Policy Gradient With Serial Markov Chain Reasoning

Edoardo Cetin
Department of Engineering
King's College London
edoardo.cetin@kcl.ac.uk

Oya Celiktutan
Department of Engineering
King's College London
oya.celiktutan@kcl.ac.uk

Abstract

We introduce a new framework that performs decision-making in reinforcement learning (RL) as an iterative *reasoning* process. We model agent behavior as the *steady-state distribution* of a parameterized *reasoning* Markov chain (RMC), optimized with a new tractable estimate of the policy gradient. We perform action selection by simulating the RMC for enough *reasoning steps* to approach its steady-state distribution. We show our framework has several useful properties that are inherently missing from traditional RL. For instance, it allows agent behavior to approximate any continuous distribution over actions by parameterizing the RMC with a simple Gaussian transition function. Moreover, the number of reasoning steps to reach convergence can scale adaptively with the difficulty of each action selection decision and can be accelerated by re-using past solutions. Our resulting algorithm achieves state-of-the-art performance in popular Mujoco and DeepMind Control benchmarks, both for proprioceptive and pixel-based tasks.

1 Introduction

Reinforcement learning (RL) has the potential to provide a general and effective solution to many modern challenges. Recently, this class of methods achieved numerous impressive milestones in different problem domains, such as games [1–3], robotics [4–6], and other meaningful real-world applications [7–9]. However, all these achievements relied on massive amounts of data, controlled environments, and domain-specific tuning. These commonalities highlight some of the current practical limitations that prevent RL to be widely applicable [10].

In the deep RL framework, practitioners train agents with the end goal of obtaining optimal *behavior*. Traditionally, agent behavior is modeled with *feed-forward policies* regressing from any state to a corresponding distribution over actions. Such formulation yields practical training objectives in both off-policy [11–13] and on-policy settings [14–16]. However, we identify three inherent properties of this rigid representation of behavior that could considerably impact expressivity and efficiency in continuous control tasks. First, agent behavior is restricted to a class of tractable distributions, which might fail to capture the necessary complexity and multi-modality of a task. Second, the policy performs a fixed *reasoning* process with a feed-forward computation, which potency *cannot adapt* to the varying complexity of individual action selection problems. Third, decision-making is performed every time *from scratch*, without re-using any past information that might still inform and facilitate the current action selection problem.

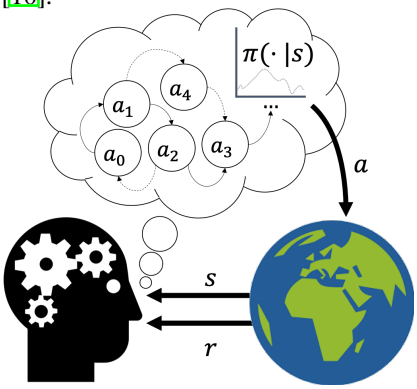

Figure 1: Depiction of agent decision-making with serial Markov chain reasoning.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Unlike RL policies, human reasoning does not appear to follow a rigid feed-forward structure. In fact, a range of popular psychological models characterize human decision-making as a *sequential* process with adaptive temporal dynamics [17–20]. Many of these models have found empirical groundings in neuroscience [21–24] and have shown to effectively complement RL for capturing human behavior in experimental settings [25, 26]. Partly inspired by these works, we attempt to *reframe* the deep RL framework by making use of a similar flexible model of agent behavior, in order to counteract its aforementioned limitations.

We introduce *serial Markov chain reasoning* - a new powerful framework for representing agent behavior. Our framework treats decision-making as an *adaptive reasoning process*, where the agent sequentially updates its beliefs regarding which action to execute in a series of *reasoning steps*. We model this process by replacing the traditional policy with a parameterized *transition function*, which defines a *reasoning Markov chain* (RMC). The steady-state distribution of the RMC represents the distribution of agent behavior after performing enough *reasoning* for decision-making. Our framework naturally overcomes the aforementioned limitations of traditional RL. In particular, we show that our agent's behavior can approximate any arbitrary distribution even with simple parameterized transition functions. Moreover, the required number of *reasoning steps* adaptively scales with the difficulty of individual action selection problems and can be accelerated by re-using samples from similar RMCs.

To optimize behavior modeled by the steady-state distribution of the RMC, we derive a new tractable method to estimate the *policy gradient*. Hence, we implement a new effective off-policy algorithm for maximum entropy reinforcement learning (MaxEnt RL) [27, 28], named *Steady-State Policy Gradient* (SSPG). Using SSPG, we empirically validate the conceptual properties of our framework over traditional MaxEnt RL. Moreover, we obtain state-of-the-art results for popular benchmarks from the OpenAI Gym Mujoco suite [29] and the DeepMind Control suite from pixels [30].

In summary, this work makes the following key contributions:

1. We propose *serial Markov Chain reasoning* a framework to represent agent behavior that can overcome expressivity and efficiency limitations inherent to traditional reinforcement learning.
2. Based on our framework, we derive *SSPG*, a new tractable off-policy algorithm for MaxEnt RL.
3. We provide experimental results validating theorized properties of *serial Markov Chain reasoning* and displaying state-of-the-art performance on the Mujoco and DeepMind Control suites.

2 Background

2.1 Reinforcement learning problem

We consider the classical formulation of the reinforcement learning (RL) problem setting as a Markov Decision Process (MDP) [31], defined by the tuple $(S, A, P, p_0, r, \gamma)$. In particular, at each discrete time step t the agent experiences a state from the environment's state-space, $s_t \in S$, based on which it selects an action from its own action space, $a_t \in A$. In continuous control problems (considered in this work), the action space is typically a compact subset of an Euclidean space $\mathbb{R}^{dim(A)}$. The evolution of the environment's state through time is determined by the transition dynamics and initial state distribution, P and p_0. Lastly, the reward function r represents the immediate level of progress for any state-action tuple towards solving a target task. The agent's behavior is represented by a state-conditioned parameterized policy distribution π_θ. Hence, its interaction with the environment produces trajectories, $\tau = (s_0, a_0, s_1, ..., s_T, a_T)$, according to a factored joint distribution $p_{\pi_\theta}(\tau) = p_0(s_0) \prod_{t=0}^{T} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$. The RL objective is to optimize agent behavior as to maximize the discounted sum of expected future rewards: $\arg\max_\theta \mathbb{E}_{p_{\pi_\theta}(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]$.

2.2 Maximum entropy reinforcement learning and inference

Maximum entropy reinforcement learning (MaxEnt RL) [32] considers optimizing agent behavior for a different objective that naturally arises when formulating action selection as an inference problem [33–36]. Following Levine [28], we consider modeling a set of binary optimality random variables with realization probability proportional to the exponentiated rewards scaled by the temperature α, $p(O_t|s_t, a_t) \propto \exp(\frac{1}{\alpha} r(s_t, a_t))$. The goal of MaxEnt RL is to minimize the KL-divergence between

trajectories stemming from agent behavior, $p_{\pi_\theta}(\tau)$, and the inferred optimal behavior, $p(\tau|O_{0:T})$:

$$\begin{aligned} D_{KL}\left(p_{\pi_\theta}(\tau)||p(\tau|O_{0:T})\right) &= \mathbb{E}_{p_{\pi_\theta}(\tau)}\left[\log \frac{p_0(s_0)\prod_{t=0}^{T}\pi_\theta(a_t|s_t)P(s_{t+1}|s_t,a_t)}{p_0(s_0)\prod_{t=0}^{T}\exp(\frac{1}{\alpha}r(s_t,a_t))P(s_{t+1}|s_t,a_t)}\right] \\ &= -\mathbb{E}_{p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{T} r(s_t,a_t) + \alpha H(\pi(\cdot|s_t))\right]. \end{aligned} \tag{1}$$

The resulting entropy-regularized objective introduces an explicit trade-off between exploitation and exploration, regulated by the temperature parameter α scaling the policy's entropy. An effective choice to optimize this objective is to learn an auxiliary parameterized *soft Q-function* [37]:

$$Q^{\pi}_{\phi}(s_t,a_t) = \mathbb{E}_{p_{\pi_\theta}(\tau|s_t,a_t)}\left[r(s_t,a_t) + \sum_{t'=t+1}^{T} r(s_{t'},a_{t'}) + \alpha H(\pi(a_{t'}|s_{t'})\right]. \tag{2}$$

Given some state, $Q^{\pi}_{\phi}(s,\cdot)$ represents an energy-function based on the expected immediate reward and the agent's future likelihood of optimality from performing any action. Thus, we can locally optimize the MaxEnt objective by reducing the KL-divergence between π and the canonical distribution of its current soft Q-function. This is equivalent to maximizing the expected soft Q-function's value corrected by the policy's entropy, resembling a regularized policy gradient objective [11, 12]:

$$\arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_\theta(\cdot|s)}\left[Q^{\pi}_{\phi}(s,a) + \alpha H(\pi_\theta(a|s))\right]. \tag{3}$$

The policy is usually modeled with a neural network outputting the parameters of some tractable distribution, such as a factorized Gaussian, $\pi_\theta(\cdot|s) = N(\mu_\theta(s);\Sigma_\theta(s))$. This practice allows to efficiently approximate the gradients from Eqn. 3 via the reparameterization trick [38]. We consider the off-policy RL setting, where the agent alternates learning with storing new experience in a data buffer, D. We refer the reader to Haarnoja et al. [13, 39] for further derivation and practical details.

3 Policy Gradient with serial reasoning

3.1 Reasoning as a Markov chain

We introduce *Serial Markov Chain Reasoning*, a new framework to model agent behavior, based on conceptualizing action selection as an adaptive, sequential process which we refer to as *reasoning*. Instead of using a traditional policy, the agent selects which action to execute by maintaining an internal *action-belief* and a *belief transition (BT-) policy*, $\pi^b(a'|a,s)$. During the *reasoning process*, the agent updates its action-belief for a series of *reasoning steps* by sampling a new action with the BT-policy π^b taking both environment state and previous action-belief as input. We naturally represent this process with a *reasoning Markov chain* (RMC), a discrete-time Markov chain over different action-beliefs, with transition dynamics given by the BT-policy. Hence, for any input environment state s and initial action-belief a_0, the n-step transition probabilities of the RMC for future *reasoning steps* $n = 1, 2, 3, ...$ are defined as:

$$\pi^b_n(a|a_0,s) = \int_A \pi^b(a|a',s)\pi^b_{n-1}(a'|a_0,s)da', \quad \textit{for } n>1, \textit{ and} \quad \pi^b_1 = \pi^b. \tag{4}$$

Given a compact action space and a BT-policy with a non-zero infimum density, we can ensure that as the number of reasoning steps grows, the probability of any action-belief in the RMC *converges* to some *steady-state probability* which is independent of the initial action-belief.[1] We denote this *implicit* probability distribution as the *steady-state (SS-) policy*, symbolized by $\pi^s(a|s)$:

Lemma 3.1. ***Steady-state convergence.*** *For any environment state s, consider a reasoning Markov chain (RMC) defined on a compact action space A with transition probabilities given by $\pi^b(a'|a,s)$. Suppose that $\inf\{\pi^b(a'|a,s) : a', a \in A\} > 0$. Then there exists a steady-state probability distribution function $\pi^s(\cdot|s)$ such that:*

$$\lim_{n\to\infty} \pi^b_n(a|a_0,s) \to \pi^s(a|s) \quad \textit{for all } a \in A. \tag{5}$$

Proof. See Appendix A. □

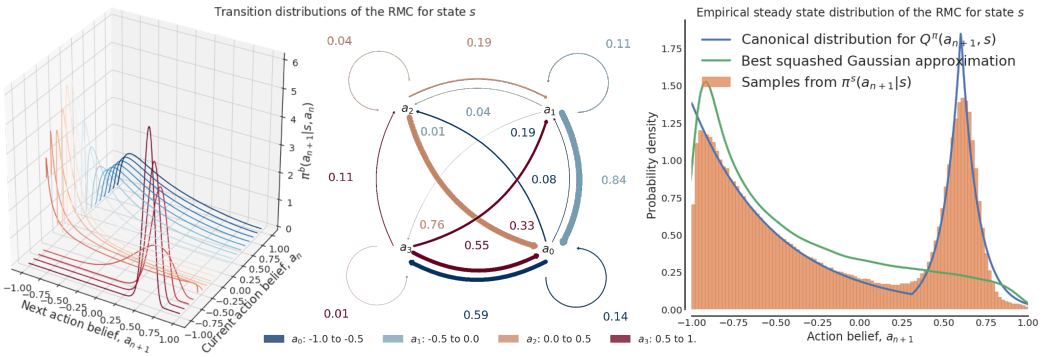

Figure 2: (**Left**) BT-policy transition probabilities and quantized RMC state diagram. (**Right**) Sample approximation of related steady-state distribution as compared to canonical distribution of the soft Q-function.

The RMC's steady-state probabilities can be interpreted as representing the distribution of agent's behavior after an *appropriate* number of reasoning steps are performed. In this work, we strive to optimize the agent's behavior following the MaxEnt RL framework described in Section 2. In particular, we consider learning a parameterized BT-policy, π^b_θ, to produce appropriate transition probabilities for each environment state such that the SS-policy, π^s_θ, from the resulting RMC optimizes:

$$\arg\max_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi^s_\theta(\cdot|s)}\left[Q^s_\phi(s,a) + \alpha H(\pi^s_\theta(a|s))\right]. \tag{6}$$

Here, Q^s_ϕ is a parameterized soft Q-function for the agent's behavior from π^s, which we learn by minimizing a squared *soft* Bellman loss utilizing delayed parameters ϕ' and samples from π^s_θ:

$$\arg\min_\phi J(\phi) = \mathbb{E}_{s,a,s'}\left[\left(Q^s_\phi(s,a) - \left(r(s,a) + \gamma\,\mathbb{E}_{a'\sim\pi^s_\theta(\cdot|s)}\left[Q^s_{\phi'}(s',a') + \alpha H(\pi^s_\theta(a'|s')\right]\right)\right)^2\right]. \tag{7}$$

In Fig. 2, we illustrate the relationship between a learned BT-policy, the corresponding SS-policy, and the soft Q-function in a 1-dimensional toy task (see App. C for details). In this example, the BT-policy is parameterized as a simple squashed Gaussian distribution, with unimodal transitions between consecutive action beliefs (Fig. 2, Left). We obtain samples of agent behavior (the SS-policy) by performing a series of reasoning steps, using the BT-policy to simulate the RMC until we approach steady-state convergence. By plotting the resulting empirical distribution of agent behavior, we see it closely matches the multi-modal, non-Gaussian canonical distribution from its soft Q-function (Fig. 2, Right). This example shows how the expressive power of agent behavior in our framework can go far beyond the BT-policy's simple parameterization, enabling for the effective maximization of complex and multi-modal MaxEnt objectives.

3.2 Learning the belief transition policy

We propose a new method to estimate the policy gradient of the BT-policy, π^b_θ, for optimizing the steady-state MaxEnt objective described in Section 3.1. We note that the gradient from Eq. 6 involves differentiating through an expectation of the steady-state policy, π^s_θ. However, π^s_θ is only *implicitly* defined, and its connection with the actual BT-policy or its parameters does not have a tractable closed-form expression. To approach this problem, we introduce a family of *n-step extensions* to the soft Q-function, $Q^s_n : S \times A \mapsto \mathbb{R}$ for $n = 0, 1, 2, \ldots$, defined as:

$$Q^s_n(s,a) = \int_A \pi^b_n(a'|a,s) Q^s_\phi(s,a')da', \quad \textit{with} \quad \nabla_\theta Q^s_n(s,a) = \mathbf{0}. \tag{8}$$

Intuitively, each *n-step soft Q-function* $Q^s_n(s,a)$ outputs the *expected soft Q-value* after performing n reasoning steps in the RMC from the initial action-belief a. However, we treat the output of each n-step soft Q-function as being *independent* of the actual parameters of the BT-policy, θ. Hence, we can interpret computing $Q^s_n(s,a)$ as simulating the RMC with a *fixed and immutable copy* of the current π^b_θ. We use this definition to provide a convenient notation in the following new Theorem that expresses the policy gradient without differentiating through π^s_θ:

[1]This is unrelated to the *steady-state* distribution for infinite-horizon MDPs considered in prior work [40].

Theorem 3.2. ***Steady-state policy gradient.*** *Let $\pi^b_\theta(\cdot|a,s)$ be a parameterized belief transition policy which defines a reasoning Markov chain with a stationary distribution given by the steady-state policy $\pi^s_\theta(\cdot|s)$. Let Q^s be a real function defined on $S \times A$, with a family of n-step extensions $\{Q^s_n\}$ as defined in Eq. 8. Suppose π^b, Q^s and their gradient with respect to θ (denoted ∇_θ) are continuous and bounded functions. Then*

$$\nabla_\theta \, \mathbb{E}_{a\sim\pi^s_\theta(\cdot|s)} \left[Q^s(s,a)\right] = \mathbb{E}_{a\sim\pi^s_\theta(\cdot|s)} \left[\lim_{N\to\infty} \sum_{n=0}^{N} \nabla_\theta \, \mathbb{E}_{a'\sim\pi^b_\theta(\cdot|a,s)} \left[Q^s_n(s,a')\right]\right]. \tag{9}$$

Proof. See Appendix A. □

Using Lemma 3.1 (*steady-state convergence*), we can approximate the policy gradient expression in Eq. 9 with an arbitrarily small expected error using a finite number of n-step soft Q-functions, i.e., N (see App. A). An intuition for this property follows from the fact that for large enough n, Lemma 3.1 implies that $\pi^b_n(a|a',s) \approx \pi^s_\theta(a|s)$ and, thus, $Q^s_n(s,a') \approx \int_A \pi^b_\theta s(a|s) Q^s_\phi(s,a) da$. Therefore, the value of each $Q^s_n(s,a')$ will be independent of the BT-policy's action a', such that $\nabla_\theta \, \mathbb{E}_{a'\sim\pi^b_\theta(\cdot|a,s)} \left[Q^s_n(s,a')\right] \approx \mathbf{0}$. In other words, each subsequent step in the RMC introduces additional randomness that is independent of a', causing a warranted *vanishing gradient* phenomenon [41] which culminates with converging to π^s_θ. Using a similar notation as Haarnoja et al. [39], we apply the reparameterization trick [38] to express the BT-policy in terms of a deterministic function $f^b_\theta(a,s,\epsilon)$, taking as input a Gaussian noise vector ϵ. This allows to rewrite the gradient in each inner expectation in the sum from Eq. 9 as:

$$\nabla_\theta \, \mathbb{E}_{a'\sim\pi^b_\theta(\cdot|a,s)} \left[Q^s_n(s,a')\right] = \mathbb{E}_{\epsilon_0\sim N(0,1)} \left[\nabla_{a_0} Q^s_n(s,a_0) \nabla_\theta f^b_\theta(a,s,\epsilon_0)\right], \tag{10}$$

where $a_0 = f^b_\theta(a,s,\epsilon_0)$. We can apply the same reparameterization for all n-step soft Q-functions, to establish a new relationship between the gradient terms $\nabla_{a_0} Q^s_n(s,a_0)$:

$$\begin{aligned}\nabla_{a_0} Q^s_n(s,a_0) &= \nabla_{a_0} \int_A \pi^b_n(a_n|a_0,s) Q^s_\phi(s,a_n) da_n = \nabla_{a_0} \int_A \pi^b(a_1|a_0,s) Q^s_{n-1}(s,a_1) da_1 \\ &= \mathbb{E}_{\epsilon_1} \left[\nabla_{a_1} Q^s_{n-1}(s,a_1) \nabla_{a_0} f^b(a_0,s,\epsilon_1)\right], \quad \textit{where, } a_1 = f(a_0,s,\epsilon_1).\end{aligned} \tag{11}$$

In Eq. 11, we purposefully omit the dependence of f^b and π^b from θ since each Q^s_n term is a *local* approximation of the RMC that does not depend on θ (as defined in Eq. 8). By recursively applying this relationship (Eq. 11) to $\nabla_{a_1} Q^s_{n-1}(s,a_1)$ and all subsequent gradient terms we obtain:

$$\nabla_{a_0} Q^s_n(s,a_0) = \mathbb{E}_{\epsilon_1,\ldots,\epsilon_n} \left[\nabla_{a_n} Q^s_\phi(s,a_n) \prod_{i=0}^{n-1} \nabla_{a_i} f^b(a_i,s,\epsilon_{i+1})\right], \tag{12}$$

where $a_i = f^b(a_{i-1},s,\epsilon_i)$ for $i = 1,\ldots,n$. By combining Eq. 10 and Eq. 12, we can thus reparameterize and express the whole sum in Eq. 9 as:

$$\begin{aligned}\nabla_\theta \, \mathbb{E}_{a\sim\pi^s_\theta(\cdot|s)} \left[Q^s(s,a)\right] &\approx \sum_{n=0}^{N} \nabla_\theta \, \mathbb{E}_{a'\sim\pi^b_\theta(\cdot|a,s)} \left[Q^s_n(s,a')\right] \\ = \mathbb{E}_{\epsilon_0,\ldots,\epsilon_N} &\left[\sum_{n=0}^{N} \nabla_{a_n} Q^s_\phi(s,a_n) \left(\prod_{i=0}^{n-1} \nabla_{a_i} f^b(a_i,s,\epsilon_{i+1})\right) \nabla_\theta f^b_\theta(a,s,\epsilon_0)\right].\end{aligned} \tag{13}$$

Eq. 13 intuitively corresponds to differentiating through each $Q^s_n(s,a')$ term by *reparameterizing the RMC*. Hence, to get a sample estimate of the policy gradient we can simulate the reparameterized RMC for N reasoning steps to obtain $a_1, \ldots, a_N$, compute each $Q^s_\phi(s,a_n)$ term, and backpropagate (e.g., with *autodifferentiation*). Following Haarnoja et al. [13, 39], we can apply Theorem 3.2 and easily extend the same methodology to estimate the MaxEnt policy gradient from Eq. 6 that also involves an extra entropy term. We include this alternative derivation in App. A for completeness.

Algorithm 1 Agent Acting

input: s, current state
$\mathbf{a_0} \sim \hat{A}$
$N \leftarrow 0$
$R^p \leftarrow +\infty$
while $R^p > 1.1$ **do**
  $\mathbf{a_{N+1}} \sim \pi^b_\theta(\cdot|\mathbf{a_N})$
  $N \leftarrow N + 1$
  Update R^p with $\mathbf{a_{1:N}}$ ▷ Eq. 16
$\hat{N} \leftarrow \rho\hat{N} + (1-\rho)N$ ▷ $\rho \in [0, 1)$
$\hat{A} \leftarrow \hat{A} \cup \mathbf{a_{1:N}}$
output: $a \sim \mathbf{a_{1:N}}$

Algorithm 2 Agent Learning

input: D, data buffer
$(s, a, s', r) \sim D$
$a_0 \sim \pi^b_\theta(\cdot|a, s')$
for $n \leftarrow 0, \lceil \hat{N} \rceil$ **do**
  $Q^s_n \leftarrow Q^s_\phi(s', a_n)$ ▷ Eq. 8
  $\epsilon_{n+1} \sim N(0, 1), \quad a_{n+1} = f^b(a_n, s, \epsilon_{n+1})$
$\nabla_\theta Q^s_\phi \leftarrow \nabla_\theta(\sum_{n=0}^{\lceil N \rceil} Q^s_n)$ ▷ Thm. 3.2
$\arg\min_\theta J(\theta)$ ▷ Eq. 6
$a' \sim a_{1:\lceil \hat{N} \rceil}$
$\arg\min_\phi J(\phi)$ ▷ Eq. 7

3.3 Action selection and temporal consistency

To collect experience in the environment, we propose to perform reasoning with the BT-policy starting from *a set* of different initial action-beliefs $\{a^0_0, ..., a^M_0\}$. We *batch* this set as a single input matrix, $\mathbf{a_0}$, to make effective use of parallel computation. To reduce the required number of reasoning steps and facilitate detecting convergence to π^s_θ, we identify two desirable properties for the distribution of action-beliefs in $\mathbf{a_0}$. In particular, initial action-beliefs should 1) be likely under π^s_θ, and 2) cover diverse modes of π^s_θ. Property (1) should logically accelerate reasoning by providing the BT-policy with already-useful information about optimal behavior. Property (2) serves to provide the BT-policy with initial information of diverse behavior, which facilitates convergence detection (Sec. 3.4) and expedites reasoning even if the RMC has slow mixing times between multiple modes. To satisfy these properties, we use a simple effective heuristic based on common temporal-consistency properties of MDPs [42, 43]. Especially in continuous environments, actions tend to have small individual effects, making them likely relevant also for environment states experienced in the near future. Thus, we propose storing past action-beliefs in a fixed sized buffer, called the *short-term action memory*, $\hat{A}$, and use them to construct $\mathbf{a_0}$. We find this strategy allows to effectively regulate the initial action-beliefs quality and diversity through the size of $\hat{A}$, accelerating convergence at negligible cost.

3.4 Detecting convergence to the steady-state policy

A key requirement for learning and acting with BT-policies, as described in Sections 3.2 and 3.3, is the ability to determine a *sufficient* number of reasoning steps (N) for the action-belief distribution to converge. Given the properties of the RMC, there exist different analytical methods that provide a priori bounds on the rate of convergence [44–46]. However, using any fixed N would be extremely limiting as we expect the BT-policy and the properties of its resulting RMCs to continuously evolve during training. Moreover, different tasks, states, and initial action-beliefs might affect the number of reasoning steps required for convergence due to different levels of complexity for the relative decision-making problems. To account for similar conditions, in the Markov Chain Monte Carlo literature, the predominant approach is to perform a statistical analysis of the properties of the simulated chain, choosing from several established convergence diagnostic tools [47–49]. Hence, we propose to employ a similar *adaptive strategy* by analyzing the history of the simulated RMC to determine the appropriate number of reasoning steps. Since we apply π^b_θ from a diverse set of initial action beliefs (see Section 3.3), we base our convergence-detection strategy on the seminal Gelman-Rubin (GR) diagnostic [50] and its multivariate extension [51]. In particular, the multivariate GR diagnostic computes the *pseudo scale reduction factor* (PSRF), a score representing whether the statistics of a multivariate variable of interest have converged to the steady-state distribution. The intuition behind this diagnostic is to compare two different estimators of the covariance for the unknown steady-state distribution, making use of either the samples *within* each individual chain and *between* all different chains. Thus, as the individual chains approach the true steady-state distribution, the two estimates should expectedly get closer to each other. The PSRF measures this precise similarity based on the largest eigenvalue of their matrix product.

For our use-case, we employ the PSRF to determine the convergence of the set of action-beliefs $\mathbf{a_{1:N}}$, as we perform consecutive reasoning steps with π^b_θ. Following [51], we calculate the average sample

covariance of the action-beliefs *within* each of the parallel chains (W) computed from a batched set of initial action-beliefs $\mathbf{a_0} = [a_0^1, a_0^2, \dots a_0^M]$:

$$\bar{a}^m = \frac{1}{N}\sum_{n=1}^{N} a_n^m, \quad W_m = \frac{1}{N-1}\sum_{n=1}^{N}(a-\bar{a}^m)(a-\bar{a}^m)^T, \quad W = \frac{1}{M}\sum_{m=1}^{M} W_m. \tag{14}$$

We compare W with an unbiased estimate of the target covariance, constructed from the sample covariance *between* the different parallel chains (B):

$$\bar{a} = \frac{1}{N \times M}\sum_{n=1}^{N}\sum_{n=1}^{M} a_n^m, \quad B = \frac{1}{M-1}\sum_{n=1}^{N}(\bar{a}^m - \bar{a})(\bar{a}^m - \bar{a})^T. \tag{15}$$

The PSRF for $\mathbf{a_{1:N}}$ is then computed from the largest eigenvalue (λ_{max}) of the product $W^{-1}B$, as:

$$R^p = \sqrt{\frac{N-1}{N} + \lambda_{max}(W^{-1}B)}. \tag{16}$$

Thus, as the individual chains approach the distribution of π_θ^s, the PSRF (R^p) will approach 1. Following Brooks and Gelman [51], we use $R^p < 1.1$ as an effective criterion for determining the convergence of $\mathbf{a_{1:N}}$. In practice, we also keep a *running mean* of the current number of reasoning steps for convergence, $\hat{N}$. We use $\lceil \hat{N} \rceil$ as the number of reasoning steps to simulate the RMC with π_θ^b when computing gradients from Eqs. 6-7. $\lceil \hat{N} \rceil$ is a *safe* choice to ensure near unbiased optimization since $R^p < 1.1$ is considered a very conservative criterion [52] and we can learn by simulating the RMC from recent actions stored in the data buffer, which are already likely close to optimal. We provide further details regarding our implementation and its rationale in App. B. We provide a simplified summary of our adaptive reasoning process for acting and learning in Algs. 1-2.

3.5 Advantages of serial Markov chain reasoning

Based on the above specification, we identify three main conceptual advantages of our serial Markov chain reasoning framework. **1. Unlimited expressiveness.** The distribution of agent behavior given by the SS-policy π_θ^s, is a mixture model with potentially infinitely many components. Thus, even a simple Gaussian parameterization of the BT-policy π_θ^b would make π_θ^s a *universal approximator of densities*, providing unlimited expressive power to the agent [53, 54]. **2. Adaptive computation.** The number of reasoning steps performed to reach approximate convergence is determined by the properties of each environment state's RMC. Hence, the agent can flexibly spend different amounts of computation time based on the complexity of each action-selection problem, with potential gains in both precision and efficiency. **3. Information reuse.** By storing past solutions to similar RMCs, we can initialize the reasoning process with initial action-beliefs that are already close to π_θ^s. This allows using the temporal-consistency properties of the MDP to exploit traditionally discarded information and accelerate agent reasoning. We provide empirical validation for these properties in Section 4.2.

4 Experimentation

4.1 Performance evaluation

We evaluate the *serial Markov chain reasoning* framework by comparing its performance with current state-of-the-art baselines based on traditional RL. We consider 6 challenging Mujoco tasks from Gym [29, 56] and 12 tasks pixel-based tasks from the DeepMind Control Suite (DMC) [30]. In both settings, we base our implementation on MaxEnt RL, replacing the traditional policy with a Gaussian BT-policy optimized with the training procedures specified in Sec. 3. Other orthogonal design choices (e.g., network architectures) follow contemporary RL practices, we refer to App. C or the code for full details. We call the resulting algorithm *Steady-State Policy Gradient* (SSPG).

We report the mean performance curves and aggregate metrics using the statistical tools from *Rliable* [55]. In particular, we compare normalized *performance profiles* [57], *interquantile mean* (IQM), and *probability of improvements* over baselines with the Mann-Whitney U statistic [58]. The reported ranges/shaded regions represent 95% *stratified bootstrap confidence intervals* (CIs) [59]. In App. D, we provide per-task results and further statistical analysis. For each experiment, we collect the returns of SSPG over five seeds, by performing 100 evaluation rollouts during the last 5% of steps.

Figure 3: Performance evaluation of SSPG and recent state-of-the-art baselines using *Rliable* [55]. We consider six OpenAI Gym Mujoco tasks [29] (**Top**) and twelve DeepMind Control tasks from pixels [30] (**Bottom**).

Mujoco suite. We evaluate on a challenging set of Mujoco tasks popular in recent literature. We compare SSPG with recent RL algorithms achieving state-of-the-art sample-efficiency performance on these tasks, which utilize large critic ensembles and high update-to-data (UTD) ratios. We consider *REDQ* [60] and *MBPO* [61] for state-of-the-art algorithms based on the traditional model-free and model-based RL frameworks. We also compare with *iterative amortized policy optimization* (IAPO) [62], in which the agent performs iterative amortization to optimize its policy distribution [63]. This procedure for action selection is more computationally involved than our agent's *reasoning process*, as it requires both evaluating the policy and computing gradients at several iterations. Yet, as IAPO is still based on the traditional policy gradient framework, its benefits are solely due to reducing the *amortization gap* with an alternative action inference procedure. To ground different results, we also show the performance of the seminal *Soft Actor-Critic* (SAC) algorithm [39], upon which all considered policy gradient baselines are based on. To account for the additional computational cost of training an agent with serial Markov chain reasoning, we use a UTD ratio that is *half* the other algorithms. On our hardware, this makes SSPG faster than all other modern baselines (see App. D).

Figure 3 (Top) shows the performance results after 100K environment steps. Individual scores are normalized using the performance of SAC after 3M steps, enough to reach convergence in most tasks. SSPG considerably outperforms all prior algorithms with *statistically meaningful* gains, as per the conservative Neyman-Pearson statistical testing criterion [64]. Furthermore, SSPG even *stochastically dominates* all considered state-of-the-art baselines [65]. We obtain similar results evaluating at 50K and 200K steps (App. D). In comparison, IAPO obtains lower performance than other non-iterative baselines while being the most compute-intensive algorithm. This indicates that, for sample-efficiency, only reducing the amortization gap beyond direct estimation might not provide significant benefits. Instead, serial Markov chain reasoning's improved expressivity and flexibility appear to considerably accelerate learning, yielding state-of-the-art performance in complex tasks.

DeepMind Control suite. To validate the generality of our framework, we also evaluate on a considerably different set of problems: 12 *pixel-based* DMC tasks. We follow the recent task specifications and evaluation protocols introduced by Yarats et al. [66]. We compare SSPG with *DrQv2* [66], the current state-of-the-art policy gradient algorithm on this benchmark, which employs a deterministic actor and hand-tuned exploration. We also compare with additional baselines that, like SSPG, are based on MaxEnt RL: *DrQ* [67], *CURL* [68], and a convolutional version of *SAC* [39].

Figure 3 (Bottom) shows the performance results after 1.5M environment steps. DMC tasks yield returns scaled within a set range, $[0, 1000]$, which we use for normalization. Remarkably, also in this domain, SSPG attains state-of-the-art performance with statistically significant improvements over all baselines. Unlike for the Mujoco tasks, the other considered algorithms based on MaxEnt RL underperform as compared to the deterministic DrQv2, a result Yarats et al. [66] attributed to ineffective exploration. In contrast, SSPG yields performance gains *especially* on sparser reward tasks where the other baselines struggle (see App. D). These results validate the scalability of our framework to high-dimensional inputs and its ability to successfully complement MaxEnt RL.

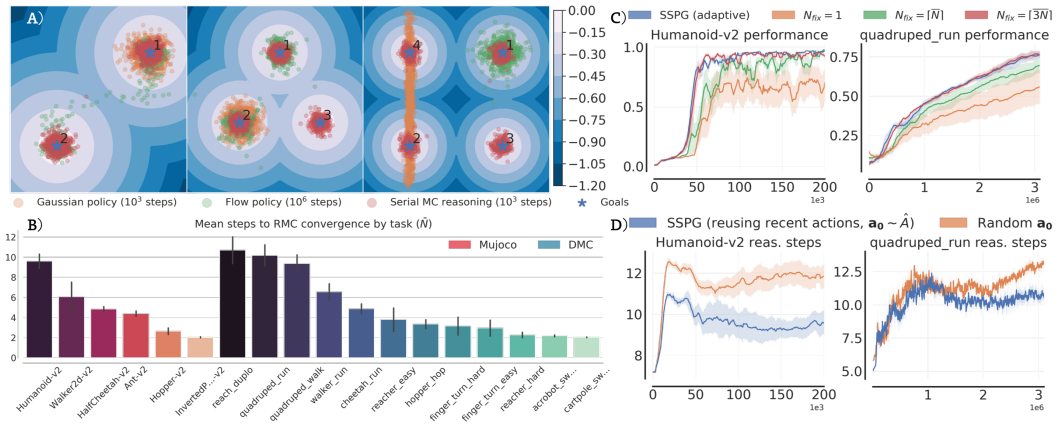

Figure 4: (**A**) Samples visualizations for learned policies in *positional bandits* of increasing complexity. (**B**) Mean number of *reasoning steps* required to reach convergence in each task with SSPG. (**C**) Mean performance ablating the adaptive strategy for detecting *reasoning convergence* and using fixed numbers of steps. (**D**) Number of *reasoning steps* throughout training with and without reusing recent actions as initial *action-beliefs*.

4.2 Properties of serial Markov chain reasoning

We test if theorized benefits of our framework (Sec. 3.5) hold in practical settings with deep networks and stochastic optimization. We provide further ablation studies and analysis of SSPG in App. E.

1. Policy expressiveness. First, we test the expressiveness of the behavior learned with SSPG using a Gaussian BT-policy. We design a series of single-step toy RL problems where the agent needs to position itself on a small 2D environment with a reward function based on unknown goal locations, which we name *positional bandits* (see App. C for details). The objective of these experiments is to isolate how our framework compares with traditional policies for MaxEnt RL to explore the environments and learn to match the true canonical distributions of returns. As displayed in Fig. 4 A, even in highly multi-modal positional bandits, the SS-policy successfully learns to visit all relevant goals with similar frequencies. Furthermore, quantizing the state space around the goals reveals that the relative RMC intuitively learns to *transition between action-beliefs that visit the different goals* as reasoning progresses, with a transition matrix matching a *cyclic permutation* (App. F). In comparison, a squashed Gaussian policy expectedly fails to capture the complexity of the canonical distribution, with samples either collapsing to a single mode or covering large suboptimal parts of the action space. We also show results for a policy based on normalizing flows [69, 70], modeled with a deep expressive network (App. C). After several attempts, we find these models require orders of magnitude more training iterations and data to learn any behavior that is more complex than a uni-modal distribution. Yet, even after increasing training by a factor of 1000, we still observe the flow policy distribution collapsing in the more complex positional bandits. We attribute our findings to training inefficiencies from a lack of proper inductive biases for flow models in the *non i.i.d.* RL problem setting [71]. In particular, as flows can assign arbitrarily low probability mass to some regions of the action space, initial local optima can greatly hinder future exploration, exacerbating coverage of the data buffer distribution in a vicious circle.

2. Policy adaptivity. Second, we examine the adaptivity of our framework for tackling decision-making problems with different complexities. We compare the average number of reasoning steps ($\bar{N}$) performed by SSPG for each task from Sec. 4.1 (Fig. 4 B). We identify a general correlation between task difficulty and reasoning computation, with complex robotic manipulation and humanoid locomotion problems requiring the most steps. By concentrating on two representative tasks, we validate the effectiveness of the reasoning process and our adaptive convergence detection strategy with an ablation study where we train SSPG using a *fixed* number of reasoning steps $N_{fix} \in \{1, \lceil \bar{N} \rceil, \lceil 3\bar{N} \rceil\}$. For the case $N_{fix} = 1$, which closely resembles traditional RL, we use double the UTD ratio to improve performance and offset any training-time gains from multi-step reasoning. As shown in Fig. 4 C, increasing N_{fix} yields clear performance improvements, validating that agents can greatly benefit from performing longer reasoning processes. Furthermore, our adaptive SSPG

attains the same performance as $N_{fix} = \lceil 3\bar{N} \rceil$ and visibly outperforms $N_{fix} = \lceil \bar{N} \rceil$. These results show how different action selection problems require different amounts of *reasoning computation* and validate the practical effectiveness of our adaptive strategy to detect steady-state convergence. We obtain analogous findings for additional tasks and values of N_{fix} in App. F.

3. Solution reuse. Last, we examine the effects of the *short-term action memory* buffer ($\hat{A}$) to sample initial action beliefs ($\mathbf{a_0}$) in two tasks. We evaluate ablating $\hat{A}$, randomly re-initializing $\mathbf{a_0}$ from a uniform distribution. While there are only minor differences performance-wise between the two approaches (App. F), sampling $\mathbf{a_0}$ from the short-term action memory considerably decreases the number of reasoning steps for convergence (Fig. 4 D). Moreover, we observe the gap in reasoning efficiency expands throughout training as the agent's steady-state behavior further improves for the target task. This result validates that a simple temporal heuristic can provide considerable efficiency benefits, amortizing the additional computational cost of our powerful new framework.

5 Related work

There have been several prior attempts to extend ubiquitous Gaussian policies [13, 39, 72, 73] with simple normalizing flows [69, 70], both to improve expressiveness [74, 75] and to instantiate behavior hierarchies [76]. Yet, the expressiveness of normalizing flows is coupled with some training challenges [71], which we show can lead to premature convergence to suboptimal solutions in RL (Sec. 4.2). Other works also considered entirely replacing policy models with gradient-free [4] or gradient-based optimization over the predicted values [77]. Marino et al. [62] similarly considered *learning* an optimizer to *infer* Gaussian behavior [28] with iterative amortization [63]. However, while all these works consider alternative modeling of agent behavior, they are still based on the traditional RL framework of representing decision-making as the output of a *fixed* process. Instead, our work entails a conceptually different approach and enables implicit modeling of agent behavior as the result of an *adaptive* reasoning process, orthogonally providing agents also with additional flexibility to scale computation based on the properties of each individual input state.

Outside RL, there have been efforts to model generation processes with parameterized Markov chains learned to revert fixed noise injection processes acting on data [78–82]. Based on this framework, diffusion models [83–85] recently achieved remarkable results for image generation [85, 86]. While applied to inherently different problem settings, these works share some conceptual resemblances with our framework and highlight the vast *scaling* potential of implicit modeling.

6 Conclusion

We introduced *serial Markov chain reasoning*, a novel framework for modeling agent behavior in RL with several benefits. We showed our framework allows an agent to 1) learn arbitrary continuous action distributions, 2) flexibly scale computation based on the complexity of individual action-selection decisions, and 3) re-use prior solutions to accelerate future reasoning. Hence, we derived *SSPG* an off-policy maximum entropy RL algorithm for serial Markov chain reasoning, achieving state-of-the-art performance on two separate continuous control benchmarks. While for problems with discrete action spaces simple multinomial policy distributions already provide unlimited expressivity, we note that the inherent computational adaptivity of our framework could still yield benefits over traditional fixed policies in these settings. Furthermore, we believe our motivation and early results provide a strong argument for the future potential of serial Markov chain reasoning, even beyond off-policy RL and simulation tasks. We provide our implementation for transparency and to facilitate future extensions at `sites.google.com/view/serial-mcr/`.

Acknowledgments

We thank Johannes Lutzeyer for providing valuable feedback on an earlier draft of this work. Edoardo Cetin would like to acknowledge the support from the Engineering and Physical Sciences Research Council [EP/R513064/1]. Oya Celiktutan would also like to acknowledge the support from the LISI Project, funded by the Engineering and Physical Sciences Research Council [EP/V010875/1]. Furthermore, we thank Toyota Motor Europe and Toyota Motor Corporation for providing support towards funding the utilized computational resources.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[3] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354, 2019.

[4] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[5] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *CoRR*, abs/1910.07113, 2019.

[6] Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021.

[7] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

[8] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897): 414–419, 2022.

[9] Amol Mandhane, Anton Zhernov, Maribeth Rauh, Chenjie Gu, Miaosen Wang, Flora Xue, Wendy Shang, Derek Pang, Rene Claus, Ching-Han Chiang, et al. Muzero with self-competition for rate control in vp9 video compression. *arXiv preprint arXiv:2202.06626*, 2022.

[10] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[11] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[14] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[16] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

[17] AR Pike. Stochastic models of choice behaviour: response probabilities and latencies of finite markov chain systems 1. *British Journal of Mathematical and Statistical Psychology*, 19(1): 15–32, 1966.

[18] Roger Ratcliff. A theory of memory retrieval. *Psychological review*, 85(2):59, 1978.

[19] Marius Usher and James L McClelland. The time course of perceptual choice: the leaky, competing accumulator model. *Psychological review*, 108(3):550, 2001.

[20] Scott D Brown and Andrew Heathcote. The simplest complete model of choice response time: Linear ballistic accumulation. *Cognitive psychology*, 57(3):153–178, 2008.

[21] Joshua I Gold and Michael N Shadlen. Neural computations that underlie decisions about sensory stimuli. *Trends in cognitive sciences*, 5(1):10–16, 2001.

[22] Leendert van Maanen, Scott D Brown, Tom Eichele, Eric-Jan Wagenmakers, Tiffany Ho, John Serences, and Birte U Forstmann. Neural correlates of trial-to-trial fluctuations in response caution. *Journal of Neuroscience*, 31(48):17488–17495, 2011.

[23] Sebastian Gluth, Jörg Rieskamp, and Christian Büchel. Deciding when to decide: time-variant sequential sampling models explain the emergence of value-based decisions in the human brain. *Journal of Neuroscience*, 32(31):10686–10698, 2012.

[24] Leendert Van Maanen, Laura Fontanesi, Guy E Hawkins, and Birte U Forstmann. Striatal activation reflects urgency in perceptual decision making. *Neuroimage*, 139:294–303, 2016.

[25] Mads Lund Pedersen, Michael J Frank, and Guido Biele. The drift diffusion model as the choice rule in reinforcement learning. *Psychonomic bulletin & review*, 24(4):1234–1251, 2017.

[26] Laura Fontanesi, Sebastian Gluth, Mikhail S Spektor, and Jörg Rieskamp. A reinforcement learning diffusion decision model for value-based decisions. *Psychonomic bulletin & review*, 26(4):1099–1121, 2019.

[27] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[28] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

[29] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[30] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[31] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

[32] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.

[33] Hagai Attias. Planning by probabilistic inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 9–16. PMLR, 2003.

[34] Emanuel Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292. IEEE, 2008.

[35] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. In *Proceedings of the 23rd international conference on Machine learning*, pages 945–952, 2006.

[36] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

[37] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[39] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[41] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[42] Sahil Sharma, Aravind S Lakshminarayanan, and Balaraman Ravindran. Learning to repeat: Fine grained action repetition for deep reinforcement learning. *arXiv preprint arXiv:1702.06054*, 2017.

[43] Will Dabney, Georg Ostrovski, and André Barreto. Temporally-extended {\epsilon}-greedy exploration. *arXiv preprint arXiv:2006.01782*, 2020.

[44] Jeffrey S Rosenthal. Minorization conditions and convergence rates for markov chain monte carlo. *Journal of the American Statistical Association*, 90(430):558–566, 1995.

[45] Peter H Baxendale. Renewal theory and computable convergence rates for geometrically ergodic markov chains. *The Annals of Applied Probability*, 15(1B):700–738, 2005.

[46] Christophe Andrieu, Gersende Fort, and Matti Vihola. Quantitative convergence rates for subgeometric markov chains. *Journal of Applied Probability*, 52(2):391–404, 2015.

[47] Mary Kathryn Cowles and Bradley P Carlin. Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[48] Stephen P Brooks and Gareth O Roberts. Assessing convergence of markov chain monte carlo algorithms. *Statistics and Computing*, 8(4):319–335, 1998.

[49] Vivekananda Roy. Convergence diagnostics for markov chain monte carlo. *Annual Review of Statistics and Its Application*, 7:387–412, 2020.

[50] Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

[51] Stephen P Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455, 1998.

[52] Dootika Vats and Christina Knudson. Revisiting the gelman–rubin diagnostic. *Statistical Science*, 36(4):518–529, 2021.

[53] D Michael Titterington, Smith Afm, Adrian FM Smith, UE Makov, et al. *Statistical analysis of finite mixture distributions*, volume 198. John Wiley & Sons Incorporated, 1985.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[55] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[56] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[57] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[58] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947. doi: 10.1214/aoms/1177730491. URL https://doi.org/10.1214/aoms/1177730491.

[59] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics*, pages 569–593. Springer, 1992.

[60] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

[61] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*. 2019.

[62] Joseph Marino, Alexandre Piché, Alessandro Davide Ialongo, and Yisong Yue. Iterative amortized policy optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[63] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018.

[64] Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of Machine Learning and Systems*, 3:747–769, 2021.

[65] Rotem Dror, Segev Shlomov, and Roi Reichart. Deep dominance-how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, 2019.

[66] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=_SJ-_yyes8.

[67] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*. 2021.

[68] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[69] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[70] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[71] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

[72] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[73] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[74] Yunhao Tang and Shipra Agrawal. Boosting trust region policy optimization by normalizing flows policy. *arXiv preprint arXiv:1809.10326*, 2018.

[75] Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pages 430–444. PMLR, 2020.

[76] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[77] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[78] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. *Advances in neural information processing systems*, 26, 2013.

[79] Guillaume Alain, Yoshua Bengio, Li Yao, Jason Yosinski, Eric Thibodeau-Laufer, Saizheng Zhang, and Pascal Vincent. Gsns: generative stochastic networks. *Information and Inference: A Journal of the IMA*, 5(2):210–249, 2016.

[80] Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. *arXiv preprint arXiv:1703.06975*, 2017.

[81] Anirudh Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational walkback: Learning a transition operator as a stochastic recurrent net. *Advances in Neural Information Processing Systems*, 30, 2017.

[82] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *Advances in Neural Information Processing Systems*, 32, 2019.

[83] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[84] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[85] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[86] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[87] Steven Orey. *Limit theorems for Markov chain transition probabilities*. van Nostrand London, 1971.

[88] Esa Nummelin. *General irreducible Markov chains and non-negative operators*. Number 83. Cambridge University Press, 2004.

[89] Gareth O Roberts and Jeffrey S Rosenthal. Small and pseudo-small sets for markov chains. *Stochastic Models*, 17(2):121–145, 2001.

[90] Esa Nummelin. *General irreducible Markov chains and non-negative operators*. Number 83. Cambridge University Press, 2004.

[91] G Ch Pflug. Gradient estimates for the performance of markov chains and discrete event processes. *Annals of Operations Research*, 39(1):173–194, 1992.

[92] Bernd Heidergott, Arie Hordijk, and Heinz Weisshaupt. Measure-valued differentiation for stationary markov chains. *Mathematics of Operations Research*, 31(1):154–172, 2006.

[93] Tom Rainforth, Rob Cornish, Hongseok Yang, Andrew Warrington, and Frank Wood. On nesting monte carlo estimators. In *International Conference on Machine Learning*, pages 4267–4276. PMLR, 2018.

[94] Don McLeish. A general method for debiasing a monte carlo estimator. *Monte Carlo methods and applications*, 17(4):301–315, 2011.

[95] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[96] Vassili Nestoridis and Vangelis Stefanopoulos. Universal series and approximate identities. Technical report, Technical Report TR-28-2007, Department of Mathematics and Statistics . . . , 2007.

[97] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[98] Edoardo Cetin and Oya Celiktutan. Learning pessimism for robust and efficient off-policy reinforcement learning. *arXiv preprint arXiv:2110.03375*, 2021.

[99] Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning. *arXiv preprint arXiv:2106.01151*, 2021.

[100] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, pages 1582–1591, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

[101] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3 for the derivation of our new framework and Section 4 for empirical evaluation.
   (b) Did you describe the limitations of your work? [Yes] We describe limitations related to our implementation in Sections B and C of the Appendix. We also discuss limitations related to computational costs in Section D of the Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section G of the Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section A of the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include code and instructions in the supplementary material of our submission. We will provide a link to our open-sourced implementation at `sites.google.com/view/serial-mcr/`.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section C of the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section D of the Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We include the licenses with the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]