Stochastic MuZero

1. 概述

❤️
- 基于模型的强化学习已经被证明非常成功。然而，在复杂的环境中，在学习一个模型时，如果不考虑它在规划时的使用是有问题的。迄今为止，最有效的技术是将 value-equivalent 模型学习与强大的树搜索方法相结合起来。这种方法以 MuZero 为例，它在广泛的领域取得了最先进的性能，包括棋盘游戏和输入为视觉图像的环境，在线和离线设置，离散和连续的动作空间。
- 然而，这种方法以前的实例仅限于使用确定性模型 (deterministic models)。这限制了它们在**固有随机的、部分可观察的环境的应用（注：对于 finite agent 来说，庞大和复杂的环境也相当于具有随机属性）**。
- 在本文中，作者扩展了这种方法，使其使用**随机模型**进行学习和规划。具体来说，作者介绍了一种新的算法，Stochastic MuZero，它**学习一个包含 afterstates 的随机模型，并使用该模型执行随机树搜索**。Stochastic MuZero 在一组规范的单智能体和多智能体环境中匹配或超过了最先进的水平，包括2048和双陆棋，同时保持了标准 MuZero 在围棋中的超人性能。

2. 研究背景与相关工作

2.1 观测模型（Observation models）

通过拟合观测和奖励模型，作者能显式地学习环境动态，然后可以以Dyna的范式[2]和无模型的学习规则相结合。然而，这种方法在处理高维图像观测建模时可能存在计算挑战和累积误差，而且学习对模型无益的背景特征可能降低建模效果。

2.2 隐状态模型（latent models）

隐状态（latent states）下的递归网络模型学习[3][4]能够解决显式模型的局限性。在这个框架下，模型基于当前观测和未来动作条件化，并在k步内展开。然后，它根据当前的隐状态在每个时间步对奖励、价值、策略或观测进行预测。通过这些联合的预测损失函数来训练模型。最近MuZero [3]等方法已经利用此思路在许多任务上实现了高样本效率和高性能。 然而，大多数方法，例如 MuZero [3]，采用确定性函数进行环境建模，限制了它们在随机环境中的适用性。

2.3 随机隐模型 （Stochastic latent models）

环境的动态模型的随机性也可以被建模。

- DreamerV2 [5]提出了一个由循环模块（recurrent module）、表征模块（结合隐状态和当前观测）和转换预测器（仅依赖隐状态和作用于模型的先验）组成的循环状态空间模型（recurrent state-space model）。
  - $$\begin{aligned} &\text{Recurrent model:} && h_t = f_\phi\left(h_{t-1}, z_{t-1}, a_{t-1}\right) \\ &\text{Representation model:} && z_t \sim q_\phi\left(z_t \mid h_t, x_t\right) \\ &\text{Transition predictor:} && \hat{z}_t \sim p_\phi\left(\hat{z}_t \mid h_t\right) \end{aligned}$$
  - 该方法优于无模型方法，但性能不及 MuZero 等方法。
- VQ-MuZero [6] 使用VQ-VAE 生成网络学习随机转换模型，然后把它与MCTS结合，将问题转化为单人任务，并隐式地学习对手的行为。这种方法可以与MuZero在双人任务中达到相匹配的效果。但是，这种方法需要显式地学习观测的表示，这对于高维观测（如图像）不利。最后，作者采用了两阶段的训练过程：首先，模型学习观测的隐性表示，然后这些表征被用来学习转换模型。这使得在强化学习设置中应用这种方法变得困难。
- Stochastic MuZero 和 VQ-MuZero [6] 的区别
  a. VQM-MCTS是一个 two state training 的过程， Stochastic MuZero 是end to end的过程
  b. VQM-MCTS有真正的VQVAE整个的训练模块，即 encoder， table， decoder的训练，不过在真正使用的时候，只用到了encoder和table用来选择索引值，在后续中索引值起到主要作用；但是Stochastic Model更像是一个分类网络，并没有显式的训练vqvae
  c. VQM-MCTS无论是 action node 还是stochastic node都需要使用UCB公式，但是Stochastic MuZero只有decision node 需要UCB，而 chance node 则使用prior分布进行采样。

2.4 MuZero

MuZero 是一种 model-based 的强化学习算法，融合了学习环境动态模型与蒙特卡罗树搜索规划这2个思想。该模型输入是时间步 t 的观测历史 $o_{\leq t}$ 和未来动作序列 $a_{t:t+K}$ ，被训练来预测未来每个时间步的 searched policy $\pi_{t:t+K}$, value $v^{\pi}_{t:t+K}$ ，和 intermediate rewards $r_{t:t+K}$ 。
MuZero 使用确定性函数来构建其模型，因此它隐含地假设底层环境动态是确定性的。MuZero 在每个时间步使用学习到的动态模型进行规划，并根据其 MCTS 搜索结果来选择动作，并将 MCTS searched policy $\pi_{t:t+K}$ 作为策略提升算子（policy improvement operator）的目标。

- **模型**

  MuZero 的学习模型包括三个函数，即表征(representation)函数 h , 动力学(dynamics)函数 g 和预测(prediction)函数 f 。其中，表征函数 $s^0_t = h(o_{<=t})$ 将观测历史观测历史 $o_{\leq t}$ 映射到 latent state s^0_t ; 动力学函数 s^{k+1}_t, $r^{k+1}_t = g(s^k_t, a_{t+k})$, 输入为 latent state 和 action， 输出为下一个时刻的 latent state 和 预测 reward。预测函数 p^k_t, $v^k_t = f(s^k_t)$ ，输入为 latent state，输出为对应的 policy 和 value。用于训练 MuZero 的3个网络的总损失函数如下:

$$L^{MuZero} = \sum_{k=0}^{K} l^{\mathrm{p}}(\pi_{t+k}, p^k_t) + \sum_{k=0}^{K} l^{\mathrm{v}}(z_{t+k}, v^k_t) + \sum_{k=1}^{K} l^{\mathrm{r}}(u_{t+k}, r^k_t)$$

- **MCTS 搜索**

  MuZero 使用了一种基于MCTS树的变体算法，该树通过多次模拟（simulations）递归地构建。每次模拟包括三个阶段：选择、扩展和反向传播（selection, expansion and backpropagation）。

  - Selection: 从根节点（root node）开始遍历树，直到到达叶子边 (leaf edge)。在每个内部节点（internal node） s 上，算法选择最大化 upper confidence bound 的动作 a：

$$a = \arg\max_a [Q(s,a) + P(a|s) \cdot \frac{\sqrt{1+\sum_b N(s,b)}}{1+N(s,a)}(\alpha_1 + log(\frac{\sum_b N(s,b) + \alpha_2 + 1}{\alpha_2})$$

    - 其中， $Q(s,a)$ 代表了对于动作 a 的值估计； $N(s,a)$ 代表了访问次数； $P(a|s)$ 代表了访问动作 a 的先验概率；

  - Expansion: 通过查询 MuZero 模型，扩展叶子边 (leaf edge)并将新节点添加到树中。
  - Backpropagation: 新添加边（edge)的 n 步回报估计(N-step return estimate)沿着树反向传播。

2.5 VQVAE

Vector Quantised Variational AutoEncoder 是一种生成模型技术，它包含四个关键组件：编码器神经网络 e ，解码器神经网络 d ，向量量化层 vq 和自回归模型 m 。

- 编码器： $c_t^e = e(x_t)$ 输入 x_t ， 输出一个 embedding c_t^e 。
- 向量量化层： $c_t, k_t = vq(c_t^e)$ ，是一个由 M 个编码 $\{c_i\}_{i=0}^M$ 组成的 codebook。它根据编码器的输出，从codebook 里面选择一个距离最近的 quantised embeddings $c_t = c_t^k$ ，以及对应的索引 $k_t = \text{argmin}_i ||c_i - c_t^e||$ 。在梯度反向传播中，将此量化操作视为恒等函数，称为 straight-through estimator。
- 解码器： $\hat{x_t} = d(c_t)$ ，输入编码 c_t ，返回对 x_t 的重构向量 $\hat{x_t}$ 。
- 自回归模型： $p(k_t|c_{<t}) = m(c_{<t})$ ， 根据小于t时刻的 quantised embeddings，预测一个关于编码索引的分布 $p(k_t|c_{<t})$ 。

通常，首先训练编码器、解码器和 codebook，然后在额外的第二阶段（second stage）中将它们的参数固定，训练自回归模型。VQ-VAE 的总损失函数为：

$$L_\phi^{vqvae} = \sum_{t=0}^{N-1} [\underbrace{\|\hat{x_t} - x_t\|}_{\text{reconstruction}} + \underbrace{\beta\|c_t - c_t^e\|^2}_{\text{commitment}} - \underbrace{\gamma log\ p(k_t|c_{<t})}_{\text{second stage}}]$$

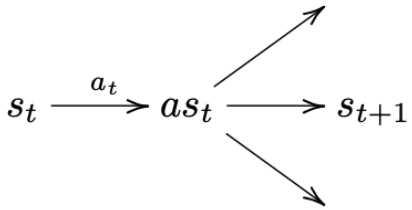
3. Stochastic MuZero

3.1 Stochastic Model

Afterstates

当环境的动力学模型是随机的时候，作者定义一个“执行动作后，在环境转换到真实状态之前的假想环境状态”，命名为 afterstates （An afterstate as_t is the hypothetical state of the environment after an action is

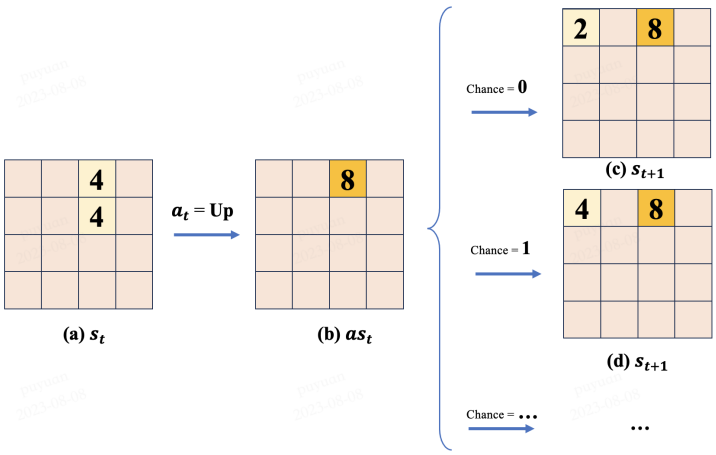
applied but before the environment has transitioned to a true state）。 在执行动作后转移到 afterstates 期间，这个过程是 deterministic 的； 在 afterstates 转移到下一个 state 之间，这个过程是 stochastic 的。



（图1: afterstates的示意图。）

通过使用 afterstates，作者可以将 applying an action to the environment 与 chance transition given an action 的影响区分开。

- 例如在2048游戏中，如图2(a)所示，当前 state s_t，下执行 'Up’ 动作，环境首先会把具有相同数字的相邻方块合并得到图2(b) 所示的 afterstate as_t ，然后随机地从任意一个空的位置生成方块 2 或者 4 （如图2(c)，(d)所示)。
- 在这个过程中，从 图2(a)到 图2(b)的过程是确定的， 没有随机性； 而从图2(b) 到 图2(c)或 图2(d)（或者其他的可能状态）是环境的随机性带来的。



（图2: 2048 游戏中，afterstates 的示意图。）

定义 afterstate 的值为 $V(as_t) = Q(s_t, a_t)$，环境 dynamics 的转移概率为 $Pr(s_{t+1}|as_t) = Pr(s_{t+1}|s_t, a_t)$。基于概率事件（chance event），一个 afterstate 可以转移到多个状态。作者假设环境在给定 afterstate 的情况下可以转移到有限数量的可能状态 M，这样可以将每个可能的转换与 chance outcome 关联起来。 chance outcome 的典型例子包括2048游戏中随机出现的数字的值及其位置 和 backgammon 游戏中的骰子结果。

通过定义 afterstates 和chance outcome，便可以以更灵活的方式建模随机环境动态。具体地，作者使用一个确定性模型 $s_{t+1}, r_{t+1} = M(as_t, c_t)$ 以及一个概率分布 $Pr(s_{t+1} \mid as_t) = Pr(c_t \mid as_t)$ 来建模 chance transition。 这样，学习 stochastic 模型就被简化为学习 afterstates 和 chance outcome，即公式中 as_t 和 c_t。

Model

Stochastic MuZero 共学习五个函数：

- Representation network

$$s_t^0 = h(o_{<=t})$$

  - 将当前时刻以前的观察 $o_{<=t}$ 映射到一个 latent state s_t^0。
- Prediction

$$p_t^k, v_t^k = f(s_t^k)$$

  - 根据 latent state 预测对应的策略分布和值。
- Afterstate Dynamics

$$as_t^k = \phi(s_t^k, a_{t+k})$$

  - 根据 latent state 和 动作，得到一个 afterstate。并不会输出 value 或者 reward。
- Afterstate Prediction

$$\sigma_t^k, Q_t^k = \psi(as_t^k)$$

  - 以 afterstate 作为输入，输出一个 先验 chance 分布 σ_t^k 以及一个 Q value。在搜索时，会从先验 chance 分布中采样出一个 chance outcome $c_{t+k+1} \sim \sigma_t^k$。
- Dynamics

$$s_t^{k+1}, r_t^{k+1} = g(as_t^k, c_{t+k+1})$$

  - 根据 afterstate 和 chance outcome，得到下一时刻的 latent state 和 reward 预测值。

Chance Outcome

Stochastic MuZero 模型使用 VQ-VAE 方法的一种变体来建模概率事件（chance outcomes）。具体来说，

- 它通过使用大小为 M 的固定 codebook 来简化 VQ-VAE。codebook 中的每个条目都是大小为 M 的固定的 one-hot 向量。在这种情况下，作者将编码器输出的大小为 M 的 embedding c_t^e 建模为 categorical variable，并选择最接近的 code c_t ， 这等价于计算表达式 $\text{onehot}(\text{argmax}_i\ c_t^{e,i})$ 。
- 这样得到的编码器可以视为 observation 的随机函数，它利用了 Gumbel softmax reparameterization trick，当 进行前向传递时 temprature 是0，在反向传播时相当于一个 straight through estimator。
- 没有明确的解码器（explicit decoder），也没有使用重构损失（reconstruction loss）。相反，该网络以类似于MuZero 的方式端到端训练。

Model Training

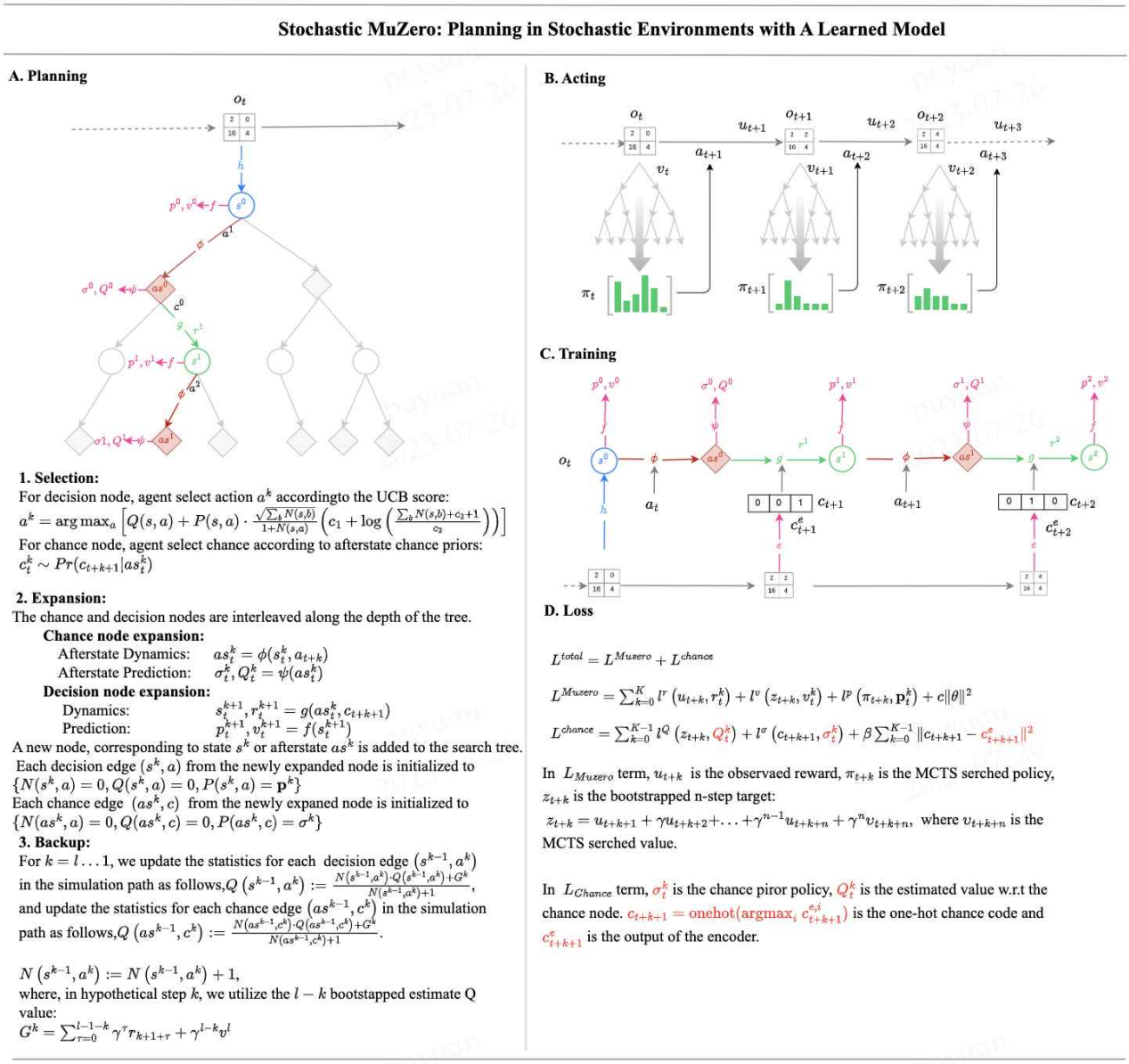
Stochastic MuZero 以类似于MuZero的端到端方式进行 unroll 和训练。具体而言，给定一个长度为 K 的轨迹，包括observations $o_{<=t:t+K}$ ，actions $a_{t:t+K}$ ，value targets $z_{t:t+K}$ ，policy targets $\pi_{t:t+K}$ 和 rewards $r_{t:t+K}$ 。该模型按照图3所示展开 K 步。优化loss为两个部分： MuZero loss 和 chance loss （用于学习随机模型):

$$L^{total} = L^{MuZero} + L^{chance}$$

MuZero Loss 的组成部分和原来一致，chance loss 主要用于估计 afterstate as^k 对应的 Q_t^k 和 σ_t^k 。具体地，chance loss 包含下面3部分：

- afterstate value loss : Q_t^k 用于匹配 afterstate 的 target value z_{t+k} ；
- afterstate policy loss : σ_t^k 用于估计 chance 分布，它的预测目标是 $c_{t+k+1} = onehot(argmax_i\ c_t^{e,i})$ ，也即 codebook 里面距离 encoder 输出的 embedding 最近的 **one hot chance code** ；
- commitment loss : 让 VQ-VAE 中 encoder 的输出 $c_{t+k+1}^e = e(o_{<=t+k+1})$ 和 codebook 里面对应的 one-hot code c_{t+k+1} 足够接近。

$$L_w^{chance} = \sum_{k=0}^{K-1} l^Q(z_{t+k}, Q_t^k) + \sum_{k=0}^{K-1} l^\sigma(c_{t+k+1},\ \sigma_t^k) + \beta \sum_{k=0}^{K-1} \|c_{t+k+1} - c_{t+k+1}^e\|^2$$



（图 3：Stochastic MuZero 算法概述图。 (A) 描述了Stochastic MuZero 所采用的蒙特卡洛树搜索方法，其中的菱形节点和圆形节点分别代表 chance node 和 decision node。在selection 阶段，decision node 通过应用 pUCT 算法选取边，而 chance node 则基于先验概率分布 σ 采样选择一个 chance。 (C) 描述了Stochastic MuZero 中随机模型的训练过程。以长度为2的给定轨迹为例，轨迹包含观察值 $o_{t:t+2}$ ，动作 $a_{t:t+2}$ ，价值目标 $z_{t:t+2}$ ，策略目标 $\pi_{t:t+2}$ ，以及奖励 $u_{t+1:t+K}$ ，模型会展开两步。在展开的过程中，编码器 e 接受观察值 $o_{<=t+k}$ 作为输入，并以确定性的方式生成chance code c_{t+k} 。模型的策略，价值和奖励输出被训练以分别逼近目标 π_{t+k} ， z_{t+k} 和 u_{t+k} 。关于 future codes 的分布 σ_k 的目标是由编码器生成的 chance code。）

3.2 Stochastic Search

- decision node 和 chance node在树上交替扩展，树的根节点一定是 decision node。
- 每一个 chance node 对应一个 latent afterstate。 它是由 decision node，输入action以后得到的。每个 chance node 对应了一个关于 future codes 的先验分布 $Pr(c|as)$ 和一个数值 afterstate value。
- 对于 decision node， 使用 pUCT 公式计算并取最大值对应的动作。

- 对于 chance node， 不再使用 pUCT 公式，而是根据上面得到的先验分布随机采样一个chance $c_{t+k+1} \sim \sigma_t^k$ 。

（注：在实际操作中，作者采用了与 Ozair 等人 (2021) (A.3) 中相同的**准随机抽样**(quasi-random sampling)方法，其中 chance code 的选择使用公式 $argmax_c \frac{Pr(c|as)}{N(c)+1}$ 进行。）

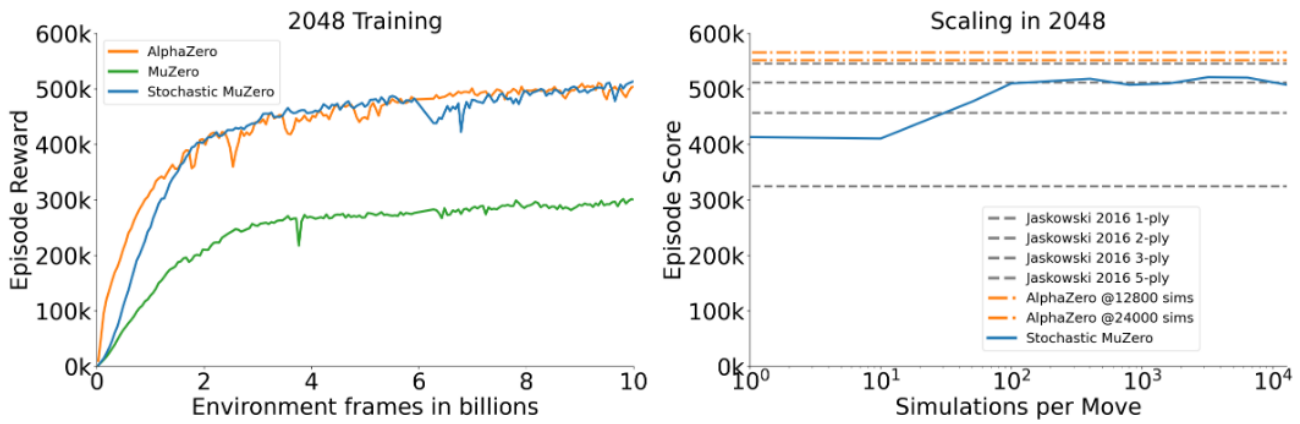
4. 实验

作者在下面3个环境中验证Stochastic MuZero的性能：

- 2048：这是一个单玩家的，带有随机性的游戏；
- Backgammon：一种 two player zero-sum stochastic 的棋盘类游戏，但是由于使用了骰子而有随机性；
- Go：确定性的环境，用来测衡量确定性环境中，使用 Stochastic MuZero 建模（相对于MuZero来说）带来的可能性能损失。

在每个环境中，作者评估作者的算法学习环境动力学模型的能力以及在搜索过程中有效利用它的能力。为此，作者将Stochastic MuZero（使用学习的随机模型）与 MuZero（使用学习的确定模型），AlphaZero（使用完美的模拟器）和一种强大的基准方法（也使用完美的模拟器）进行比较。在以下各节中，作者将分别介绍每个环境的结果。

4.1 2048

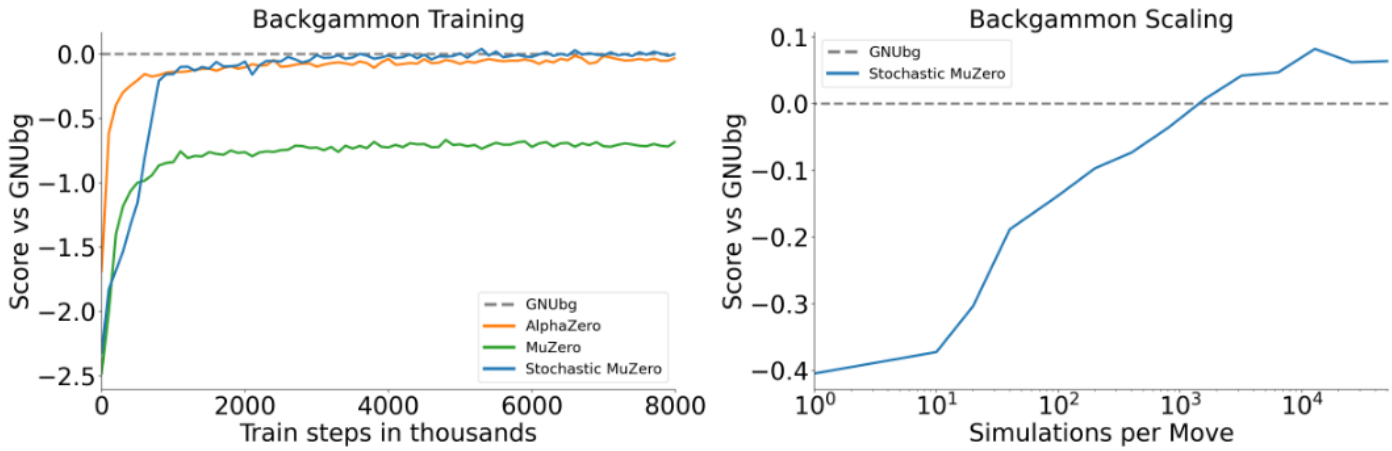


（图4：Planning in 2048. (a) num_simulations = 100 的 Stochastic MuZero 的性能几乎与 AlphaZero（使用 a perfect stochastic simulator）性能相匹配，而只针对确定性环境进行建模的 MuZero 表现会差很多。

(b) 评估使用不同搜索开销的最终智能体的性能。Stochastic MuZero 在评估过程中能够很好地处理到中等水平的搜索（大概相当于 3-ply lookahead），甚至超越了最先进的基准线（Ja´skowski，2016）所展现出的游戏实力。然而，作者发现，**随着模拟次数的增加，由于学习模型的不完美，收益开始出现递减的趋势**。）

- 2048游戏（受Threes!游戏启发）是一款在4x4的棋盘上单人完美信息随机拼图游戏。玩家在4x4的棋盘上滑动带有数字的方块，目标是将它们组合在一起创建一个数字为2048的方块；达成目标后仍可以继续游戏，创建更大数字的方块。每局的奖励是所有创建的方块数字之和。
- 在之前的大量研究中（Szubert & Ja´skowski, 2014; Yeh等人，2017; Oka & Matsuzaki, 2016; Rodgers & Levine, 2014; Neller, 2015），已经有不少工作尝试通过结合强化学习和树搜索方法来解决2048问题。尽管无模型方法在理论上较为简单，但实际上，传统的无模型方法往往难以取得优秀的性能，而基于规划的方法则成功利用了模拟器的完全知识。至今，性能最佳的智能体采用了在(Ja´skowski, 2016)中提出的基于规划的方法。这种方法利用了完美模拟器上的 expectimax 树搜索，结合了特定领域知识以及一些针对该特定问题结构的创新算法思想。
- 然而，作者的方法则采用了一个学习过的模型，并且没有对环境的先验知识。如图2所示，作者比较了Stochastic MuZero 在2048游戏中与 AlphaZero，MuZero 以及最新的 Jaskowski 2016 智能体的性能。尽管只使用了四分之一的训练数据，作者的方法还是超过了 Jaskowski 2016 的性能。值得注意的是，Stochastic MuZero 在学习模型的过程中，实现了与 AlphaZero（使用完美模拟器）相同的性能，并且远优于 MuZero（使用确定模型）。

4.2 BACKGAMMON



（图5：Stochastic MuZero 在西洋双陆棋中的应用。a) 通过使用学习得到的随机模型进行1600次模拟规划训练，Stochastic MuZero 的性能达到了与 AlphaZero 相当的水平，后者使用完美的随机模拟器进行了1600次模拟训练。此外，它的性能还达到了超人级别的程序 GNUbg Grandmaster 的水平。相比之下，使用确定性学习模型的 MuZero 表现不佳。b) Stochastic MuZero 的模型在大规模搜索中表现出良好的规模性，当使用超过 10^3 次模拟时，其游戏实力甚至超过了 GNUbg Grandmaster。）

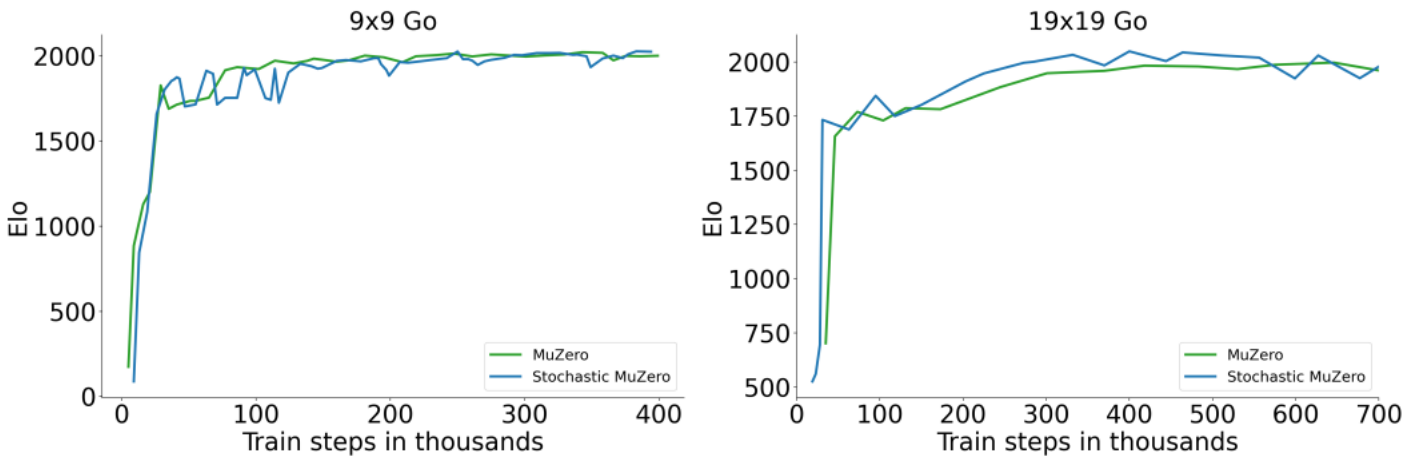
- 西洋双陆棋是一款经典的两人零和随机棋盘游戏，由于 TD-gammon (Tesauro, 1995)的推广，它已成为强化学习和人工智能的标准测试平台。作者的研究主要关注单局游戏的设定，其中最终得分为1代表简单的胜负，2代表大比分胜，3代表双倍大比分胜。

- 在所有的实验中，作者都将其与 GNUbg Grandmaster（Free Software Foundation, 2004）进行对比，这是一款超人级别的开源西洋双陆棋程序。GNUbg Grandmaster结合了基于手工特征的学习价值函数和使用完美随机模拟器的专业 min-max 树搜索。GNUbg Grandmaster采用的是一个在平均20个合法移动和21个chance转换中进行三层深度搜索 (a 3-ply look-ahead search)。
- 如图3b所示，Stochastic MuZero 采用学习得到的环境随机模型，并只进行每步1600次模拟，就达到了与 GNUbg 相同的游戏实力。Stochastic MuZero 学习的模型具有高质量：其游戏实力与使用完美随机模拟器的 AlphaZero 相当，且远高于使用确定性学习模型的 MuZero。
- 此模型也能稳定地拓展到更大的规划预算（如图5b所示）：随着每步模拟次数的增加，Stochastic MuZero 的性能有所提高，并最终超越了 GNUbg Grandmaster。
- 考虑到西洋双陆棋的动作空间维度高（详细信息请见附录），作者的西洋双陆棋实验采用了 Hubert 等人（2021）提出的 **sample-based 搜索方法**。

4.3 GO

围棋是一款古典的两个玩家、完美信息、零和棋盘游戏，在人工智能领域得到了广泛研究。AlphaZero 和随后的MuZero 是唯一通过自我对弈在这个具有挑战性的领域中实现超人类表现的算法。由于 Stochastic MuZero的目标是将MuZero的适用范围扩展到随机环境，同时在确定性环境中保持后者的性能，作者比较了两种算法在围棋游戏中的表现。

尽管由于使用了随机MCTS而不是确定性MCTS， Stochastic MuZero 需要两倍于 MuZero 的网络扩展才能达到相同的性能。作者为确保这些方法在计算上等效，通过将 Stochastic MuZero 网络的 chance 部分和 dynamic 部分的网络深度减半。



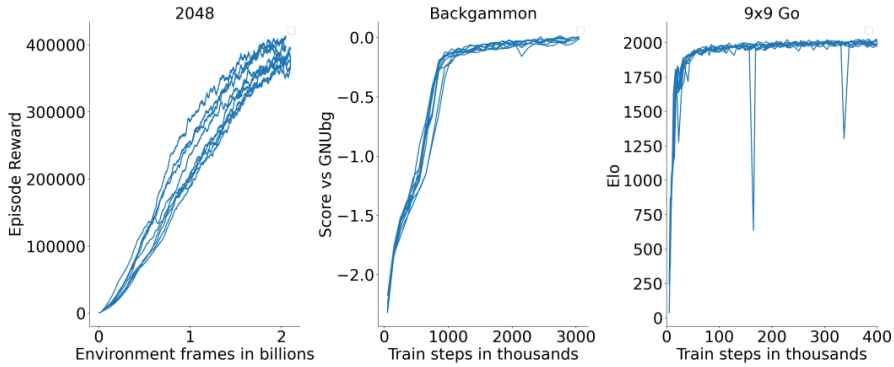
（图6：Stochastic MuZero 在围棋中的表现。Stochastic MuZero和MuZero在围棋游戏中的性能比较。（a）在9x9围棋中，Stochastic MuZero 和 MuZero 的性能比较。MuZero 的 num_simulation 在训练的时候设定为200, 在评估的时候设定为800;而Stochastic MuZero的 num_simulation 在训练的时候设定为400，在评估的时候设定为1600。

Elo的幅度被锚定，以使MuZero基线的最终性能相当于2000的Elo。（b）在19x19围棋中，Stochastic MuZero和MuZero的性能比较。MuZero的num_simulation在训练的时候设定为400, 在评

估的时候设定为800, 而Stochastic MuZero的num_simulation在训练的时候设定为800，在评估的时候设定为1600。Elo的幅度被锚定，以使MuZero基线的最终性能相当于2000的Elo。）

4.4 REPRODUCIBILITY

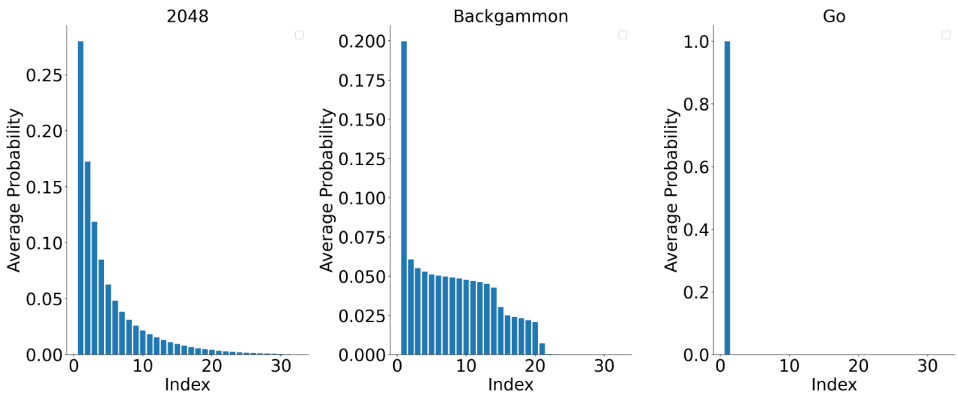
为了评估各种方法在不同环境下的稳健性，作者使用了九个不同的初始随机种子复制了上面的实验（见图5.4）。作者观察到各种方法对于随机初始化是鲁棒的，并且在多次运行之间其性能的变化很小。由于每个实验的计算成本，作者为每个实验使用了较少的训练步骤。



（图7：Stochastic MuZero 在各个环境上领域的可重复性。在所有环境中使用9个不同的种子运行了Stochastic MuZero，以衡量其对随机初始化的稳健性。对于所有不同的种子，Stochastic MuZero 的性能变化很小。由于每个实验的计算成本很大，作者为每个实验使用了较少的训练步骤。）

4.5 CHANCE ANALYSIS

我们对 Stochastic MuZero 中每个chance 节点的结果分布进行了研究。通过在每款游戏中存储节点上的概率分布，即所有后状态预测网络评估的 $\sigma_t^k = Pr(c_{t+k+1}|as_t^k)$，我们为每款游戏收集了数据集，这些评估都在5局的所有搜索过程中被调用。随后，我们对每个节点分布进行排序，并最终计算出平均分布，如图8所示。我们观察到，在围棋这样的确定性环境中，分布坍塌成一个单一的 code，而在随机环境中，模型会使用多个 code。另外，在西洋双陆棋中，分布有21个具有 non-negligible 概率的 code，这对应于两个骰子可能不同点数的所有组合。



（图8：学习到的 chance outcomes 的平均分布。在每个游戏中运行 Stochastic MuZero 5局后，所有 chance 节点上学习到的 chance outcomes 的平均分布。）

5. 总结与展望

在本研究中，作者提出了一种全新的方法，能够在全在线的强化学习环境中，学习环境的随机模型，并证明了这个学习到的模型可以有效地与规划相结合。作者的研究策略是基于 MuZero，一个在各种环境和设定中表现出色的基于模型的强化学习 agent。然而，MuZero 的应用主要局限于确定性或弱随机性环境。通过本研究，作者证明了 Stochastic MuZero ，能够克服 MuZero 的这些限制，在随机环境中的表现显著优于 MuZero，其性能甚至可与使用完美模拟器的 AlphaZero 相媲美。此外，作者还展示了在纯强化学习环境中，不借助任何环境先验知识的情况下， Stochastic MuZero 的性能能够匹配甚至超越那些使用完美随机模拟器的先前方法。

> 注：为了让研究社区能够复现作者的研究结果，作者提供了详细的伪代码，列出了所有使用过的环境和数据集，以及详细地描述了作者所使用的超参数（详见附录）。由于作者的完整代码依赖于大量的专有内部基础设施，作者并未公开全部代码。同时，作者也在论文中提供了一项研究，讨论了作者的方法在不同随机初始化条件下的稳健性（详见5.4节）。

6. 参考文献

[1] Model-Based Reinforcement Learning for Atari

[2] Dyna, an integrated architecture for learning, planning, and reacting

[3] Mastering Atari, Go, chess and shogi by planning with a learned model

[4] Value Prediction Network

[5] Mastering Atari with Discrete World Models

[6] Vector Quantized Models for Planning