MCTS + RL 系列技术科普和研讨博客（2）：MuZero

0. 引言

- 在 MuZero 的名字中的 "Mu"，可以理解为有“无”或“梦”的含义。MuZero 算法的核心思想是，在不了解环境动态转移规则的情况下，在“梦境”或“想象的空间”中进行搜索。具体来说，它构造了一个抽象的马尔可夫决策过程（abstract MDP）模型，并在该模型上预测与规划直接相关的未来数据（策略、价值函数以及奖励），然后基于这些预测数据进行规划。简洁地说，MuZero 首先在 latent 状态空间中学习出一个环境模型，然后基于这个学习到的环境模型，在不与真实环境进行交互的情况下进行规划。

1. 概述

- 构建具有规划能力的智能体 (Agents With Planning Capabilities) 一直以来都是人工智能领域的主要挑战之一。在诸如国际象棋和围棋等具有挑战性的（且具有完善的模拟器（perfect simulator））领域，基于树的搜索方法 (Tree-based Search) 取得了巨大的成功，然而与这些棋类环境不同，在现实世界中，往往缺少完善的模拟器并且环境动力学一般是复杂与未知的。
- 在本篇文章中，我们将介绍 MuZero 算法，该算法**在不知道环境动力学模型 (Environment’s Dynamics)** 的情况下，通过将**基于树的搜索**与**学习模型 (learned model)** 相结合，在一系列挑战性的输入为复杂视觉图像的领域中实现了超越人类的性能。
- 具体地，MuZero 智能体通过学习一个抽象的动力学模型 (latent dynamics model)，通过预测（predict）与决策直接相关的未来信息（策略、价值以及奖励函数）来学习这个模型，然后借助这个模型进行规划。当在57个不同的 Atari 游戏（用于测试人工智能技术的经典视频游戏环境，并且基于模型的规划方法在此类环境上一直表现不佳）上进行评估时，达到了当时的最佳性能。当在围棋、国际象棋和将棋游戏上进行评估时，在未获取任何游戏规则知识的情况下，MuZero 实现了与依赖游戏规则的 AlphaZero 算法一样的超人性能。

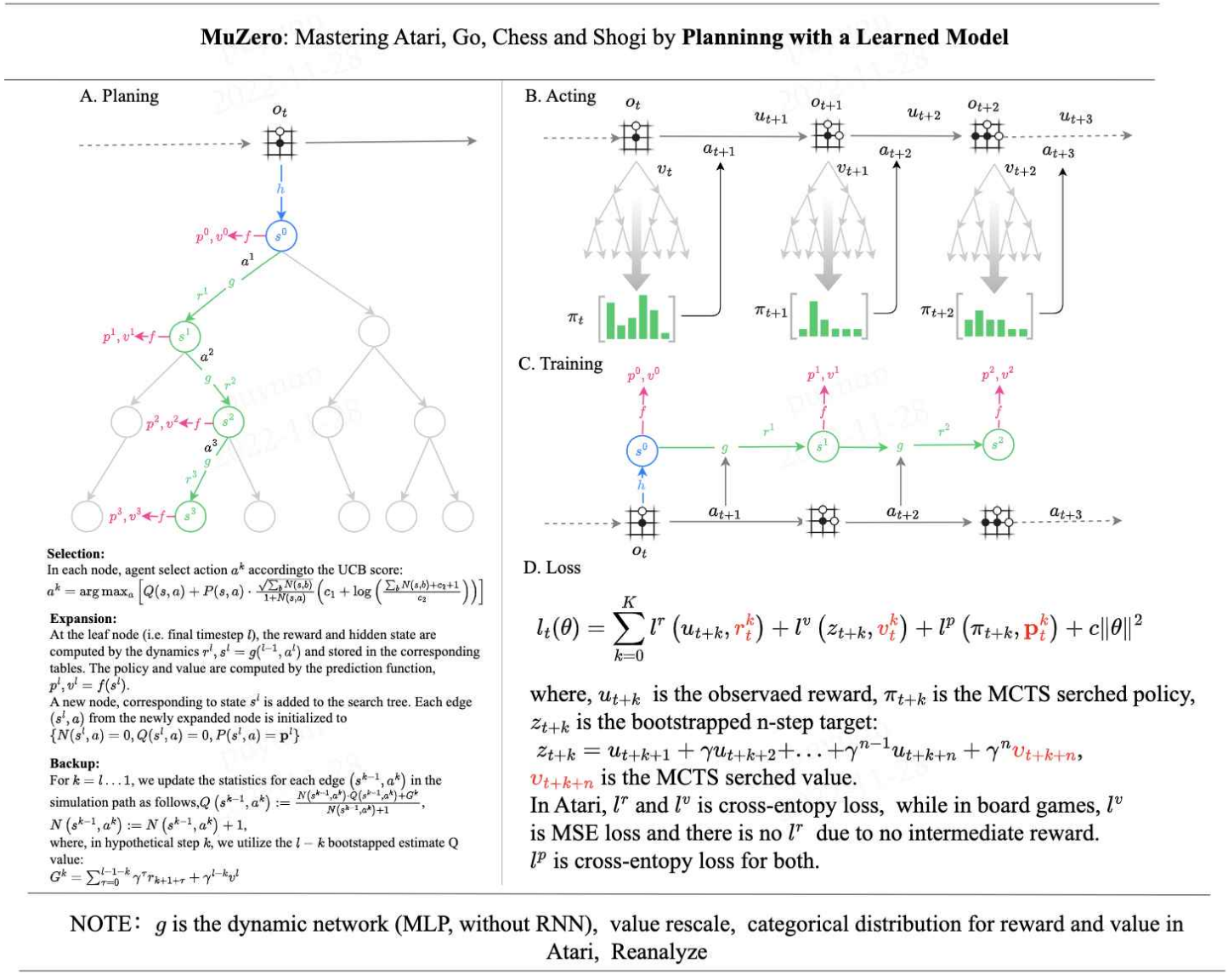
2. 研究背景与相关工作

- 一般的规划算法都依赖于环境动态转移规则，而在现实应用场景中智能体通常无法完整获取此类信息。基于模型规划的强化学习方法（Model-Based RL）通过学习一个环境动态模型来解决这个问题。经典的 Model-Based RL 算法有：Dyna，MVE，PILCO 等，详细的整理可见 Awesome Model-Based Reinforcement Learning。而无模型强化学习 (Model-Free RL) 恰好相反，无模型方法不需要先学习环境的模型，而是通过不断试错直接学习从状态到动作的映射关系。经典的算法有 Deep Q-Network (DQN)、Proximal Policy Optimization (PPO) 等。

- 环境模型 (Model of The Environment) 是强化学习系统的重要元素之一，它被用来模仿环境行为，或者更普遍的说，让人们可以推断环境的行为。传统上，该模型由马尔可夫决策过程 (Markov-decision process，MDP) 表示，给定当前时刻状态和动作，环境模型可以预测奖励和下一时刻状态。一旦构建好了模型，就可以直接应用 MDP 规划算法，例如值迭代或蒙特卡洛树搜索 (MCTS)，来计算 MDP 的最优值或最优策略。在大型或部分可观察的环境中，算法首先必须构建好模型将要预测的状态表征（State Representation）。*表征学习 (Representation Learning)*、*模型学习 (Model Learning)* *和规划 (Planning) 三者之间的分离*可能会存在问题，因为智能体可能不是以有效规划的目的来优化其表征或模型，例如，模型误差可能会在规划期间加重。
- Model-Based RL 的一种常见策略是直接在像素层面建模观测流。有研究假设，深度随机模型可能能够缓解错误累积的问题 [2] [3]。然而，对于大型问题，像素级别的精细规划在计算上并不实际。另一些方法则构建了一个能够在像素级别重建观测流的隐状态空间模型，或者预测其未来的隐状态 [4]，这虽然有助于提高规划效率，但可能会**过度关注无关的细节**。所有这些先前的方法都未能成功构建出能在视觉复杂领域（如Atari游戏）中进行有效规划的模型，即使在数据效率方面，其表现也落后于经过精细调优的无模型方法。
- 近期，有一种全新的 Model-Based RL 方法开始出现，其主要目标是直接预测价值函数 [5]。这种方法的核心思想是**构建一个抽象的马尔可夫决策过程（abstract MDP）模型，使得在此抽象 MDP 模型中进行规划，等同于在真实环境中进行规划**。为了实现这种等价性，需要保证*价值等价（value equivalence）*，即，从同一实际状态出发，通过抽象 MDP 模型得到的轨迹累积奖励与在实际环境中得到的轨迹累积奖励相一致。
  - Predictron [5] 首次引入了用于预测价值（无动作）的价值等价模型 (**Value equivalent models**)。尽管其底层模型仍为 MDP 形式，但并无要求其转移模型与环境中的真实状态匹配。反之，这个 MDP 模型被视作深度神经网络的一个隐含层。这个展开的MDP 被训练以使得奖励的预期累积和与真实环境中的预期价值相匹配，例如，通过时间差分学习算法实现。
  - 后续，Value equivalent models 被扩展到包含动作的价值优化。TreeQN [6] 学习了一个 abstract MDP 模型，使得该模型上的树型搜索（通过树形神经网络表示）能近似最优价值函数。**Value iteration networks** [7] 学习了一个局部 MDP 模型，使得在该模型上进行的值迭代（通过卷积神经网络表示）能近似最优价值函数。**Value prediction networks** [8] 或许是与 MuZero 最接近的先驱：它们学习了一个以实际动作为基础的MDP模型；在实际动作序列的条件下，通过训练展开的 MDP 以使得奖励累积和与真实环境相匹配。不过，与 MuZero 不同，它并没有进行策略预测，搜索过程仅依赖价值预测。

3. MuZero 算法详解

3.1 算法概览图



(图1：MuZero 算法概览图。**A**：MuZero agent 利用 MCTS 进行规划： representation network h 将连续 t 帧观测 (observation) $\{o_1, ..., o_t\}$ 编码为 latent state s^0 ； dynamics network g 预测在 s^{k-1} 下执行 action a^k 会转移到的 latent state s^k 和 reward r^k ； prediction network f 在 s^k 下输出 policy p^k (选每个 action 的概率)和 value v^k 。**B**：MuZero 如何与环境交互：对于图中 o_t 而言，使用蒙特卡洛树搜索得到一个策略，根据此策略进行采样，选出可执行动作，动作被选择的概率与 MCTS 根结点的每个动作的 visit count 成正比。在执行动作之后，环境产生真实奖励 u_{t+1} ，得到下一时刻的观测 o_{t+1} ，就这样不断与环境交互，在 episode 结束时，轨迹数据被存储到回放缓冲区（replay buffer）中。**C**：MuZero 在训练时需要用 dynamics network 展开 K 步，通过预测每一步的 reward, value, policy 来学习模型。**D**: 具体的损失函数定义。符号的含义参考算法概览图符号表。)

3.2 MuZero 模型

- AlphaZero 依靠 MCTS 做决策，MCTS 是在**真实的环境模拟器**中进行的，搜索的过程中需要策略网络和价值网络的辅助。
- MuZero 同样依靠 MCTS 做决策， 不过是在**学习到的抽象状态空间**中进行的，搜索的过程中需要如下3个网络的辅助：

- 表征网络 (Representation Network)
  - $s^0 = h_\theta(o_1, ..., o_t)$。将过去的观察情况编码为潜在状态（latent state，或称为 abstract state，有时也简称为 state）对应于时刻 t 的根节点，后面的预测都在潜在空间中进行。
- 动力学网络 (或称为机制网络) (Dynamics Network)
  - $r^k, s^k = g_\theta(s^{k-1}, a^k)$。即模拟环境动力学机制，给定状态和所选动作，给出相应的 reward 和转移到的下一时刻 latent state。
- 预测网络 (Prediction Network)
  - $\mathbf{p}^k, v^k = f_\theta(s^k)$。给定潜在状态，预测此潜在状态的策略和价值。
  - 概率分布 $\mathbf{p}^k$ 的优化目标为 MCTS 搜索得到的 π，通过交叉熵损失 $-\pi^T log(p)$ 进行优化。
  - 值函数 v^k 的优化目标为 TD target 值 z，通过 MSE 损失 $l = (v - z)^2$ 进行优化。

- 其中 prediction network 与 AlphaZero 中用到的策略网络和价值网络类似，也就是说，MuZero 在 AlphaZero 的基础上额外增加了 representation network 和 dynamics network。而增加的这两个网络即是为了让 MuZero 在学习到的抽象状态空间中进行搜索。
- 在下面我们将会依次探讨如下问题：
  - MuZero 所说的“在不知道规则的情况下，在抽象状态空间中进行搜索”的具体含义？
  - MuZero 中的 MCTS 过程有何特别，如何与环境交互？
  - MuZero 如何训练模型？

3.3 抽象状态空间

3.3.1 棋类环境规则

所谓 MuZero 在不知道规则的情况下，进行学习和搜索，是因为 MuZero 在深度神经网络和 MCTS 运行的过程中，没有使用游戏的如下规则：

- 规则1：终止状态和最终奖励的判断。
- 规则2：给出状态，动作，如何转移到下一个状态的规则（即环境动力学规则，Dynamics）。

        - 例如围棋游戏中的提子（在围棋中被对方围住的棋子会被提出棋盘）。
    - 规则3：给定一个状态，由谁落子，即需要采取动作的 agent是哪个玩家；给定一个状态，允许的落子，即合法动作集合。

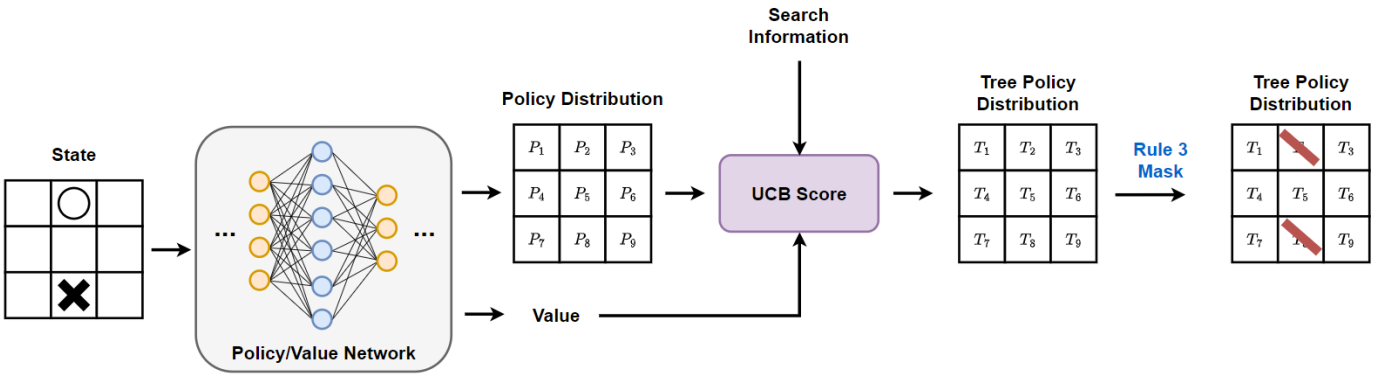
3.3.2 AlphaZero 是如何使用规则的：以井字棋为例

在井字棋中，假设当前棋面为 s_0 ，也即 MCTS 搜索的根节点为 s_0 ，在 MCTS 每次搜索的每次模拟时，会把当前状态输入给策略网络 π_θ ，得到9个概率值，然后通过如下公式得到最终9个动作的score：

$$score(a) = Q(s,a) + P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}[c_1 + \log(\frac{\sum_b N(s,b) + c_2 + 1}{c_2})]$$

其中 N 为访问次数、 Q 为 MCTS 模拟估计的平均价值、 p 为策略、 s 为状态。公式的第2项中分子是该节点总的访问次数的平方根，分母是子节点的访问次数，这样的设计会让访问比较少的子节点对应该项的值相对较大。

（TODO(pu): action mask 图）



(图2：AlphaZero 在决策时对规则3的使用。)

- 但是如果此时棋盘上本来就有两个棋子，即这2个位置对应的动作是非法的，就需要屏蔽这两个动作对应的概率值，然后进行 argmax 或采样得到实际执行的动作，这里就用到了上面的规则3。
- 选择 action 后，棋盘进入新的状态， $(s_t, a_t) \rightarrow s_{t+1}$ ，这里就用到了规则2。
- 在搜索的时候如果遇到游戏结束，到达终止状态（需要判断是否是终止状态），此时不会用神经网络来估计此节点的价值，而是直接用真实的输赢 $v = \pm 1$ （需要给出奖励，在最终 Q 的计算时也融合了价值网络的输出），这里用到了规则1。

3.3.3 MuZero 对规则的使用情况

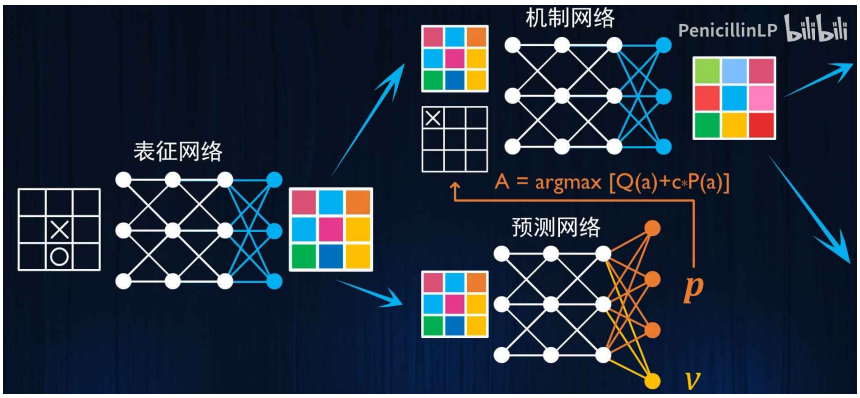
- 用 representation network: $s^0 = h_\theta(o_1, ..., o_t)$ ，将过去的真实棋盘状态转换为一个抽象的潜在状态（Latent State） s_0 。
- 以 s_0 作为搜索树的根节点，将这个潜在状态而不是真实局面输入到 prediction network: $\mathbf{p}^k, v^k = f_\theta(s^k)$ ，**但是在根节点处 MuZero 是用到规则3的**，也就是哪些地方不能下子（已经有子），先 mask 掉非法动作概率，再根据 ucb_score 选择动作。

- MuZero 不使用规则2，但是使用机制网络 dynamics network: r^k, $s^k = g_\theta\left(s^{k-1}, a^k\right)$ 进行模拟，执行动作后转移到下一个潜在状态。
- 潜在状态空间中不存在最终状态，且潜在状态下任何动作都是合法的，所以 MuZero 不使用规则1。
- 根节点往后的搜索中，因为潜在状态和真实状态无关，所以不知道哪些动作合法或者游戏是否结束，可以一直搜索下去，此时 MuZero 是不使用规则3的。

3.4 MuZero 中的 MCTS 过程

和 AlphaZero 类似，MuZero 也是通过从 MCTS 得到的 visit count 的分布进行采样得到实际与环境交互的动作。

只是在 MCTS 的时候，MuZero 首先需要将当前状态通过 representation 网络映射到潜在状态空间，然后利用dynamic 网络与 prediction 网络进行搜索，这个映搜索的过程如下图所示：



(图3：MuZero 利用 representation 网络，dynamic 网络与 prediction 网络在潜在空间进行搜索的示意图。[9])

假设此时 MuZero 的三个模型（表征网络，动力学网络，预测网络）已经训练好，初始节点 s_0 已经由原始棋盘状态 通过表征网络生成： $s_0 = h_\theta\left(o_1, ..., o_t\right)$。每条边保存的信息为：$N(s,a), P(s,a), Q(s,a), R(s,a), S(s,a)$，MCTS 在抽象状态空间中搜索的过程可以分为以下3个阶段：

❤️ 1. 选择 (selection)

- 在每次模拟的每一个 hypothetical time-step $k = 1, ..., l$，结合价值函数，策略函数与节点访问次数，选择 UCB score 最高的动作：

$$a^k = \arg\max_a \{Q(s,a) + P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}[c_1 + \log\left(\frac{\sum_b N(s,b)+c_2+1}{c_2}\right)]\}$$

- 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 和状态转移 S。
- $s^k = S(s^{k-1}, a^k)$, $r^k = R(s^{k-1}, a^k)$ 为根据 dynamic 网络记录得到的 state transition table 和 reward table。

2. 扩展 (expansion) 与 评估 (evaluation)
    - 根据上面选择的动作，拓展至新的状态节点，dynamic 网络会预测下一个状态和奖励。
    - 直到时刻 l 到达叶子节点，通过 dynamic 网络预测得到 r^l, s^l ，将其保存到 $R(s^{l-1}, a^l)$, $S(s^{l-1}, a^l)$ 表里，通过 prediction 网络预测得到 p^l, v^l ，用于 UCB score 的计算。
3. 回溯/反向传播 (backpropogation)
    - 更新搜索轨迹所在支路的 Q 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，而它的价值则会更新为其所有子节点的平均价值。

$$Q(s^{k-1}, a^k) := \frac{N(s^{k-1},a^k) \times Q(s^{k-1},a^k) + G^k}{N(s^{k-1},a^k)+1},$$

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

    - 其中 G^k 为 $l-k$ 步基于 v^l 对累计奖励的 bootstrapped 估计:
        - $G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l$
    - 由于在一些环境中，奖励的幅度变化很大，因此需要将奖励以及价值估计标准化到 [0,1] 区间:
        - $\bar{Q}(s^{k-1}, a^k) = \dfrac{Q(s^{k-1}, a^k) - \min\limits_{s,a \in \text{Tree}} Q(s,a)}{\max\limits_{s,a \in \text{Tree}} Q(s,a) - \min\limits_{s,a \in \text{Tree}} Q(s,a)}.$

- 执行：上述过程会重复多次（在 MuZero 中，每步棋会进行约800次迭代），在 MCTS 搜索完成后，在根节点 s_0 返回 visit counts 集合 $\{N(s,a)\}$ ，这些 visit counts 经过归一化后便得到了相对上一次训练有提升的策略 (improved policy) $\mathcal{I}_\pi(a|s) = N(s,a) / \sum_b N(s,b)$ 。然后从这个分布中采样得到实际与环境交互的动作。

3.5 模型训练

3.5.1 MuZero 模型训练的损失函数

- 该模型的所有参数都是联合训练的，以准确地将每一个 hypothetical 步骤 k 上的 的 policy 、value 和 reward，与 k 个实际时间步后观察到的相应目标值相匹配。但是，与传统的基于模型的强化学习方法不同，MuZero 中的潜在状态 s_k 没有附加环境状态的语义，它只是模型的潜在状态，其唯一目的是准确预测未来的相关的信息：policy、value、reward。

- 由于 MuZero 是在抽象 MDP 空间中搜索，为保证抽象 MDP 中的 planning 与真实环境中的 planning 等价，通过确保***价值等价 (Value Equivalence) 准则***来实现。即从相同的真实状态开始，通过抽象 MDP 的轨迹的**累积return** 与真实环境中轨迹的**累积 return** 相匹配。于是对于 MuZero 有以下3个优化目标。
  - 目标一：与 AlphaZero 类似，目标改进策略是通过 MCTS 搜索产生的，为了进行策略提升，需要最小化预测策略 p_t^k 和 MCTS 搜索得到的改进策略 π_{t+k} 之间的误差 $l^{\mathrm{p}}(\pi_{t+k}, p_t^k)$。
  - 目标二：与在 AlphaZero 中目标 value 使用完整序列的平均动作价值不同，在 MuZero 中融合带衰减因子的中间奖励和 MCTS 搜索得到的估计价值来更新目标 value，相当于使用了 $TD(n)$，速度更快、方差更小。
    - n-step bootstrapped 目标 value 定义为：$z_t = u_{t+1} + \gamma\, u_{t+2} + \ldots + \gamma^{n-1} u_{t+n} + \gamma^n \mathrm{v}_{\mathrm{t+n}}$。其中 u_{t+i} 表示观测奖励、v_{t+n} 表示 **MCTS 搜索得到的估计价值**。然后最小化预测网络的预测价值 z_{t+k} 和 目标 value 的误差：$l^{\mathrm{v}}(z_{t+k}, v_t^k)$
  - 目标三：最小化预测的奖励和实际观察到的奖励之间的误差：$l^{\mathrm{r}}(u_{t+k}, r_t^k)$。

$$l^{\mathrm{p}}(\pi, p) = \pi^T \log p \tag{1}$$

$$l^{\mathrm{v}}(z, v_t) = \begin{cases} (z-v)^2 \text{ for games} \\ \phi(z)^T \log v \text{ for general MDPs} \end{cases} \tag{2}$$

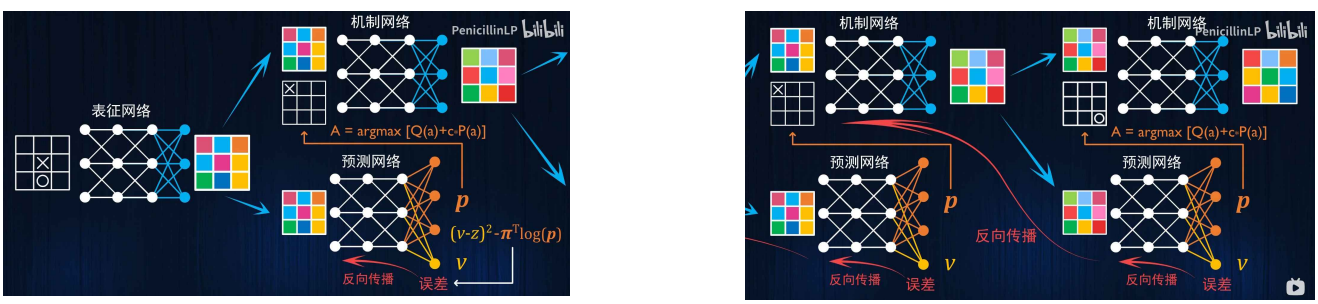
$$l^{\mathrm{r}}(u, r) = \begin{cases} 0 \text{ for games} \\ \phi(u)^T \log r \text{ for general MDPs} \end{cases} \tag{3}$$

综上所述，用于训练 MuZero 的3个网络的总损失函数如下:

$$l_t(\theta) = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p_t^k) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v_t^k) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r_t^k) + c||\theta||^2$$

3.5.2 重要细节

训练所需序列最小长度



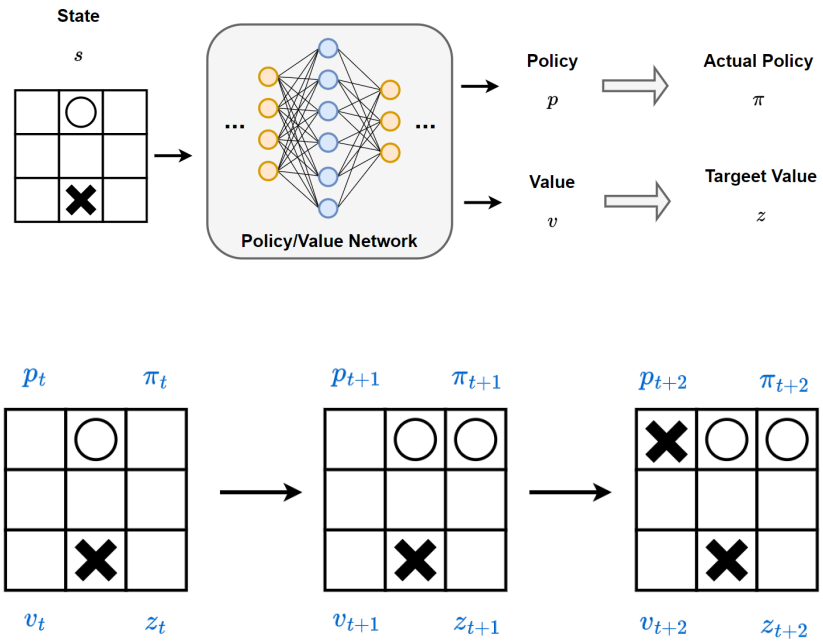
(图4：（左）MuZero 中误差由 prediction network 反向传播到 representation network 的示意图。（右）MuZero 中误差由 prediction network 反向传播到 dynamics network 的示意图。[9])

根据前面所述的潜在空间中的搜索过程，可以看到在时刻 t

- 对于 representation network $s^0 = h_\theta(o_1, ..., o_t)$，它的输出 s^0 是 prediction network 的输入，所以 representation network 可以直接由 prediction network 反向传播的梯度进行更新。
- 而对于 dynamics network $r^k, s^k = g_\theta(s^{k-1}, a^k)$，它的输出 s^k 将会是下一个 prediction network $\mathbf{p}^k, v^k = f_\theta(s^k)$ 的输入，所以下一个 prediction network 的梯度可以反向传播到 dynamics network。
- 所以至少要有两个状态及其相关信息，才能进行不同网络之间的梯度传播，即训练所需序列最小长度为2。

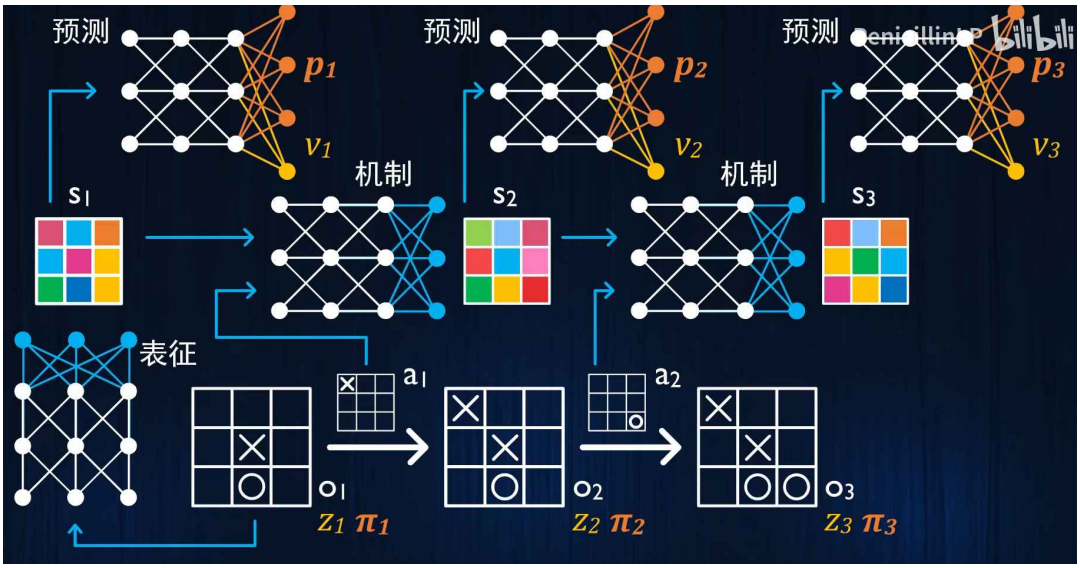
Replay buffer

- AlphaZero 存储的是每一个真实状态 o_t 及其 （MCTS 搜索得到的用于选择动作的）真实策略 π_t，策略网络产生的策略 p_t，价值网络预测的价值 v_t，和真实累计奖励 z_t 信息，即 $(o_t, p_t, \pi_t, v_t, z_t)$，不用存储实际选择的动作。



（图5： （上）AlphaZero 中每个状态 s 输入策略价值网络的输出以及优化目标。（下）AlphaZero 的 replay buffer 存储的每个状态 s 的信息）

- 和 AlphaZero 相比， MuZero 还存储了连续状态之间采取的实际动作 a_t、潜在状态 s_t，即 $< o_t, a_t, z_t, \pi_t, s_t, p_t, v_t >$，且如上面提及的 MuZero 训练所需序列最小长度是2，而实际 MuZero 用的序列长度是6，即 $< (o_t, a_t, z_t, \pi_t, s_t, p_t, v_t), ..., (o_{t+5}, a_{t+5}, z_{t+5}, \pi_{t+5}, s_{t+5}, p_{t+5}, v_{t+5}) >$ 作为一条训练数据存储在 replay buffer 中。



（图6： MuZero 的 replay buffer 存储的信息示意图。[9] ）

4. MuZero 实验

4.1 实验设置

4.1.1 实验环境

- 经典棋类游戏作为规划问题的基础验证环境：Go，chess，shogi。
- Atari 环境中的57个游戏作为视觉复杂RL领域的基础验证环境。

4.1.2 网络结构与主要超参数

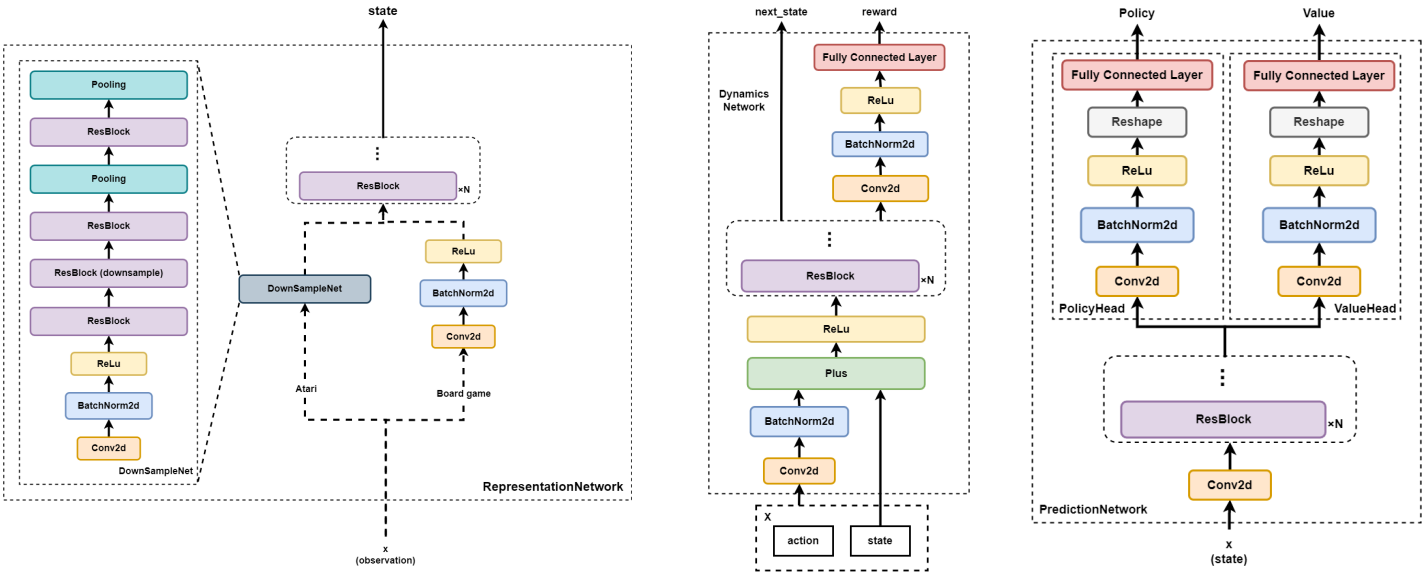
网络结构

MuZero 用到的三个网络的具体结构如图7所示。

- 在MuZero中，表示函数和动态函数采用的架构与AlphaZero相同，但是使用的ResBlock数量为16个，而非20个。每个卷积过程采用3x3的卷积核和256个隐藏层。
- representation network ： 对于Atari游戏，由于其观测数据具有较大的空间分辨率，我们首先通过一系列步长为2的卷积操作减小其空间分辨率。具体来说，我们从一个96x96分辨率、128个平面的输入观测开始（包含32个历史帧，每帧包含3个颜色通道，以及32个动作，这些动作被广播到各个平面上，于是32*3+32=128），然后采用以下策略进行降采样：
  - 使用步长为2和128个输出平面的卷积，输出分辨率为48x48。
  - 使用128个平面的2个 ResBlock。
  - 使用步长为2和256个输出平面的卷积，输出分辨率为24x24。
  - 使用256个平面的3个 ResBlock。
  - 使用步长为2的平均池化，输出分辨率为12x12。

- 使用256个平面的3个 ResBlock。
- 使用步长为2的平均池化，输出分辨率为6x6。
- 以上所有这些操作都使用3x3的卷积核。

- dynamics network： 使用与 representation network 类似的架构。它始终在 latent state 上进行操作。首先，将动作编码为图像，然后将其与前一个步骤的隐藏状态沿平面维度进行叠加。
- prediction network 使用与 AlphaZero 类似的架构，先是N个公用的 ResBlock，然后分为2个 head，分别输出策略和值。



(图7： 左：MuZero 的表征网络结构图。中：MuZero 的动力学网络结构图。右：MuZero 的预测网络结构图。注：本文档网络结构图均按照 LightZero 实际实现，与原论文有细微区别。)

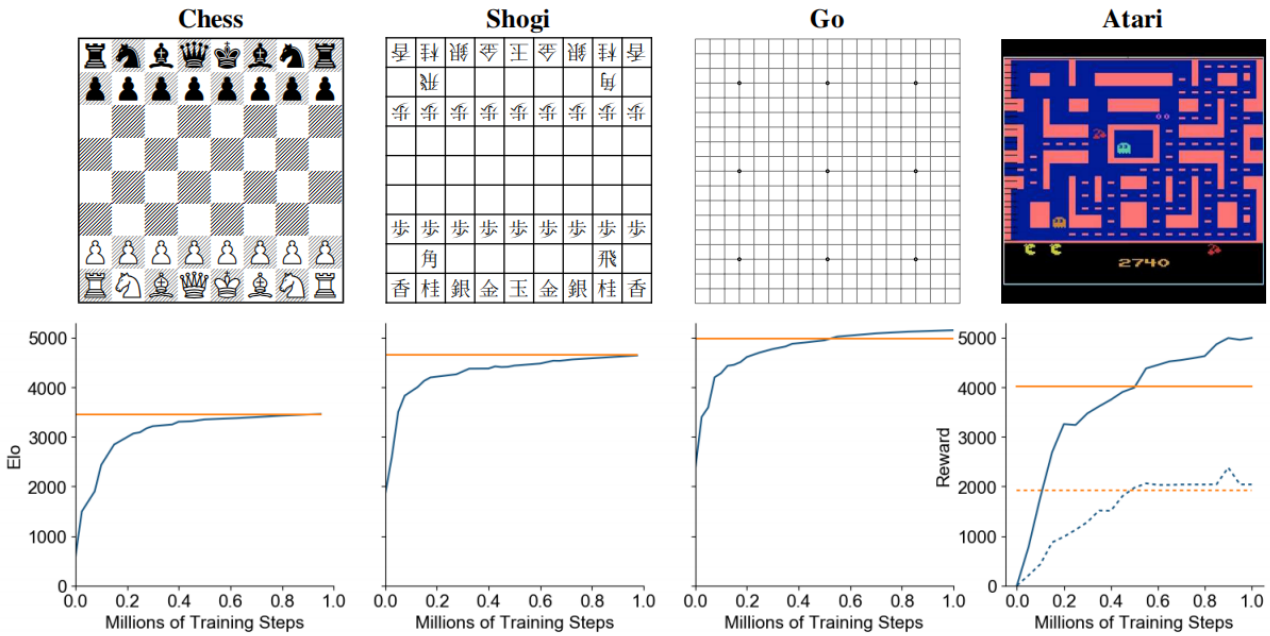
主要超参数

> - hypothetical steps：在每种环境下，作者训练 MuZero 的 hypothetical steps K = 5 ，即通过 dynamics network 从当前状态向下推导5步，在每一步预测转移到的 latent state 和相应 reward， 并根据 prediction network 得到预测的 policy 和 value，在这5步上计算相应的 loss 更新网络。

参数	hypothetical steps	discount	mini-batches size	simulations for each search	unroll steps num	weight_decay
棋类游戏	5	0.97	2048	800	5	0.0001
Atari			1024	50		

(表1：MuZero 主要超参数。其他详细参数见原论文补充材料伪代码。)

4.2 实验结果



(图8: 在国际象棋、将棋、围棋和 Atari 上 MuZero 的学习曲线。)

4.2.1 经典棋类游戏

在图8中，展示了在国际象棋、将棋、围棋和 Atari 上 MuZero 的学习曲线。 蓝色线为 MuZero 的 Elo（或奖励）随着训练步数增加的变化。橘色线为 AlphaZero 的 Elo。由图8中前3列可以看出，在 Chess 和 Shigi 上 ，MuZero 达到了和 AlphaZero 类似的性能。而在 Go 中，MuZero 性能略好于 AlphaZero 。

4.2.2 Atari

图8中第4列中，纵轴 reward 表示游戏中的得分，虚线表示57个游戏上得分的中位数，实线表示57个游戏上得分的平均数。蓝色线为 MuZero 的得分随着训练步数的变化，橙色实线为当时的 SOTA R2D2（一种model-free RL方法）的得分。可以看到，在 Atari 中，MuZero 达到了当时 SOTA 的水平，并大大超出之前同类 model-based 方法中最好的 SimPLe （橙色虚线）。

4.2.3 Reanalyze

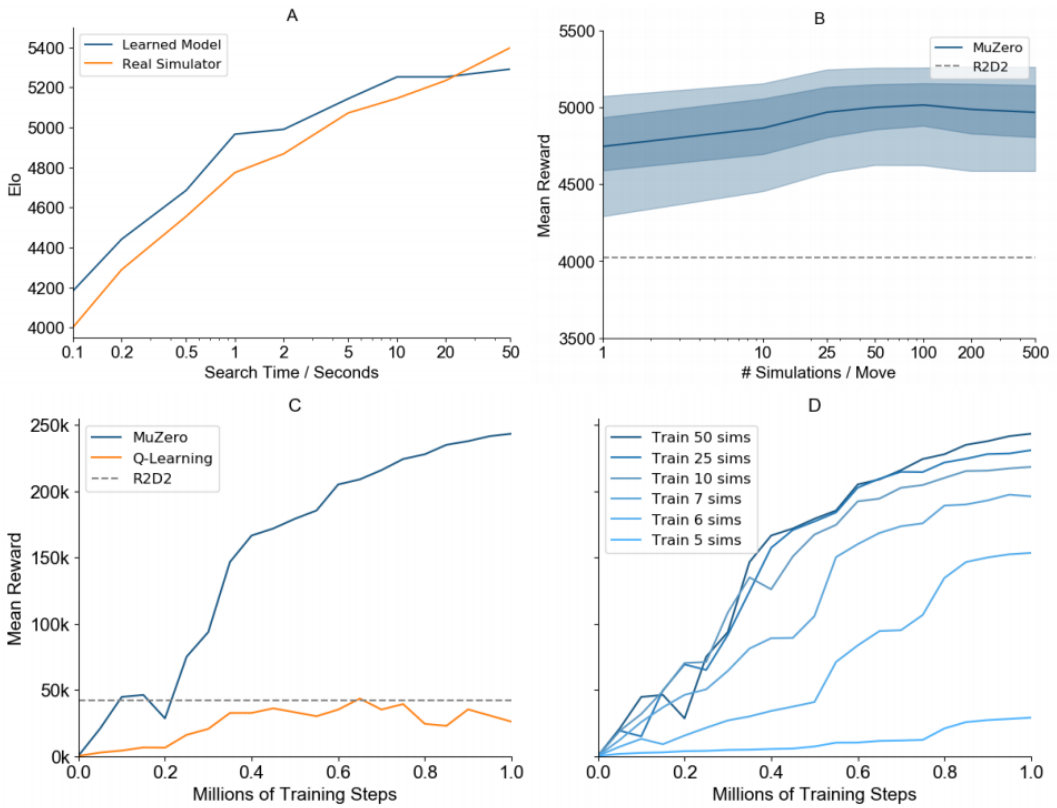
简单来说，Reanalyze 技术是指使用最新的网络在收集的数据上重新运行 MCTS 来**重新分析**旧的轨迹，得到新的更加准确的 target policy/value 用于训练。从表2可以看出，在 Atari 上，MuZero Reanalyze 优于之前所有的 model-free RL 方法。

Agent	Median	Mean	Env. Frames	Training Time	Training Steps
Ape-X [18]	434.1%	1695.6%	22.8B	5 days	8.64M
R2D2 [21]	1920.6%	4024.9%	37.5B	5 days	2.16M
MuZero	**2041.1%**	**4999.2%**	20.0B	12 hours	1M
IMPALA [9]	191.8%	957.6%	200M	–	–
Rainbow [17]	231.1%	–	200M	10 days	–
UNREAL[a] [19]	250%[a]	880%[a]	250M	–	–
LASER [36]	431%	–	200M	–	–
MuZero Reanalyze	**731.1%**	**2168.9%**	200M	12 hours	1M

(表2：在大数据量（上）和小数据量（下）设置下，MuZero 与之前 model-free RL 算法在 Atari 游戏上的性能比较。除 MuZero 外，其他所有智能体都使用了 model-free RL 技术。表中给出了平均分，中位数分，以及训练用到的环境帧数，训练时间和训练步数。最佳结果以粗体字突出显示。在这两种设置上，MuZero 都达到了新的最高性能。)

4.2.4 潜在环境模型的作用

为了探究 MuZero 学习到的潜在环境模型的作用，在棋类游戏 Go 以及 Atari 游戏中的 Ms. Pacman 游戏环境上做了如下系列探究实验。



(图9：MuZero 在 Go 上的探究实验 (A)，在57个 Atari 游戏上的探究实验 (B)，和在 Ms. Pacman 的探究实验 (C-D)。)

围棋环境下规划的可扩展性 (Scalability)

- 如图9 (A) 所示，对比了已经训练好的使用完美环境模型的 AlphaZero ，和已经训练好的学习环境模型的MuZero（注意两个网络都在每次搜索800次模拟的情况下进行训练，相当于每次搜索0.1秒），在评估时使用不同搜索时间下的性能。可以看到在相同搜索时间下 MuZero 的性能甚至比使用完美环境模型的 AlphaZero 略好。
- 值得注意的是，在评估时的单步搜索时间比训练时多两个数量级时，使用学习环境模型的 MuZero 依然表现良好。

Atari 环境下规划的可扩展性

- 如图9 (B) 所示，使用训练好的 MuZero（在每次搜索进行50次模拟的情况下进行训练），但是在评估时 MCTS 时（每次move）使用不同的 simulations 次数，观察 MuZero 平均奖励的变化情况，以 R2D2 为基线。深色线表示平均分数，阴影区域表示25到75和5到95的百分位数。
- 在每次搜索的模拟次数小于100时，MuZero 的性能随着模拟次数的增加而增加。而且注意到，MuZero 只进行一次 simulations（相当于仅依靠策略网络）效果依然不错，说明策略网络在训练中已经吸收了搜索学到的经验（learned to internalise the benefits of search）
- 此外，即使在评估时使用比训练期间的搜索次数大得多的情况下（ simulations=500），学习模型的性能仍然稳定，只是稍微下降。这与围棋中更好的扩展性形成鲜明对比，可能是由于相比围棋，在 Atari 环境上学习的模型不准确性更大。

Model-based 方法与 Model-free 比较

- 将 MuZero 的 Loss 替换为 R2D2 中 Q-learning 的损失函数、策略价值双头网络替换为单头的 Q 值网络，训练和评估时都不使用 MCTS，得到一个 'Model-free' 版 MuZero，即图9 (C) 中的 Q-learning 线所示，它与原版MuZero 进行比较的曲线如图9 (C) 所示。
- 可以看到 MuZero 的 Q-Learing 版本实现达到了与 R2D2 相同的最终分数，但与基于 MCTS 的训练相比，提高得更慢，最终性能也要差得多。这可能是由于 MuZero 基于搜索的策略改进（search-based policy improvement）想对 Q 学习所使用的高偏差、高方差目标提供了更强的学习信号。

simulation 次数的影响

- 如图9 (D) 所示，探究了在 Ms. Pacman环境中，训练时采用不同 simulation 次数对性能带来的影响。注意是训练时收集数据用到的 simulation 次数不同，但在评估时 simulation 次数都等于50。
- 明显看到当训练时的 simulation 次数增加时，策略的改进速度更快。另外可以观察到 MuZero 在 simulation 次数小于合法动作数量时仍然可以有效地学习，比如 Ms. Pacman 环境中可选择的动作为 8 个，simulation 次数为 6 时算法性能依然很好。

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）团队诚邀广大开发者共同探索 LightZero 项目。更多信息可以参见 GitHub 仓库和技术博客 LightZero：以 MCTS 为帆，航向决策 AI 的星辰大海 。
- https://github.com/opendilab/LightZerogithub.com/opendilab/LightZero
- LightZero 为众多复杂的决策问题提供了全面的基准工具，针对 MuZero，LightZero 在仓库中提供了 Atari 的 MuZero 入口文件，其树搜索部分包含了 python 版本和 cpp 版本的实现，开发者可以循序渐进地接触相关算法和应用。我们欢迎所有对此感兴趣的朋友们加入实践，尽情讨论并贡献你们的智慧。
- 为了帮助开发者更好地理解和使用 LightZero，我们还在 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

人工智能的许多突破都是基于高性能规划或无模型强化学习方法。MuZero 结合了两种方法各自的优点，不仅在逻辑复杂的棋盘游戏，如国际象棋和围棋中与高性能规划算法的超人类水平的性能相匹配，而且在视觉复杂的 Atari 游戏中超过了最先进的无模型RL算法。至关重要的是，MuZero （几乎）不需要任何游戏规则或环境动力学的知识，这使得 MuZero 能够应用于很多缺乏完美模拟器的现实世界任务。

- 未来的研究方向包括：
  - 如何与 model-based RL 的最新进展结合以进一步提升算法的样本效率？
  - 如何进行 MCTS 的加速?
  - 如何让 MuZero 学习得到的表征空间更加具有语义和可解释性?

7. 参考文献

[1] Schrittwieser, J., Antonoglou, I., Hubert, T. *et al.* Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* **588**, 604–609 (2020).

[2] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson.

Learning latent dynamics for planning from pixels. arXiv preprint arXiv:1811.04551, 2018.

[3] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski,

Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning

for atari. arXiv preprint arXiv:1903.00374, 2019.

[4] David Ha and J¨urgen Schmidhuber. Recurrent world models facilitate policy evolution. In Proceedings of

the 32Nd International Conference on Neural Information Processing Systems, NIPS’ 18, pages 2455–2467,

USA, 2018. Curran Associates Inc.

[5] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-

Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and

planning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages

3191–3199. JMLR. org, 2017.

[6] Gregory Farquhar, Tim Rocktaeschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreec: Differentiable

tree planning for deep reinforcement learning. In International Conference on Learning Representations,

2018.

[7] Aviv Tamar, YiWu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In Advances

in Neural Information Processing Systems, pages 2154–2162, 2016.

[8] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In Advances in Neural Information

Processing Systems, pages 6118–6128, 2017.

[9] 不知道规则也能下围棋！？ 【恐怖如斯 MuZero】【下】_哔哩哔哩_bilibili, https://www.bilibili.com/video/BV1Ey4y1s7zB/?spm_id_from=333.337.search-card.all.click&vd_source=35dd2c5bd9fbf55c9681c89ce63d38c1 , 2021.