在传统的 MCTS 方法中，探索项的分子仅仅是访问次数，与状态动作对 (s,a) 无关。但是，AlphaGo 则巧妙地考虑到了在人类专家经验中，不同的状态动作对 (s,a) 出现的先验概率。这种方法通过借助专家的经验，有效地减少了搜索的宽度。这两点创新，都使得AlphaGo在围棋对弈中的表现更为出色。

2.2.3 AlphaGo Zero

- AlphaGo Zero 与 AlphaGo 的核心差异
    - AlphaGo Zero 的独特之处在于，除了规则外，它无需依赖人类玩家的棋谱或其他先验知识。它完全通过自我博弈进行学习，从零开始逐渐积累经验和知识。
    - 在编码信息上，AlphaGo Zero 摒弃了所有手动编码的特征，仅保留了棋盘信息，这更加简洁和高效。
    - 在网络结构上，AlphaGo Zero 将策略网络和价值网络融合为一体，且采用了具有残差连接的 ResNet 结构。网络最后设计为多头模式，一头为策略预测网络，另一头为价值预测网络。
    - 此外，AlphaGo Zero 无需使用 SL快速策略网络进行 Rollout，仅使用价值网络预测的 ☒ 来计算叶子节点的状态价值。
- AlphaGo Zero 的训练流程
    - 首先，利用当前的策略和价值神经网络指导进行蒙特卡洛搜索，收集自我博弈棋局数据。
    - 然后，使用这些自我博弈棋局数据，训练策略网络和价值网络。训练的目标是让神经网络输出的 (p,v) 和蒙特卡洛搜索的结果中的 (π,z) 尽可能接近。也就是说，试图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。
    - 最后，训练出的神经网络继续参与自我博弈，生成更高质量的数据，然后再重复上述步骤，形成一个持续学习和优化的闭环。

3. 算法

3.1 概览图



(图3：AlphaZero 算法概览图。**A**：MCTS 算法进行搜索的过程，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 policy，value 朝着 MCTS 搜索得到的动作分布和环境真实 return 的方向逼近。符号的含义参考算法概览图符号表。)

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数据增强方式。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 $(p, v) = f_\theta(s)$。该神经网络将棋盘状态（棋面） s 作为输入，输出为每个动作 a 的概率 $p_a = P_r(a|s)$；同时网络也输出一个标量值 v ，用于预测棋面 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。每次模拟（Each simulation）会根据当前神经网络 $f(\theta)$ ，通过在棋面 s 中选择一个同时具有低访问次数（即之前没有充分探索过的）、高移动概率和高价值特点的动作 a 来进行。MCTS 搜索会返回一个向量 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通过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地选择（一般用于在collect时保持探索）或贪婪地选择（一般用于在eval） 。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。 通过最小化预测结果 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 的相似性来更新神经网络参数 θ 。 具体公式可参考3.4小节。

3.3 AlphaZero 与 AlphaGo Zero 的区别

AlphaZero采用了一种独特的算法，通过学习三种不同的棋类，都取得了超越人类的水平。相比于原先的AlphaGo Zero，AlphaZero在以下几个方面不同于原始的 AlphaGo Zero 算法：

- 在胜负预测上，AlphaGo Zero仅考虑输赢的最终结果，估计并优化获胜的概率。相反，AlphaZero则采用更全面的视角，它同时考虑平局和输赢，以此估计并优化预期的游戏结果。
- 在棋盘对称性方面，AlphaGo 和 AlphaGo Zero 都充分利用了围棋规则的特性，即其对旋转和反射保持不变。它们首先通过为每个棋局生成8个对称的棋面，从而扩充训练数据。其次，在蒙特卡洛树搜索(MCTS)期间，通过对棋盘位置进行随机选择的旋转或反射，然后再由神经网络进行评估，以实现蒙特卡洛评估的多样性。然而，为了适应更广泛的棋类游戏，AlphaZero 并未采用这种对称性假设。例如，国际象棋和将棋的规则就具有非对称性（如，兵只能向前移动，kingside 和 queenside 的易位规则不同）。因此，AlphaZero 既没有通过增强训练数据来使用对称性，也没有在 MCTS 期间变换棋盘位置。

- 在数据生成和智能体更新的机制上，AlphaGo Zero 是通过由先前迭代中最佳智能体对战产生的自我博弈数据。新的智能体表现会与最佳智能体进行对比，只有当其胜率超过55%时，才会取代最佳智能体，并由这个新的智能体进行自我博弈。相比之下，AlphaZero 则选择了更简洁的机制，它仅维护一个持续更新的神经网络，所有的自我博弈数据都是通过使用该神经网络的最新参数生成的，从而避免了评估步骤和最佳智能体的选择。
- 在超参数选择上，AlphaGo Zero 采用贝叶斯优化搜索得到的最佳超参数。而 AlphaZero 则采用统一的超参数、算法设定和网络架构，为所有游戏提供一致的设定，无需为特定的游戏进行调整。唯一的例外是需要根据不同游戏的特性进行调整的探索噪声和学习率。
- 在棋盘状态和动作的编码上，AlphaZero 与 AlphaGo Zero 保持一致，都是根据每种游戏的基本规则，通过空间平面对棋盘状态进行编码，而动作则通过空间平面或平面向量进行编码。

3.4 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：$N(s,a), P(s,a), Q(s,a)$，其中各项符号的含义为：访问次数 N 、平均价值 Q、策略 P。以下是 AlphaZero 中的 MCTS 流程的主要步骤：

💡

1. 选择 (selection)
    - 从根节点开始，递归地选择“最有价值”的子节点，直到找到一个叶子节点（即没有孩子节点的节点）。选择的策略是基于子节点的价值，先验概率和访问次数的平衡，具体地，通过下面的 UCB (Upper Confidence Bound) 公式选择得分最高的动作 [12]：

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

    - 其中各项符号的含义为，访问次数 N、平均价值 Q、策略 P、奖励 R 和真实状态转移 T。
2. 扩展 (expansion)
    - 根据上面的选择方式进行搜索，直到遇到叶子节点。
    - 将叶子节点加入到搜索树中，由环境的机制得到当前状态下的可选动作 (leagal action set)
    - 利用这些可选动作进行孩子节点的拓展，同时初始化其孩子节点的信息：$\{N(s^L,a)=0,\ Q(s^L,a)\ =\ 0,\ P(s^L,a)\}$。
3. 评估 (evaluation)
    - 通过策略网络和价值网络估计可选动作的动作概率和状态价值 p^L，v^L，并将动作概率保存到对应的边上。
    - 通过策略网络和价值网络，估计叶子节点对应的可选动作集合的概率分布和状态价值 p^L，v^L，并将每一个动作概率值 $P(s,a)$ 保存到对应的边上。

4. 回溯/反向传播 (backpropogation)
    - 在第 k 次模拟快结束时，更新搜索轨迹上的 V 和 N 。评估值会被反向传播回搜索路径中的所有节点。对于每个访问过的节点，它的访问次数会增加1，而它的价值则会更新为已经进行的 k 次模拟的平均价值。

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

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

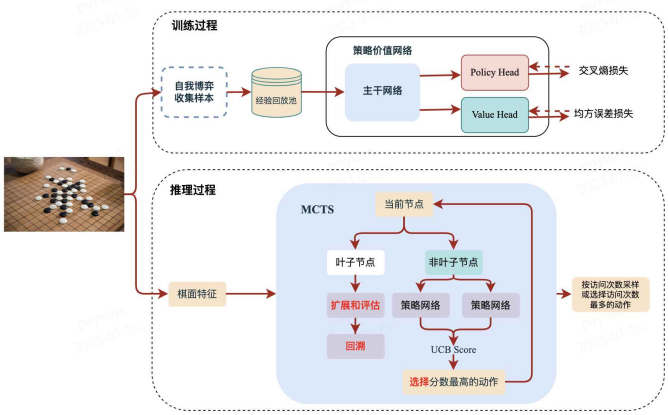
- 执行：上述过程会重复多次（在 AlphaZero 中，每步棋会进行约800次迭代），在 MCTS 搜索完成后，在根节点返回 visit counts 集合 $\{N(s,a)\}$。然后，AlphaZero 会根据根节点的子节点的访问次数来选择一个动作。具体来说，它会选择访问次数最多或者从 visit counts 导出的分布中进行采样得到最终的执行动作。

3.5 训练与评估

在早期的 AlphaGo 模型中，部分训练样本是源自人类棋手的专家数据。然而，通过对 AlphaGo 在与李世石对战的一局中的分析，我们可以发现在第87手之后，价值网络对当前棋局的评估逐渐偏离实际情况。价值网络仅能评估当前的局面，而棋局的演进往往需要多步棋的精准演算才能逐渐显现。AlphaGo 利用监督学习的策略扩展搜索树的节点，但如果对手下出“神之一手”，由于依赖人类专家数据的局限，AlphaGo 可能无法做出正确的应对。

为了解决这个问题，后续的 AlphaGo Zero 和 AlphaZero 的训练中，完全放弃了使用人类专家数据。所有训练样本都是通过自我博弈的方式生成，策略函数和价值函数在训练过程中不断优化。这两者指导蒙特卡洛树搜索(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。通过这种策略的逐步提升，可以不断地获得更高质量的训练数据。

训练流程



(图4：AlphaZero 训练概览图。上半部分：通过交叉熵损失优化策略网络，通过均方误差损失优化价值网络的训练过程。下半部分：AlphaZero 下棋时给出动作的推理过程。)

如图4所示，

- 首先通过自我博弈获得数据样本。在自我博弈中，AlphaZero 首先根据当前状态构建蒙特卡洛搜索树的根节点，利用策略网络和价值网络计算 UCB 得分，并选择得分最高的动作，直至遇到叶子节点。对叶子节点进行扩展，根据价值网络的输出，反向更新该路径的节点信息并进行多次模拟。多次模拟后，返回 visit counts 集合 $\{N(s,a)\}$，从 visit counts 导出的分布中进行采样，作为当前状态下的执行动作。
- 然后，当游戏的一局结束后，将样本数据存放在经验回放池中，从中采样样本用于训练强化学习网络。强化学习网络包括价值网络和策略网络，价值网络通过均方误差损失进行优化，策略网络通过交叉熵损失进行优化。随着策略和价值函数的不断提升，训练样本也在不断改进，从而训练出的智能体能力逐渐增强。
- 具体的损失函数为

🎉

$$l = l_v(z_t, v_t) + l_p(\pi_t, p_t) + c||\theta||^2$$

$$l_v = (z - v)^2$$

$$l_p(\pi_t, p_t) = -\pi_t^T log p_t$$

其中 l_v 为均方差损失，l_p 为交叉熵损失，z_t 为游戏的实际结果，v_t 为价值网络的输出，π_t 为 MCTS 搜索得到的策略，p_t 为策略网络的输出。

评估流程

在评估阶段，AlphaZero 经过 MCTS 的多次模拟后，返回 visit counts 集合 $\{N(s,a)\}$，选择访问次数最多的动作作为当前状态下的执行动作。

3.6 实践细节

输入输出的表征

本节将详细讨论AlphaZero中神经网络所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在围棋（Go）、国际象棋（Chess）和将棋（Shogi）中使用的输入特征。在这些游戏中，它会对最近的 $T = 8$ 步历史中的每个棋面重复应用第一组特征。而第二组特征则是利用了指定数量的二进制 one-hot 编码来表征。在这里，当前玩家标记为P1，对手标记为P2。例如。对于 Go，总的输入平面数量为 $2 * 8 + 1 = 17$。）

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 在国际象棋和将棋中所采用的动作表征。策略表示为一系列平面的堆叠，每个平面表示合法动作对应的概率分布，表格中第二部分的每一行对应一个平面。）

网络结构

AlphaZero 的网络架构与 AlphaGo Zero 基本相同，由编码器（Encoder）、策略预测网络（Policy Head）以及价值预测网络（Value Head）构成。编码器由多个卷积层和残差块（ResNet）组成，其中卷积层由256个卷积核构成，每个卷积核（kernel size）的大小为3x3，步幅（stride）为1。策略预测网络由卷积层和线性层组成，线性层的大小等于棋类游戏中可执行的动作数量。价值预测网络由卷积层和一个输出大小为1的tanh激活函数的线性层组成。

4. 实验

4.1 实验设置

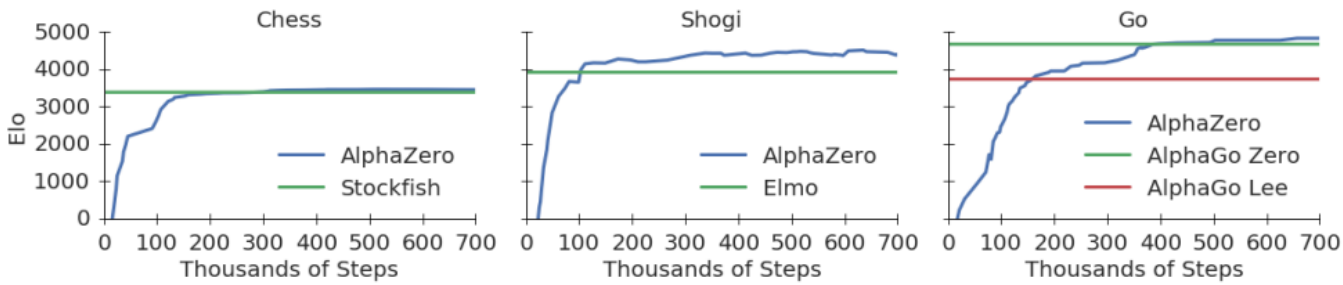
	Chess	Shogi	Go
Mini-batches	700k	700k	700k
Training Time	9h	12h	34h
Training Games	44 million	24 million	21 million
Thinking Time	800 sims 40 ms	800 sims 80 ms	800 sims 200 ms

（表4：AlphaZero 在各个棋类游戏上的用于训练的总的 mini-batch 数量，总训练时间，总的游戏局数和每步的模拟次数与近似思考时间。）

每种棋类游戏都有一个独立的 AlphaZero 实例进行训练。训练从随机初始化的参数开始，以4096为批次大小进行了700k步的训练。在此过程中，5000个第一代TPU被用于生成自我对弈游戏，16个第二代TPU被用于训练神经网络。在训练期间，每个蒙特卡洛搜索树（MCTS）进行了800次模拟。各种游戏的训练长度、思考时间等信息可以在表4中找到。每局游戏的学习率初始设定为0.2，随后进行了三次下降，分别至0.2，0.02和0.002。棋子的动作选择与MCTS中根节点的访问次数成正比。在根节点的先验概率中，加入了狄利克雷噪声 $Dir(\alpha)$，其与某个位置的合法棋步的近似数量成反比。在国际象棋、将棋和围棋中，α 分别设为0.3，0.15和0.03。除非特别说明，AlphaZero的训练和搜索算法及策略都与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：在各个棋类上，AlphaZero 训练 700k step 后与各自的基线程序的 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。）

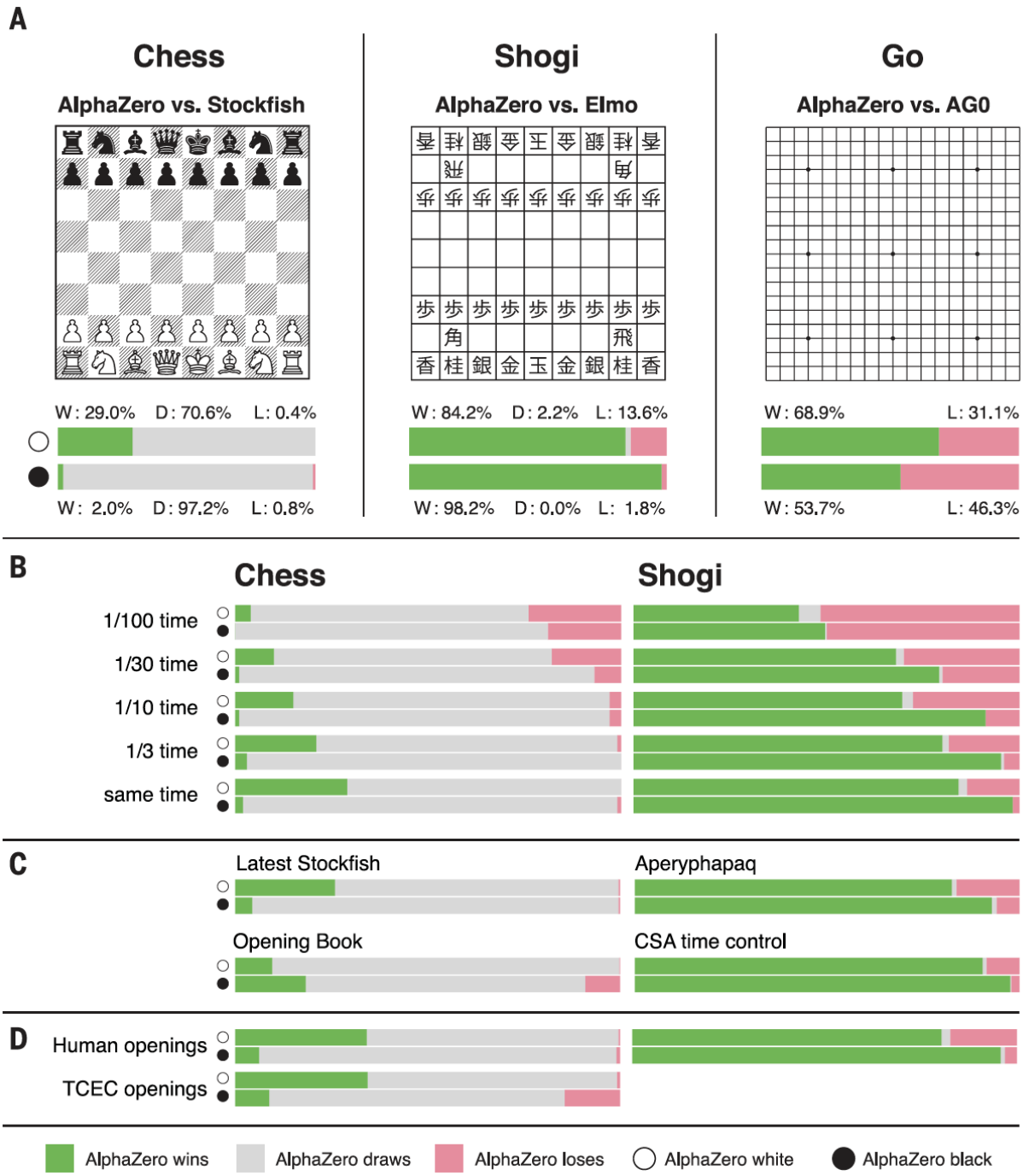
图5展示了AlphaZero在自我对弈强化学习中的性能与训练步数的关系，其中性能以 Elo [9] 评分为评估标准。在国际象棋中，AlphaZero 仅训练了4个小时（300k 步）就超过了 Stockfish。在将棋中，AlphaZero 不到2小时（110k 步）的训练就超过了Elmo。在围棋中，AlphaZero在训练了8小时（165k 步）后就超越了AlphaGo Lee。

4.2.2 AlphaZero 与其他棋类算法的胜负比较

表5展示了在比赛中1分钟思考时间的限制下，AlphaZero 在各种棋类游戏中与其他算法进行的100局比赛的胜负情况。在这些比赛中，AlphaZero 和 AlphaGo Zero 都使用了一台配备4个TPU的机器。而 Stockfish 和Elmo 则以最高级别进行比赛，使用64个线程和1GB的哈希大小。结果表明，AlphaZero在所有比赛中都取得了胜利，在与 Stockfish 的对决中无一败绩，而在与 Elmo 的比赛中仅输掉了8局。同样，AlphaZero 也击败了 AlphaGo Zero。

Game	White	Black	Win	Draw	Loss
Chess	*AlphaZero*	*Stockfish*	25	25	0
	Stockfish	*AlphaZero*	3	47	0
Shogi	*AlphaZero*	*Elmo*	43	2	5
	Elmo	*AlphaZero*	47	0	3
Go	*AlphaZero*	*AG0 3-day*	31	–	19
	AG0 3-day	*AlphaZero*	29	–	21

（表5：AlphaZero与各个棋类的基线程序（分别为Stockfish，Elmo 和训练3天的AlphaGo Zero） 在100局中胜负场次情况。每个程序每一步有1分钟的思考时间。）

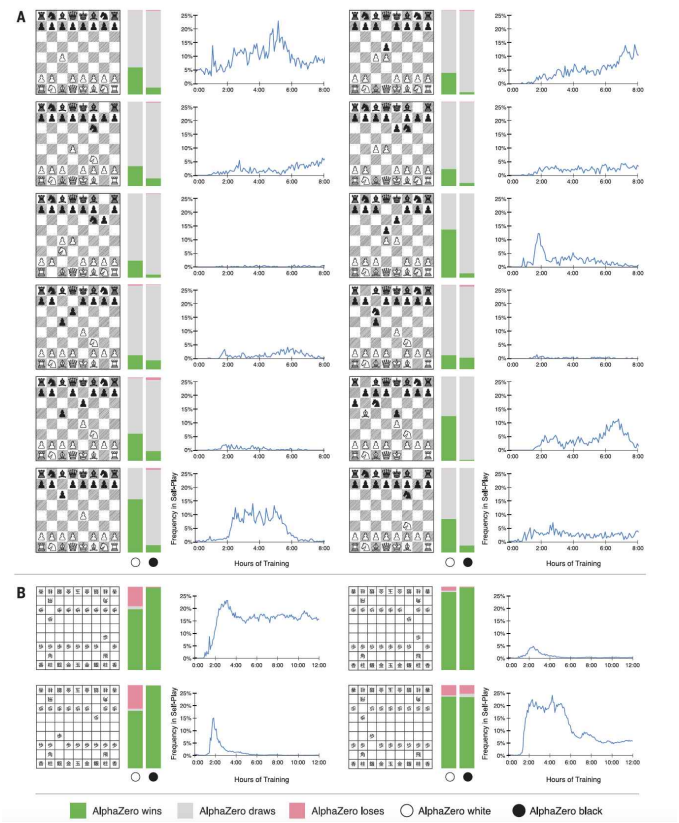


（图 6: 与现有的棋类程序对比。 (**A**) AlphaZero 在国际象棋、将棋和围棋中分别与 Stockfish、Elmo 和之前发布的经过 3 天训练的 AlphaGo Zero (AG0) 版本进行比赛评估。 在下面条形图中，上方为 AlphaZero 执白棋； 下方为 AlphaZero 执黑棋。 比赛结果都是从 AlphaZero 的角度表示：获胜（W：绿色）、平局（D：灰色）或失败（L：红色）。 (**B**) 与 Stockfish 和 Elmo 相比，AlphaZero 的可扩展性和思考时间。 Stockfish 和 Elmo 所用时间固定为每场比赛 3 小时以及每次移动限制时间为 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。）

TODO(pu)：图6展示了 AlphaZero 与现有棋类程序的性能对比 。(A) AlphaZero 在国际象棋、将棋和围棋中与 Stockfish、Elmo 以及之前发布的训练了3天的 AlphaGo Zero (AG0) 版本进行比赛评估。条形图中，上半部分表示 AlphaZero 执白棋的情况；下半部分则表示 AlphaZero 执黑棋的情况。所有比赛结果都从 AlphaZero 的视角展示，包括胜（W：绿色）、平（D：灰色）和负（L：红色）的情况。(B) 与 Stockfish 和 Elmo 相比，AlphaZero 的可扩展性与思考时间。Stockfish 和 Elmo 的比赛时间被设定为每场比赛3小时，并且每次移动的时间限制为15秒。随着思考时间的增加，AlphaZero 获胜和平局的概率呈现增长趋势。(C) AlphaZero 在国际象棋中的额外评估包括：与最新版本的 Stockfish 对决，以及与具备强大 opening book 的 Stockfish 对战。在将棋中的额外评估包括：在全时控制（full time controls）下与强大的将棋程序 Aperyqhapaq 对决，并在 2017 年 CSA 世界锦标赛的时间控制（每场比赛10分钟，每次动作10秒）下与 Elmo 对战。(D) AlphaZero 在国际象棋中不同开局下的性能评估。

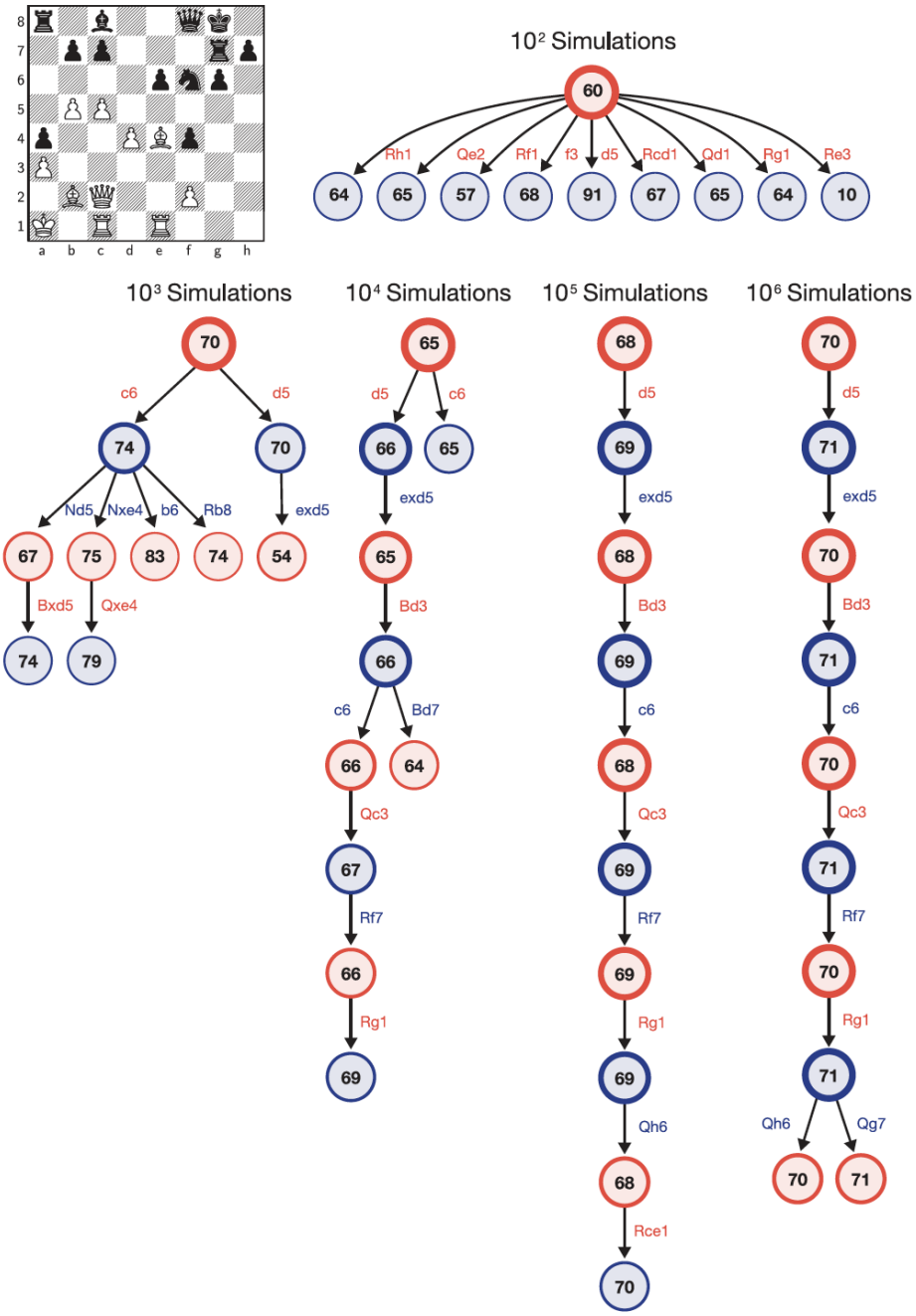
4.2.3 AlphaZero 对棋类知识的掌握

作者分析了AlphaZero发现的一些棋类知识。图7展示了国际象棋中最常见的12种开局和将棋中最常见的4种开局，这些开局都是AlphaZero独立发现并在自我对弈中常用的。同时，从每种开局开始，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谱。



（图7：从人类最常见的开局开始，AlphaZero 与 Stockfish 对战（A: 国际象棋）以及与 Elmo 对战（B: 将棋）的胜负与占比情况。 其中每个棋面右边的条形图表示胜负结果，在左栏中，AlphaZero 从给定的棋面开始执白； 在右栏中，AlphaZero 从给定的棋面开始执黑。胜负结果都是从 AlphaZero 的视角来评定：获胜（绿色）、平局（灰色）或失败（红色）。条形图右边的折线图表示 AlphaZero 在自我博弈中选择此开局所占的百分比随着训练进行的变化情况。）

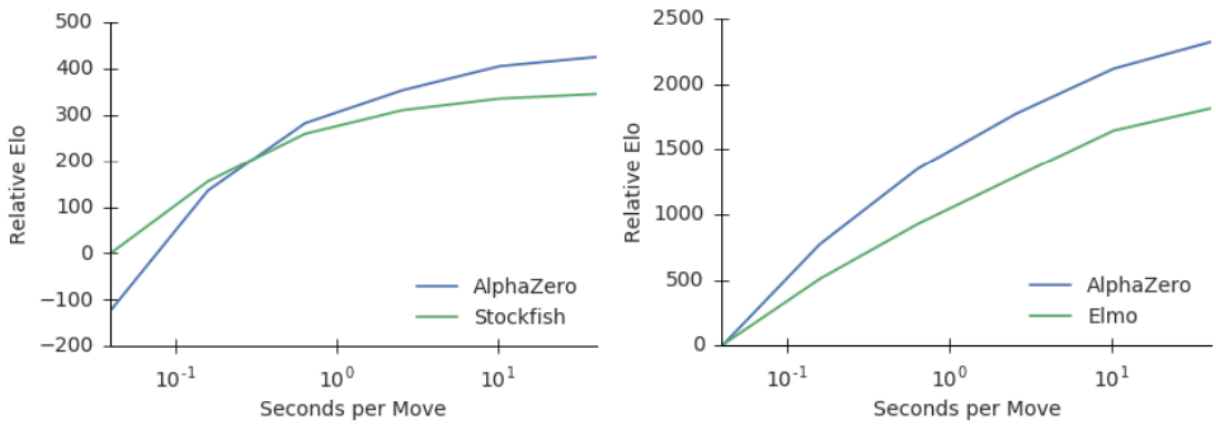
4.2.4 AlphaZero 的 MCTS 搜索过程



(图 8：AlphaZero 的搜索过程分析。 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 ）在执行29步 Qf8 之后的状态，如左上角的棋面所示。 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]。 每个内部状态对应的圆圈边界的厚度与其访问计数成正比。 AlphaZero 在执行第30步时，在模拟次数较小时考虑了动作c6，但最终随着模拟次数增加，最终选择了动作d5。)

4.2.5 AlphaZero 的性能扩展能力

- 图9展示了各种算法在思考时间延长情况下的性能扩展性，评估标准仍以Elo评分为准。令人惊讶的是，与人们通常认为的Alpha-Beta搜索在扩展性方面更具优势的观点相反，AlphaZero的蒙特卡洛树搜索（MCTS）在思考时间增长时的性能提升超过了Stockfish和Elmo，这两者的思考时间均设为40毫秒。
- Alpha-Beta剪枝搜索之所以强大，是因为它在确保找到最优解的同时，通过剪枝策略显著缩小了搜索空间，从而提高了搜索效率。然而，AlphaZero采用MCTS作为其主要搜索策略。MCTS通过处理不确定性和有效分配资源，实现了更高效的搜索。结合神经网络的评估函数，AlphaZero能够更专注于有更好前景的棋步选择，因此可以在更少的搜索步骤中找到优质的解决方案。
- 值得注意的是，AlphaZero是从零开始，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有能力发现人类可能忽略的策略，并在棋局中获取竞争优势。



（图9：左图展示了 AlphaZero 和 Stockfish 在国际象棋上的性能与每步思考时间的关系对比；右图则展示了AlphaZero 和 Elmo 在将棋上的性能与每步思考时间的关系对比。这两图进一步证实了AlphaZero在思考时间扩展性方面的优势。）

5. 算法实践

- 在充分挖掘 MCTS 系列算法技术的巨大潜力的同时，也提升这些技术在各种决策智能领域的易用性和实用性，上海人工智能实验室开源决策智能平台（OpenDILab）团队诚邀您共同探索 LightZero 项目——我们的目标是搭建一座集 MCTS 技术大成的桥梁。参见 LightZero：以 MCTS 为帆，航向决策 AI 的星辰大海 以获取更多详细信息。
- LightZero 不仅为众多决策问题提供了全面的基准工具，我们还在仓库中提供了 Gomoku/Go 的 AlphaZero 的 python 版和 cpp 版本的实现，您可以在这里找到入口文件。我们欢迎所有对此感兴趣的朋友们加入实践，激发讨论并贡献你们的智慧。
- 为了帮助您更好地理解和使用 LightZero，我们在 LightZero 开源贡献小贴士 中提供了详细的上手指南、协作规范和未来开发计划。欢迎参阅并提出宝贵的建议。演化即无限，让我们一起在 LightZero 的旅程中探索未知、创造未来！

6. 总结与展望

AlphaZero是一种将深度强化学习与蒙特卡洛树搜索相结合的人工智能算法。它能在仅知道游戏规则，无需其他先验知识的情况下自我学习，并在许多场合取得超越人类玩家的成绩。

以下是对 AlphaZero 的主要特性进行的总结：

- 强大的自我学习能力：AlphaZero能在没有先验知识的情况下自我学习。只需给出游戏规则和奖励信号，它就能逐步提升自我表现。
- 多样化的应用场景：AlphaZero不仅适用于围棋等棋类游戏，还能用于国际象棋、将棋等，显示出了广泛的适用性。
- 算法优化：AlphaZero在深度强化学习算法的各个方面，包括神经网络设计、搜索算法、训练方法等进行了优化，提升了算法的效率和性能。

对于 AlphaZero 的未来展望：

- 更广泛的应用：未来可以探究如何将AlphaZero扩展到更广泛的应用领域，比如自动驾驶、医疗保健、金融等。
- 进一步的算法优化：尽管 AlphaZero 在深度强化学习领域取得了显著的进步，但其算法仍有改进的空间，例如如何更好地处理稀疏奖励问题、如何更好地处理多智能体环境等。
- 提高算法解释性：由于深度强化学习算法的复杂性，其结果和决策过程往往难以解释和理解。未来的研究可以探索如何提高算法的可解释性，使其结果更易于被人理解和接受。

总的来说，AlphaZero是深度强化学习领域的一项重大突破，它的成功展示了深度学习和人工智能在游戏以及其他领域的广泛应用前景。随着算法的不断改进和扩展，AlphaZero 将在未来持续发挥其重要的作用，并为人类带来更多的启示。

7. 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.

[10] Rosin, Christopher D. "Multi-armed bandits with episode context."*Annals of Mathematics and Artificial Intelligence*61.3 (2011): 203-230.