SampledMuZero

1. 概述

- 许多重要的现实世界问题的动作空间是高维的、连续的或者两者兼有，这使得完全枚举所有可能的动作是不可行的。相反，在进行策略评估和策略提升时往往只能访问到原动作空间的一个子集。
- 本文提出了提出了一个通用的框架: sample-based policy iteration framework，则上可以适用于任意基于策略迭代的强化学习算法。核心思想是在动作空间的子集上计算 improved 的策略，当采样动作个数趋于整个动作空间时，sampled improved policy 按概率收敛于整个动作空间上的 improved policy。
- 具体来说，将上述框架实际运用于 MuZero 上得到 Sampled MuZero 算法。它是 MuZero 算法的扩展，能够通过在采样的动作集上规划从而在具有任意复杂动作空间的环境中学习。
- 最后作者在以下环境上验证了 Sampled MuZero 的性能：离散动作环境：Go, Atari Ms.Pacman；连续动作环境:：DM control suite，56D CMU Humanoid Locomotion task，RWRL。

2. 研究背景与相关工作

2.1 model-free RL

以前对复杂或连续动作空间的强化学习的研究通常集中在 model-free RL 算法上。

- 深度确定性策略梯度 (Deep Deterministic Policy Gradient，DDPG) 利用了这样一个事实: 即在完全连续的动作空间中，动作值函数 $Q(s,a)$ 可以被假设为相对于动作 a 可微分，以便有效地计算策略梯度。D4PG (Distributed Distributional Deterministic Policy Gradients) 通过在 DDPG 的基础上使用值分布函数和分布式训练有效地提高了性能。置信区域策略优化 (TRPO) 算法使用硬 KL 约束项，来确保在策略改进期间更新后的策略保持与先前的策略在一个小的置信区域内，以避免灾难性的崩溃。近端策略优化 (Proximal Policy Optimisation，PPO) 与 TRPO 具有相同的目标函数，但使用 KL-divergence 作为损失函数中的惩罚或价值函数中的 clipping，这样的改进得到了更简单的算法，但具有经验上更好的性能。
- 在 data-efficient off-policy 算法领域，最近的进展衍生出了优化熵正则化 RL 目标的 Actor-Critic 算法，如 SAC, MPO [1]，AWR。其中，MPO 使用了**基于采样的策略改进**，与作者的算法具有较大相关性。值分布 MPO (DMPO) 在 MPO 的基础上额外使用了 Q 值分布函数。

2.2 model-based RL

最近，在**高**维动作空间上使用基于模型的控制重新引起了人们的兴趣。这些算法中的大多数是对学习到的模型的进行直接策略优化，或者有时将 rollout based search/planning 与策略学习相结合。

- [1] 通过从 SAC 策略中对动作序列进行 sequential importance sampling 来进行规划。[3] 使用学习的模拟器构建 K 步回报，用于学习软 Q 函数。

与作者的工作最接近的是，[4] 采用与作者相似的基于采样的策略更新，但是他是使用一个基于 KL 正则化目标的策略改进算子，而作者这里考虑的是基于 MCTS 的策略改进。

稀疏采样算法 [5] 是在大型状态空间中规划的有效方法。主要思想是从每个状态中抽取可能的状态转换，从底层 MDP 对应的生成模型中提取。总的来说，这些样本提供了 MDP 子集上的搜索树；对于大的 K，在采样树上的规划提供了对于全 MDP 的最优策略的 near-optimal approximation，而与状态空间的大小无关。事实上，众所周知，抽样在某些情况下可以解决维数灾难。然而，稀疏采样通常列举来自每个状态的所有可能的动作，并且不能解决与大动作空间相关的问题。

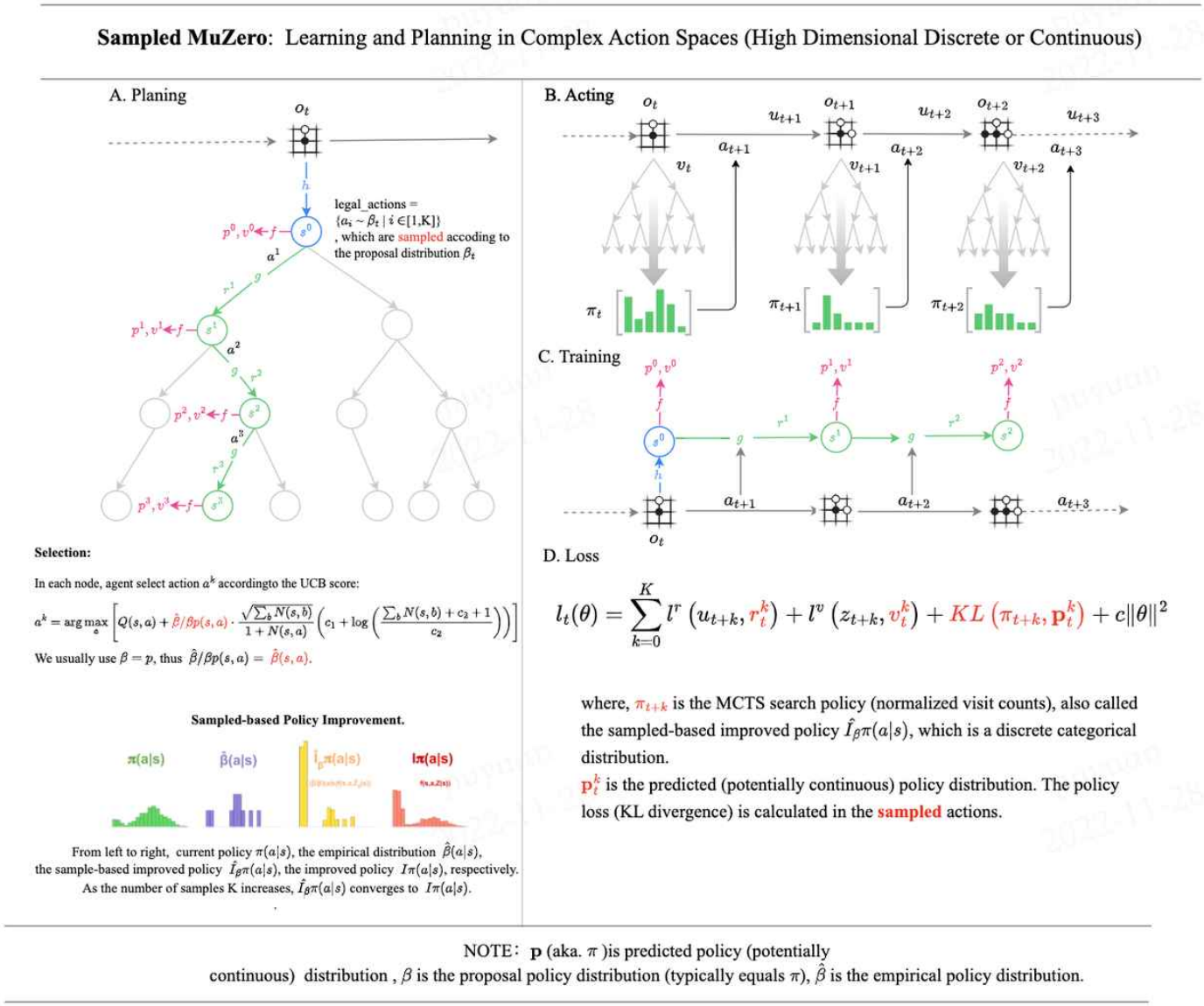
相比之下，**作者的方法采样的对象是动作而不是状态转换**。原则上，将这两种想法结合起来会很简单；然而，作者在本文中关注研究大型的动作空间，并且使用了确定性转换模型。

以前曾有几个工作尝试将 AlphaZero 和 MuZero 推广到连续动作空间。这些尝试表明，这种扩展原则上是可能的，但迄今为止**仅限于非常低维的情况**，还没有证明在高维任务中的有效性。

- A0C [6] 描述了一种方法将 AlphaZero 扩展到连续动作空间，他们使用连续策略表征和 REINFORCE 以估计神经网络策略和目标 MCTS 策略之间的 reverse KL divergence，在1D Pendulum 任务上取得了一些效果。[7] 描述了一种类似的方法将 MuZero 扩展到连续动作空间上，并在具有1维和2维动作空间的环境中得到了优于 SAC 的结果。
- [8] 描述的 factorised policy representation 在多个领域显示出良好的结果；通过用单独的 categorical distribution 表示每个动作维度，有效地避免了简单离散化方案所面临的动作数量的指数爆炸问题。

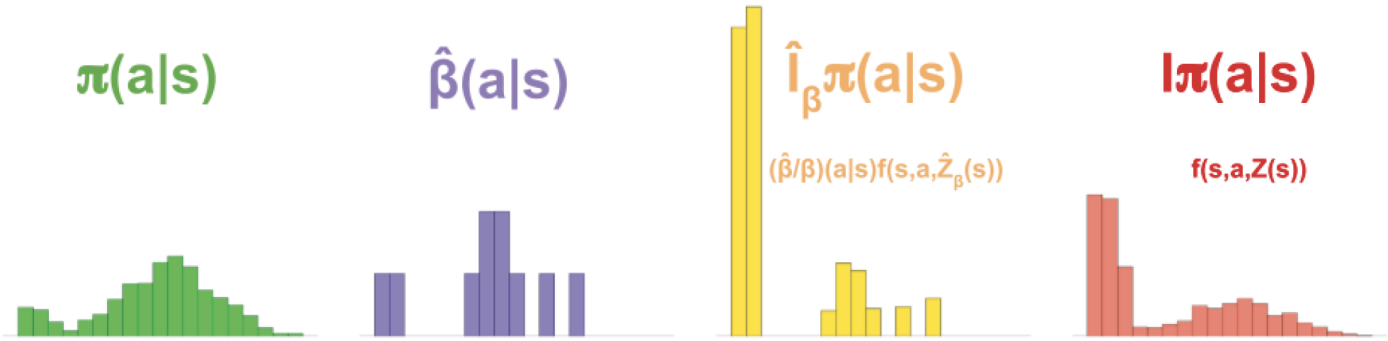
3. Sampled MuZero 算法

3.1 概览图



(图1：Sampled MuZero 算法概览图。与 MuZero 的主要区别：**A**：每次扩展节点时，是从一个提议分布中采样 K 个动作。**D**: polcy loss的计算：当动作空间为离散时，计算采样的 K 个动作的 visit count 分布和 policy 网络输出的离散分布之间的 KL 散度；当动作空间为连续时，计算采样的 K 个动作的 visit count 分布 和 这 K 个动作在 policy 网络输出的连续分布参数（例如高斯分布的均值和标准差）对应的分布的概率密度 之间的 KL 散度。符号的含义参考【算法概览图符号表.pdf】。)

3.2 基于采样的策略迭代 (Sample-based Policy Iteration)



(图1：基于采样的策略改进 (Sampled-based Policy Improvement)。最左边表示当前策略 $\pi(a|s)$ 。接着，根据提议分布 (proposal distribution) β （参见后文，通常由 π 变换得到）从动作空间中采

样 K 个动作 $\{a_i\}$，由采样的动作计算得到的 empirical distribution 记为 $\hat{\beta}$。然后，构建一个基于采样的改进策略 $\hat{I}_\beta \pi(a|s) := (\hat{\beta}/\beta)(a|s) f(s, a, \hat{Z}_\beta(s))$。当 K 增加时，$\hat{I}_\beta \pi(a|s)$ 依照概率收敛于改进的策略 $I_\pi\ (a|s)\ =\ f(s, a, Z(s))$。）

3.2.1 策略提升的算子视角 (Operator view of Policy Improvement)

- 给定策略 $\pi: S \rightarrow P(A)$ 和策略 $I_\pi: S \rightarrow P(A)$ 如果满足 $\forall\ s\ \in S,\ v^{I_\pi}(s)\ >= v^\pi(s)$ 则称 I_π 是 π 的提升策略 (improved policy)。
- 将 policy improvement 分为2个阶段：
  - a policy improvement operator I : 将任何一个策略映射到 return 严格更大的策略上。
  - a projection operator $\mathcal{P}$: 在网络可表征的空间中找到 improved policy 的最好近似。

> 备注：
>
> - MuZero 中的 MCTS Search 部分可以看做策略提升的一部分:
>   - inner process 包括 MCTS Search 部分，为 outer process 提供策略提升所需的改进的 target 策略。
>   - outer process 维护参数化的 policy function 和 value function。
>   - 策略提升：让策略网络的参数逐渐回归到 MCTS 搜索到的策略上。
>   - 策略评估：由值网络朝着 n_step return 更新得到。

- **问题：**通常情况下，作者可以访问 $\mathcal{A}$ 的整个动作空间，用于整个动作空间上的策略提升，但是当动作空间 $\mathcal{A}$ 过大时，作者只有一个局部提升的 I_π。如何用这个局部提升的 I_π 来进行完整动作空间上的策略提升?

3.2.2 独立于动作的策略提升算子 (Action-Independent Policy Improvement Operator)

- 作者定义一个算子是 **独立于动作的** (action-independent) 策略提升算子，如果它满足：

$I_\pi(a|s) = f(s, a, Z(s))$, 这里 $Z(s)$ 是一个唯一的依赖于state的归一化因数，定义为:

$$\forall a\ \in\ \mathcal{A},\ f(s, a, Z(s))\ >= 0\ and\ \sum_{a \in\ \mathcal{A}} f(s, a, Z(s)) = 1$$

3.2.3 独立于动作的基于采样的策略提升算子 (Sample-Based Action-Independent Policy Improvement Operator)

- 首先根据提议分布 (proposal distribution) β 从动作空间中采样 K 个动作。用 $\{a_i\}$ 表示 K 个从 proposal distribution β 中采样的动作集合，由采样的动作计算得到的 empirical distribution 记为 $\hat{\beta}$。

> 例子：完整动作集合 $\mathcal{A}$ 为{0,1,2,3,4,5} ，提议分布 (proposal distribution) 为 $\beta = (0.2, 0.2, 0.2, 0.2, 0.2)$

> 实际采样 K=10个动作为 {0,0,0,1,2,3,4,5,5,5}，则经验分布(empirical distribution）为 $\hat{\beta} = (0.3, 0.1, 0.1, 0.1, 0.3)$

- $\hat{\beta} = \frac{1}{K} \sum_i \delta_{a,a_i}$ ， 这个分布是一个 Kronecker delta function，仅在被采样的动作上为非0。
- 下面定义 $\hat{I}_\beta \pi(a|s) := (\hat{\beta}/\beta)(a|s) f(s, a, \hat{Z}_\beta(s))$ ，这里 $\hat{Z}_\beta(s)$ 是一个唯一的依赖于state的归一化系数，定义为:

  $\forall a \in \mathcal{A}, (\hat{\beta}/\beta)(a|s) f(s, a, \hat{Z}_\beta(s)) >= 0$ 而且 $\sum_{a \in \mathcal{A}} (\hat{\beta}/\beta)(a|s) f(s, a, \hat{Z}_\beta(s)) = 1$ ，其中

  $$(\hat{\beta}/\beta)(a|s) = \hat{\beta}(a|s) \ / \ \beta(a|s)$$

3.2.4 计算关于分布 I_π 的期望 (Computing an expectation with respect to I_π)

> 💡 定理：
>
> 对于一个给定的随机变量X，作者有:
>
> $$\mathbb{E}_{a \in I_\beta}[X|s] = \lim_{K \to \inf} \sum_{a \in A} \hat{I}_\beta \pi(a|s) X(s, a)$$
>
> 推论：
>
> Sample-Based Policy Improvement Operator 依照概率收敛于 True Policy Improvement Operator:
>
> $$\lim_{K \to \inf} \hat{I}_\beta \pi(a|s) = I_\pi(a|s)$$

3.2.5 基于采样的策略评估和策略提升 (Sample-based Policy Evaluation and Improvement)

- 上一个表达式计算 $\mathbb{E}_{a \in I_\beta}[X|s]$ 的估计值时使用 $\hat{I}_\beta \pi(a|s)$ 和采样动作 $\{a_i\}$, 可用于对 improved policy 进行策略提升和策略评估:
- 策略提升可以按照下面的流程进行：例如 $X = -\log \pi_\theta$, 最小化 π_θ 和 improved policy I_π 之间的交叉熵：

  $$CE = \mathbb{E}_{a \in I_\pi}[-\log \pi_\theta]$$

- 策略评估可以按照下面的流程进行：例如为了 planning，一个直接的 improved policy 的策略评估步骤可以是：通过估计 1 步或 n 步 returns 来估计策略的值函数。 例如让 $X = r + \gamma V'$ ，作者在搜索树中将值 V' 反向传播一步: $V(s) = E_{a \in I_\pi}[r + \gamma V']$ 。

3.3 Sampled MuZero

3.3.1 选择，扩展，备份 (Selection, Expansion, Backup)

MuZero在搜索树上选择要扩展的节点是根据 PUCT (probabilistic upper confidence tree)：

$$argmax_a Q(s,a) + c(s) * \pi(s,a) * (\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$$

直接的修改方法:

- 为了把 MuZero 的 MCTS 搜索扩展到采样的动作集合 $\{a_i\}$ 上， 一个直接的想法是保持 PUCT 的公式不变，只是利用 MCTS 搜索得到的 visit count distribution f 来构造用于采样的 $\hat{I}_\beta \pi = \hat{\beta}/\beta\ f$ ，用于矫正策略网络在训练和执行动作时采样的影响， 理论上这个方法不复杂，但是实际上由于 f/β 项的存在会很不稳定，因为 f 是 normalized visit count distribution，只有有限的数值准确度。(TODO：为什么？)

论文提出的修改方法:

- 文章中提出可以只在 MCTS search 时用 $\hat{\pi}_\beta(s,a) \propto \hat{\beta}/\beta\ \pi(s,a)$ 来替代 $\pi_\beta(s,a)$ ，而在策略学习时像 MuZero一样，直接用 MCTS search 返回的 normalizaed visit count distribution 作为 target policy。

> 💡 定理：
>
> 给定 I_π 是 在 MuZero 考虑完整动作空间 $\mathcal{A}$，使用先验 π 进行 search 得到的 visiit count distribution。给定 $I\hat{\pi}_\beta$ 是 在 MuZero 考虑完整动作空间 $\mathcal{A}$，使用先验 $\hat{\pi}_\beta$ 进行 search 得到的 visiit count distribution。
>
> 得到 $I\hat{\pi}_\beta$ 近似等于 相对于 I_π 的sampled-based policy improvement，即 $I\hat{\pi}_\beta \approx \hat{I}_\beta \pi$ 。

3.3.2 搜索树节点扩展 (Search Tree Node Expansion)

- 在 MuZero 中，每次在扩展一个叶子节点时，整个动作空间的**所有的** $N = |\mathcal{A}|$ 个动作以及策略网络分配给每一个动作的概率都会被计算并保存下来
- 论文提出的修改方法: 在 Sampled MuZero 中，作者改为从提议分布 β 中采样 $K << N$ 个动作并返回每个动作 a 及其相应的概率 $\pi(s,a)$ 和 $\beta(s,a)$ 。

3.3.3 采样分布 β (Sampling distribution)

- 理论上任意具有 wide support 的分布都可以，甚至可以使用均匀分布，不过由于 simulations 次数有限，动作的采样次数有限，最好根据当前估计的最优策略 π 来采样，可以增加一个温度系数。
- 为了让低先验的动作也能被采样到，对 $\pi(s,a)$ 和 $\beta(s,a)$ ，都增加了 Dirichlet 噪声。

> 💡 注意，如果 $\beta = \pi^{1/\tau}$ ，在PUCT公式中使用的 $\hat{\pi}_\beta$ 可以写成： $\hat{\pi}_\beta = \hat{\beta}/\beta\pi = \hat{\beta}\pi^{1-1/\tau}$ 。
>
> - 如果 $\tau = 1$， $\hat{\pi}_\beta$ 等于经验采样分布 (empirical sampling/prior distribution) $\hat{\beta}$ 。这意味着搜索是由一个潜在的可能是quasi uniform prior $\hat{\beta}$ 所指导。
> - 如果 $\tau > 1$ ，意味着搜索是由一个更加尖峰的先验分布 $\hat{\beta}\pi^{1-1/\tau}$ 所指导，可以搜索评估到更加多样性的样本。

3.3.4 外部策略改进 (Outer Policy Improvement)

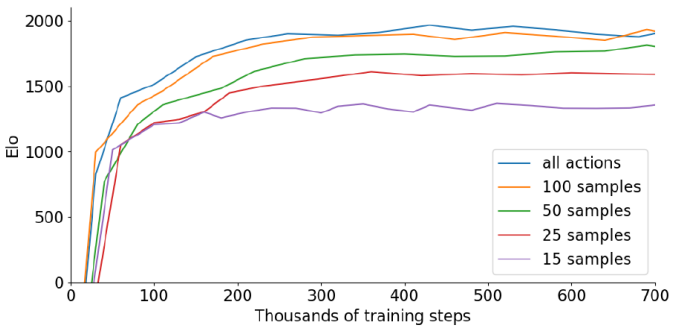
- MCTS搜索完成后，在根节点 s 返回 visit counts 集合 $\{N(s,a)\}$，这些 visit counts 经过归一化后得到 sample-based improved policy $\mathcal{I}_{\hat{\pi}_\beta}(a|s) = N(s,a)/\sum_b N(s,b)$。
- 经过前面的分析，这里得到的 visit counts 已经考虑了root action是根据 β 采样的影响。
- 所以，只需要使用一个合适的 projection operator $\mathcal{P}$ 将 sampled-based improved policy 投射到网络可表征的空间中， 根据 MuZero，论文通过最小化 KL 散度： $KL(\mathcal{I}_{\hat{\pi}_\beta}||\pi_\theta)$ 执行这一步骤。

3.3.5 外部策略评估 (Outer Policy Evaluation)

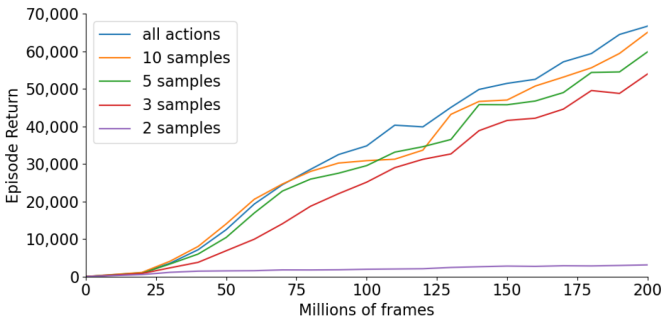
- 与MuZero一样，外部策略评估步骤，使用 sample based improved policy 生成的行为轨迹，从 *n*-step returns训练值函数。

4. 实验

4.1 离散动作空间: Go, Atari

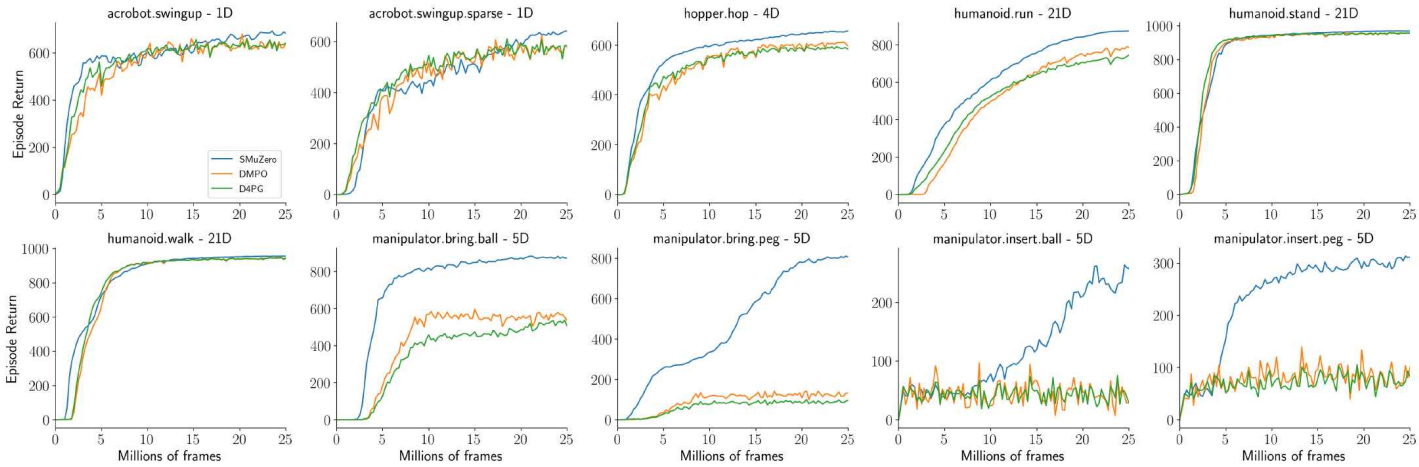


(图2：在经典的棋盘游戏-围棋中的结果。在整个训练过程中，具有不同采样动作数量 $K \in 15, 25, 50, 100$ 的 Sampled MuZero 与总是考虑所有可用的动作（动作空间大小为362）的 MuZero 基线（蓝色曲线）的性能比较（每个实验1个种子）。纵轴表示 Elo 等级，将最终基准性能锚定于2000 Elo。随着采样动作数量的增加，Sampled MuZero 的性能会迅速接近始终考虑所有动作的 MuZero 基线。)

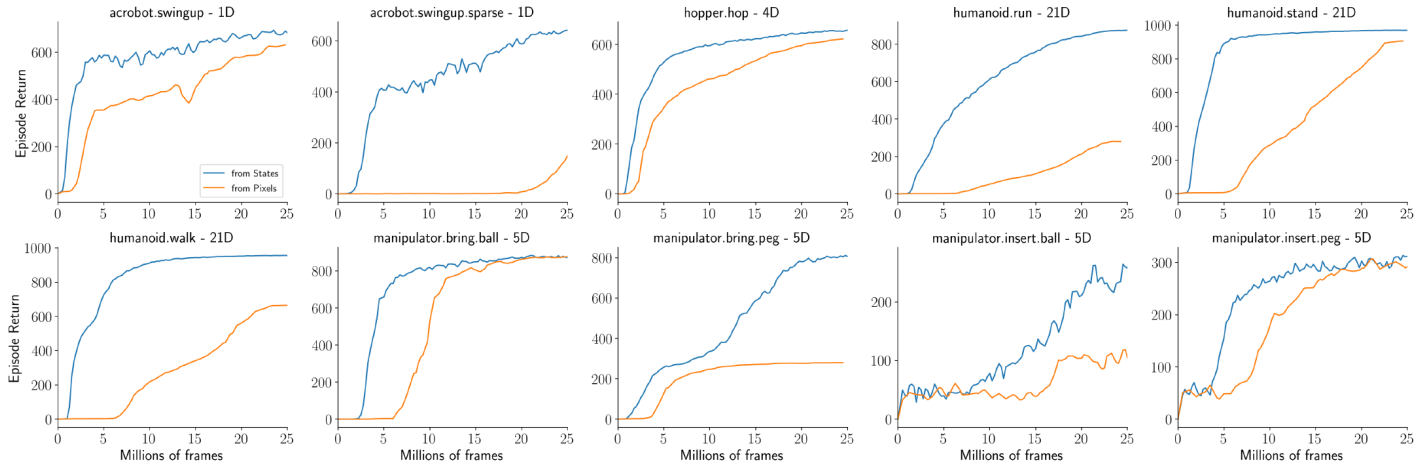


（图3：Ms.Pacman 上的结果。与始终考虑所有可用动作的 MuZero 基线相比，在整个训练期间，具有不同采样动作数量的 Sampled MuZero 的性能（每个实验1个种子）。随着采样动作数量的增加，性能会迅速接近始终考虑所有动作的 MuZero 基线。）

4.2 连续动作空间: DM Control Suite Hard and Manipulator tasks

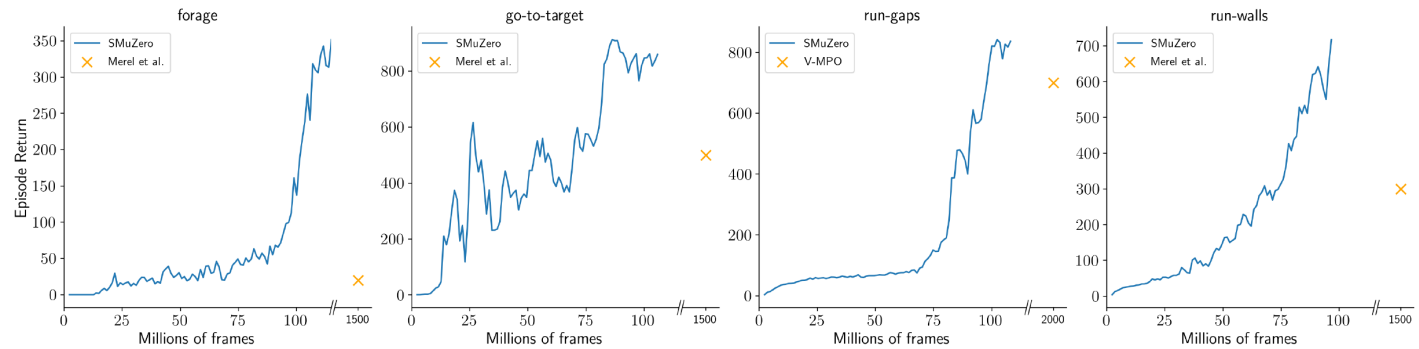


（图4：在 DM Control Suit Hard 和 Manipulator 任务上，**输入为状态向量的** Sampled MuZero 的实验结果。与 DMPO 和 D4PG 相比，Sampled MuZero 在整个训练中的表现 (每个实验3个种子) 。x 轴表示数百万个环境帧，y 轴表示 episode return。DM Control Suit Hard 是在论文 (Hoffman et al., 2020) 提出的。每个环境图片的标题包括任务名称和动作空间的维度。）



（图5：在 DM Control Suit Hard 和 Manipulator 任务上，**输入为图像的** Sampled MuZero 的实验结果。**输入为图像的** Sampled MuZero 与 **输入为状态向量的 Sampled MuZero** 在整个训练中的表现 (每个实验3个种子) 。x 轴表示数百万个环境帧，y 轴表示 episode return。）

4.3 连续动作空间: 56D CMU Humanoid Locomotion tasks



（图6：56D CMU Humanoid Locomotion 任务上的结果。在基于dm_control 的 CMU Humanoid Locomotion 任务上， Sampled MuZero 在整个训练中的性能表现 (每个实验1个种子)，动作空间维度

为 humanoid 身体的56个关节。Sampled MuZero 在 forage、go-to-target 和 run-walls 以及 run-gaps 等任务上都优于先前论文报告的结果，同时使用的环境交互步数减少了一个数量级以上。）

4.3 Real-World RL Challenge Benchmark

Agent	Cartpole	Walker	Quadruped	Humanoid
		Easy		
DMPO	464.05	474.44	567.53	1.33
D4PG	482.32	512.44	787.73	102.92
STACX	734.40	487.75	865.80	1.21
SMuZero	**861.05**	**959.83**	**987.20**	**289.36**
		Medium		
DMPO	155.63	64.63	180.30	1.27
D4PG	175.47	75.49	268.01	1.28
STACX	398.71	94.01	466.43	1.18
SMuZero	**516.69**	**448.51**	**946.21**	**108.56**
		Hard		
DMPO	138.06	63.05	144.69	1.40
D4PG	108.20	59.85	280.75	1.27
STACX	135.26	58.11	**351.56**	1.26
SMuZero	**244.71**	**71.16**	348.09	1.19

（表1：Sampled MuZero 在 Real-Word RL benchmark (RWRL)上的实验结果。在 RWRL，每项任务都有难度依次递增的简单、中等和困难版本。DMPO 和 D4PG 的结果来自论文 (Dulac-Arnold et al., 2020)，STACX 来自论文(Zahavy et al., 2020)。Sampled MuZero 显示了强大的性能，尤其是在最高维的 Humanoid 任务上（每个实验3粒种子)。）

5. 总结与展望

- Sampled MuZero 核心思路是，在 MCTS 里面扩展节点的时候，并不枚举所有动作，而是取一个动作子集来作为备选集合（可以看做 leagal actions），搜索就限制在这个集合里。作者推导了很多理论，证明了虽然是通过采样的方式来做 planning 的，但是这种方式是有收敛保证的，只要随着采样动作数量 K 趋近无穷，那么这个 sample-based policy improvement operator 就依分布收敛到 true policy improvement operator。直观上理解，这其实是显然的，因为当 K 趋近无穷时就还原到了原始的扩展所有可行动作的 MuZero 算法。
- 但实际中，如果采样动作 K 远小于总的可行动作 N，能否学到近似最优的策略呢？作者的实验结果给出了肯定答案，但直观理解其原因是什么呢？
    - 虽然每次在每个节点上都只采样了 K 个动作来做 MCTS，但是不同的节点，以及相同节点在每次新建一个search tree 的时候，每次采样的 K 个动作都是不一样的。所以随着 policy network 和 value network 的逐步更新，再加上根节点处增加的随机探索噪声，导致每个动作都有概率被采样到，只是可能不是出现在同一次 MCTS 中的同一个节点。

- 同时一个关键点在于，最优策略应该是确定性策略，所以最终一个节点只要有一个动作就够了，这也保证了采样 K << N 个动作从原理上来说是足够的。

6. 参考文献

[1] Abdolmaleki, A., Springenberg, J. T., Degrave, J., Bohez, S., Tassa, Y., Belov, D., Heess, N., and Riedmiller, M.

Relative entropy regularized policy iteration, 2018.

[2] Pich´e, A., Thomas, V., Ibrahim, C., Bengio, Y., and Pal, C. Probabilistic planning with sequential monte carlo

methods. In International Conference on Learning Representations, 2018.

[3] Bhardwaj, M., Handa, A., Fox, D., and Boots, B. Information theoretic model predictive q-learning. In Learning for Dynamics and Control, pp. 840–850. PMLR, 2020.

[4] Springenberg, J. T., Heess, N., Mankowitz, D., Merel, J., Byravan, A., Abdolmaleki, A., Kay, J., Degrave,

J., Schrittwieser, J., Tassa, Y., et al. Local search for policy iteration in continuous control. arXiv preprint

arXiv:2010.05545, 2020.

[5] Kearns, M., Mansour, Y., and Ng, A. Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI’ 99, pp. 1324–1331, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[6] Moerland, T. M., Broekens, J., Plaat, A., and Jonker, C. M. A0C: AlphaZero in continuous action space, 2018.

[7] Yang, X., Duvaud, W., and Wei, P. Continuous control for searching and planning with a learned model, 2020.

[8] Tang, Y. and Agrawal, S. Discretizing continuous action space for on-policy optimization, 2020.

[9] https://www.cnblogs.com/initial-h/p/15159611.html

[10] https://www.furidamu.org/blog/2021/07/18/sampled-muzero--learning-and-planning-in-complex-action-spaces/

[11] https://github.com/werner-duvaud/muzero-general/tree/continuous