Reinforcement Learning with Simple Sequence Priors

Tankred Saanum[1] **Noémi Éltető**[1] **Peter Dayan**[1,2] **Marcel Binz**[1] **Eric Schulz**[1]

[1]Max Planck Institute for Biological Cybernetics, [2]University of Tübingen
tankred.saanum@tuebingen.mpg.de

Abstract

Everything else being equal, simpler models should be preferred over more complex ones. In reinforcement learning (RL), simplicity is typically quantified on an action-by-action basis – but this timescale ignores temporal regularities, like repetitions, often present in sequential strategies. We therefore propose an RL algorithm that learns to solve tasks with sequences of actions that are compressible. We explore two possible sources of simple action sequences: Sequences that can be learned by autoregressive models, and sequences that are compressible with off-the-shelf data compression algorithms. Distilling these preferences into sequence priors, we derive a novel information-theoretic objective that incentivizes agents to learn policies that maximize rewards while conforming to these priors. We show that the resulting RL algorithm leads to faster learning, and attains higher returns than state-of-the-art model-free approaches in a series of continuous control tasks from the DeepMind Control Suite. These priors also produce a powerful information-regularized agent that is robust to noisy observations and can perform open-loop control.

1 Introduction

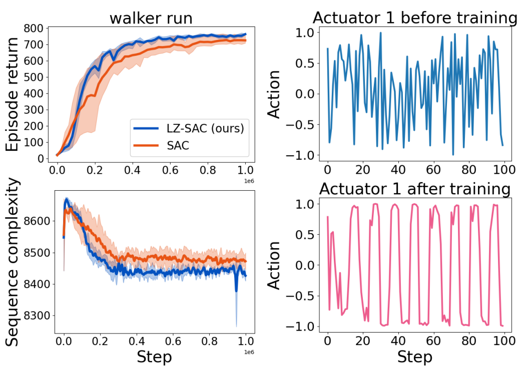

Figure 1: Action sequences produced by a bipedal walker become more compressible with learning. Our algorithm learns policies that solve tasks with simple action sequences, leading to decreased complexity and higher returns.

Simplicity is a powerful inductive bias [1–3]. In science, we strive to build parsimonious theories and algorithms that involve repetitions of the same basic steps. Simplicity is also important in the context of reinforcement learning (RL). Policies that are simple are often easier to execute, and practical to implement even with limited computational resources [4, 5]. Many control problems have solutions that are compressible: Motor behaviors like running and walking involve moving our legs in a periodic, alternating fashion (Fig. 1). Here it is the sequence of actions selected that is compressible. Sequences with repetitive, periodic elements are easier to predict and can be compressed more than sequences that lack such structure. In the current work, we augment RL agents with a prior that their action sequences should be simple: If solutions to control problems are generally compressible, one should consider only the set of *simple* solutions to a problem rather than the set

Preprint. Under review.

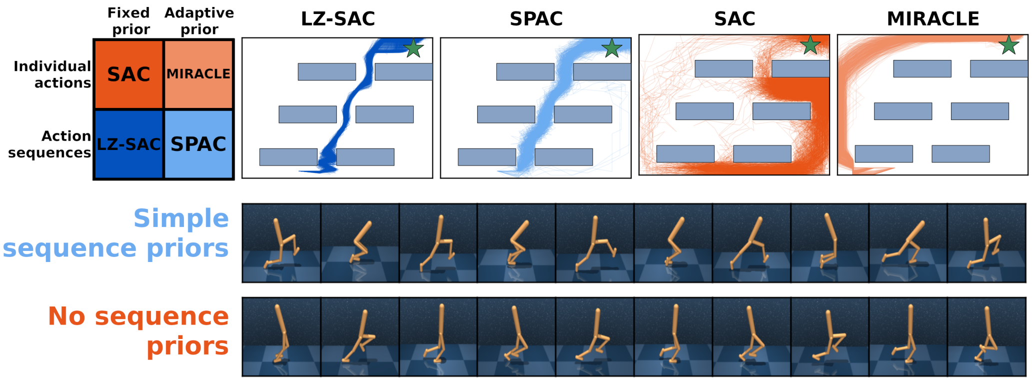

Figure 2: **Top left:** Policy regularization either incentivizes sequences or individual actions to be close to the prior. Priors may be distinct in that they stay fixed over training or change from episode to episode with learning. **Top right:** Agents need to navigate to a goal location, where the shortest path requires fine control, following a repeating pattern. After learning, SAC randomly diffuses among multiple paths. MIRACLE prefers a simple path that only goes up and then to the right. Since the optimal path is compressible (repeating UP and RIGHT in a periodic fashion), the agents with the simple sequence priors prefer this path. **Bottom:** An agent with simple sequence priors, in this case SPAC, learns simple strategies for walking, using mostly the left leg to push itself forward in a repetitive fashion.

of *all* solutions. In a series of experiments, we show that RL with simple sequence priors produces policies that perform better and more robustly than state-of-the-art approaches without such priors.

Though there are methods for regularizing policies with respect to the *individual* actions they produce [6, 7], we present a method that explicitly regularizes the *sequences* of actions used to solve a task. Our regularization incentivizes the agent to use action sequences that can be compressed with a sequence prior. If an action sequence is likely under the prior, one needs fewer bits of information to represent it [8]. We explore two types of sequence priors: *i*) Priors in the form of an autoregressive sequence model [9, 10] that learns to predict future actions based on actions that were performed in the past and *ii*) priors distilled from a pre-programmed, lossless compression algorithm. Building on the Soft Actor-Critic algorithm (SAC) [7], we introduce Lempel-Ziv Soft Actor-Critic (LZ-SAC), using an off-the-shelf compression algorithm as its prior, and Soft Predictable Actor-Critic (SPAC), using a learned sequence prior (Fig. 2).

The contributions of this paper are the following: We introduce a model-free RL algorithm for maximizing rewards with simple action sequences. In a series of continuous control tasks, we evaluate the utility of such simple sequence priors. First, we investigate whether simple sequence priors speed up policy search: In our experiments, agents with simple sequence priors consistently outperform state-of-the-art model-free RL algorithms in terms of reward maximization. This holds both in terms of learning speed and often in the final performance. Our second result is that our regularization produces an information-efficient RL agent, using fewer bits of information to solve control problems. Information-regularized models are more robust and better at generalizing [11, 4, 12]. Lastly, we demonstrate the agents' advantages in environments with noisy and missing observations.[1]

2 Related work

The idea of simplicity has received significant attention in previous work. Maximum entropy RL, for instance, augments the reward function with an entropy maximization term, effectively encouraging the agent to stay close to a simple uniform prior policy over actions [13, 14]. Many current approaches to deep RL – such as SAC [7] – rely on this principle. This concept has been further extended by models like Mutual Information Regularized Actor-Critic Learning (MIRACLE) [6] and others

[1]For videos showing behaviors learned with our algorithm, see our project website: `https://sequencepriors.github.io`

[15, 16], which use a learnable state-independent prior policy instead of the uniform prior assumed by SAC. SAC and MIRACLE both induce simplicity at the level of individual actions. In contrast, our proposed approach works on the level of action sequences.

It is not only possible to encode preferences for simplicity at the action level. Instead, simplicity can also be imposed by encouraging the agent to maintain simple internal representations – the core idea behind the information bottleneck principle [17]. Deep RL agents that rely on this principle have many appealing properties, such as improved robustness to noise, better generalization, and more efficient exploration characteristics [18–20]. Recently, [4] demonstrated how to construct RL agents that learn policies that use few bits of information by not only compressing individual observations but entire sequences of observations. In some sense, our approach can be seen as a variant of the algorithm from [4]. However, we compress sequences of *actions*, rather than sequences of observations. Thus, our regularization does not target the complexity of the sequence of internal representations, but instead the complexity of the agent's behavior, manifested in the sequence of actions selected to solve a task.

Finally, simplicity is also an important feature of natural intelligence, where it has been repeatedly argued that simplicity is a unifying principle of human cognition [2]. For instance, [21] showed that people rely on compressed policies, ultimately leading to behavioral effects such as preservation or chunking [22, 23]. Likewise, [24] demonstrated that human exploration behavior can be described by RL algorithms with limited description length, while [25] showed that compression captures human behavior in a visual search task.

3 Control with simple sequences

In this section, we demonstrate how to construct RL agents that solve tasks using simple action sequences. We start by outlining the general problem formulation. We assume that the task can be posed as a Markov Decision Process (MDP). The MDP consists of a state space $\mathbf{s} \in \mathcal{S}$, an action space $\mathbf{a} \in \mathcal{A}$, and environment dynamics $p(\mathbf{s}_0)$ and $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. The dynamics determine the probability of an episode starting in a particular state and the probability of the next state given the previous state and action, respectively. Lastly, there is the discount factor γ and a reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ that maps state-action pairs to a scalar reward term. The agent learns a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ parameterized by θ that maps states to actions in a way that maximizes the sum of discounted rewards $\mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$.

Though we want our RL agent to maximize rewards, we encourage it to do so with policies that produce simple action sequences. Inspired by previous approaches, we achieve this by augmenting the agent's objective [7, 4, 14], and search for a set of policy parameters θ that maximize reward while minimizing the *complexity* of the policy $C(\mathbf{a}_{t-\tau:t}, \mathbf{s}_t, \theta)$:

$$\max_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^{T} \gamma^t (r(\mathbf{s}_t, \mathbf{a}_t) - \alpha \underbrace{C(\mathbf{a}_{t-\tau:t}, \mathbf{s}_t, \theta))}_{\text{Complexity cost}}\right] \tag{1}$$

where the hyper-parameter α controls the trade-off between complexity and discounted rewards.

We can recover various previous approaches using this formulation. If we, for instance, set $C(\mathbf{a}_{t-\tau:t}, \mathbf{s}_t, \theta) = \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)$, we obtain maximum entropy RL algorithms such as SAC. SAC implicitly assumes a uniform prior over individual actions. An alternative to using the uniform prior in maximum entropy RL is to learn a parameterized prior over actions $p_\theta(\mathbf{a})$ based on the empirical distribution of actions the agent selects when solving the task [16, 6]. Setting $C(\mathbf{a}_{t-\tau:t}, \mathbf{s}_t, \theta) = \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) - \log p_\theta(\mathbf{a}_t)$, we obtain MIRACLE.

3.1 Simplicity with learned priors

While both SAC and MIRACLE compress sums of *individual* actions, they do not account for the structure that is present in whole action sequences. To close this gap, we present two methods for regularizing policies on the level of action sequences. For the first, we train a prior distribution $\phi_\theta(\mathbf{a}_t \mid \mathbf{a}_{t-\tau:t-1})$ to predict the agent's future actions from actions it performed in the past. We parameterize the prior as a neural sequence model. We use a causal transformer model [9, 26] to parameterize ϕ_θ, though any type of sequence model could be used in principle. We can augment the reward function to incorporate the preference for predictable action sequences as follows:

$$\tilde{r}(\mathbf{s}_t, \mathbf{a}_{t-\tau:t}) = r(\mathbf{s}_t, \mathbf{a}_t) - \alpha(\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) - \log \phi_\theta(\mathbf{a}_t \mid \mathbf{a}_{t-\tau:t-1})) \tag{2}$$

where $\mathbf{a}_{t-\tau:t-1}$ is a sequence of the last τ actions. Optimizing this objective, the agent will get rewarded for performing behaviors that the sequence model can predict better. The sequence model can learn to predict action sequences more easily if they contain structure and regularity. This has two interesting implications. *i*) The agent is incentivized to visit states where its actions will be predictable, for instance by oscillating between states in a periodic manner. *ii*) To perform actions that make it easier for the sequence model to predict future actions, for instance by performing behaviors that signal to the sequence model how it will behave in the future. We refer to this agent as the Soft Predictable Actor-Critic agent, or SPAC.

3.2 Simplicity with compression algorithms

Since the sequence model and the policy are adapting their behavior and prior towards each other, the augmented reward function will change throughout training. This plasticity can make it challenging to search for viable policies. Moreover, training a sequence model on top of the RL agent creates additional computational overhead. We, therefore, explore the possibility of instilling a simplicity preference without the use of a sequence prior that necessarily adapts over episodes.

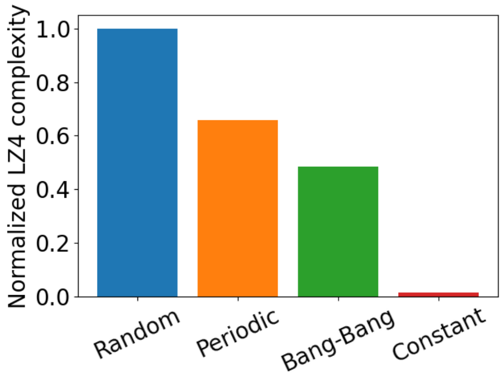

Figure 3: Some sequences are more compressible than others. A sequence of randomly generated numbers is less compressible than sequences with periodicity, sequences that only contain two types of values (also known as Bang-Bang control), or constant sequences that only contain a single number.

This second method for distilling simple sequence priors relies on off-the-shelf data compression algorithms [27]. Lossless data compression algorithms like `LZ4`, `bzip2` and `zlib` encode data into sequences of symbols from which the original data can be reconstructed or decompressed exactly. If there are repetitions, regularities, or periodicity in the data, the length of the encoded sequence can be significantly shorter than the original size of the data (Fig. 3). Relying on pre-programmed rules for data compression, this simplicity prior will not change over the course of training. Since compression algorithms like `LZ4` are fast, the sequence prior can be implemented with little computational overhead.

In this setting, we compute C using the extra number of bits needed to encode $\mathbf{a}_t$ given that we have already encoded $\mathbf{a}_{t-\tau:t-1}$:

$$\delta_t = \text{len}(g(\mathbf{a}_{t-\tau:t-1})) - \text{len}(g(\mathbf{a}_{t-\tau:t})) \tag{3}$$

$$\tilde{r}(\mathbf{s}_t, \mathbf{a}_{t-\tau:t}) = r(\mathbf{s}_t, \mathbf{a}_t) - \alpha(\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) - \delta_t) \tag{4}$$

where $g(\cdot)$ is our compression function and $\text{len}(\cdot)$ returns the length of a sequence. We use the `LZ4` compression algorithm to compute the augmented rewards and refer to this agent as the LZ-SAC agent.

3.3 Implementational details

We implement all agents as extensions of the SAC algorithm. SAC is an off-policy actor-critic algorithm that performs maximum entropy RL. We train critics to learn the augmented Q-value function $\tilde{Q}(\mathbf{s}_t, \mathbf{a}_{t-\tau:t}) = \mathbb{E}[\sum_{t=1}^{N} \gamma^t \tilde{r}(\mathbf{s}_t, \mathbf{a}_{t-\tau:t})]$ with temporal-difference learning [28]. The actors and sequence models are trained to minimize the same loss:

$$\mathcal{L} = \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_{t-\tau:t} \sim \mathcal{D}}[\alpha(\log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) - \log \phi_\theta(\mathbf{a}_t \mid \mathbf{a}_{t-\tau:t-1})) - \tilde{Q}(\mathbf{s}_t, \mathbf{a}_{t-\tau:t})] \tag{5}$$

where $\mathcal{D}$ is a replay buffer and $\mathbf{a}_t \sim \pi_\theta(\cdot \mid \mathbf{s}_t)$. The LZ-SAC actor minimizes the same loss except that $-\log \phi_\theta(\mathbf{a}_t \mid \mathbf{a}_{t-\tau:t-1})$ is replaced with the term in Eq. 3. In practice, we take the minimum of two target Q-networks to train the actor and critic. Learning is achieved by sampling experiences from a replay buffer. To calculate the augmented rewards, we further sample action sequences $\mathbf{a}_{t-\tau:t-1}$ that led to the $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t)$ tuple used for training (see Appendix A for full implementational details).

4 Simple sequence priors guide policy search

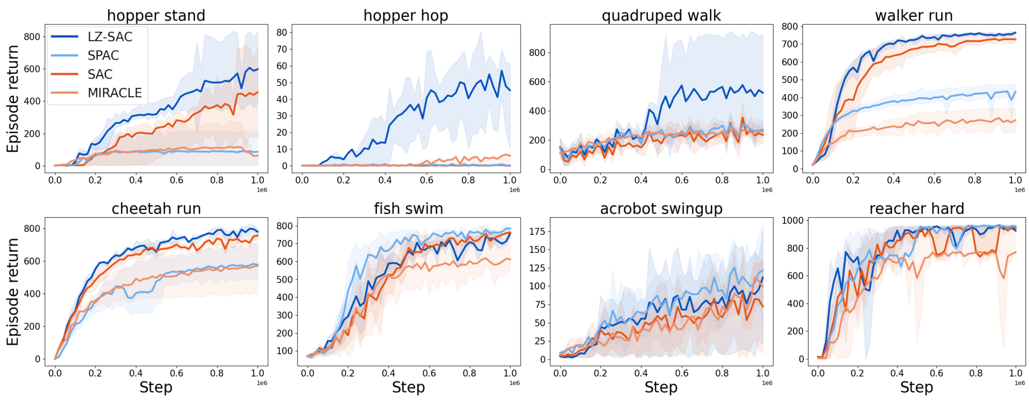

Figure 4: Learning curves of agents in the DeepMind Control suite. Overall, LZ-SAC shows the best learning speed and final performance. Lines are the average episodic returns collected in 20 test episodes with a deterministic policy, averaged over five agents trained with different seeds. Shaded regions represent 20-80 performance percentiles.

We evaluated the four agents described in Section 3 on eight continuous control tasks from the DeepMind Control Suite [29]. We trained agents for 1 million environment steps across five seeds and evaluated their abilities at regular intervals with a deterministic policy, as in [7, 30]. We tuned α for each agent and found an $\alpha = 0.1$ to give the best performance in almost all tasks. Tasks and hyperparameter fitting is described in Appendix B.

In a majority of the tasks, the LZ-SAC agent outperforms the SAC and MIRACLE agents in learning speed and often final performance (Fig. 4). At worst, the LZ-SAC agent matches the learning curves of SAC. This suggests that learning policies with simple sequence priors is indeed fruitful for policy search. We investigated whether this performance difference could simply be attributed to LZ-SAC acting more deterministically than SAC. Lowering the incentive of acting randomly for SAC did not close the performance gap, and often led to worse returns (see Appendix B.1).

In two tasks, `acrobot swingup` and `fish swim`, the SPAC agent shows a competitive advantage over the other models. However, the SPAC agent lags behind both the LZ-SAC agent and SAC agent in tasks from the `hopper`, `cheetah`, and `walker` domains. Here the policy that the SPAC agent learns achieves roughly 75% of the return of the LZ-SAC agent.

The policies learned by the SPAC agent exhibit interesting properties: The agent has discovered solutions to these tasks that effectively use fewer action dimensions than the competitors (Fig. 6): For certain actuators a_i, the agent outputs a constant value throughout the episodes. For other actuators, the agent alternates between two extreme values, like a soft bang-bang controller [31, 32]. Essentially, the SPAC agent figures out which degrees of freedom it can eliminate without jettisoning rewards. Having fewer degrees of freedom makes it easier to predict the action sequences produced by the policy. This suggests that policy compression using adaptive sequence priors is better suited in tasks with low-dimensional action spaces.

5 Simple sequence priors for information-regularized RL

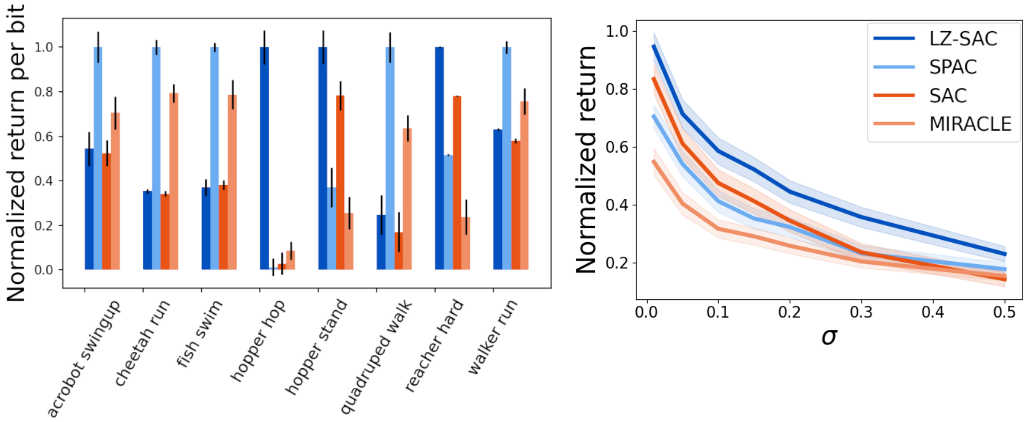

Figure 5: **Left**: Normalized return per bit attained by the agents in the eight tasks. Agents with simple sequence priors achieve better return per bit ratios. Error bars represent the standard error of the mean (SEM). **Right**: Normalized return averaged over all tasks as a function of noise scale. Error bands represent the SEM.

The expected difference in log-likelihood of the agents' actions under the policy versus the prior is an upper bound on the mutual information between states and actions [4, 11, 17]. Encouraging this difference to be low acts as an information-regularizer, the prior $p(\mathbf{a}_t|\mathbf{a}_{t-\tau:t-1})$ being the information bottleneck. We tested the information-efficiency of learned policies; that is, how much reward the agents could collect relative to the information they used to make decisions. For the experiments, we again tested the deterministic versions of the agents. Simulating 50 episodes, we computed how much reward the agents were able to collect divided by the entropy of the distribution of actions used to solve the task $\mathbb{E}\left[\frac{\sum_{t=1}^{T} r_t}{\mathcal{H}[\mathbf{a}]}\right]$ (see Appendix D.1 for details and experiments with stochastic policies). Since the policies were deterministic, this entropy term approximated the mutual information between states and actions $I(\mathbf{s};\mathbf{a})$ (see Appendix D). In the left panel of Fig. 5, we show the normalized episodic return per bit. This quantity represents how much reward the agent attains per bit of information it uses on average to make a decision over the course of the episode.

The SPAC agent attains a superior return per bit ratio in five out of eight tasks. In the other three tasks, the LZ-SAC agent attains the highest return per bit ratio. This indicates that action sequence compression is a powerful information-regularizer, allowing agents to find policies that use significantly fewer bits of information to collect a certain amount of reward than the competitor approaches.

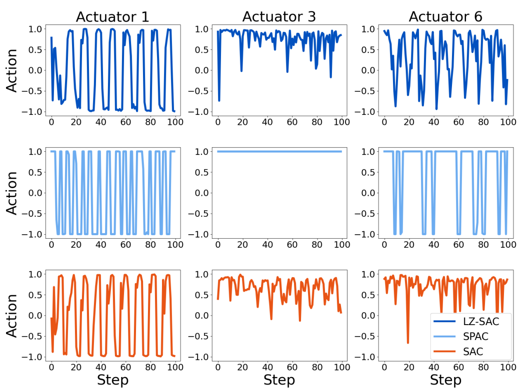

Figure 6: Time series of actions produced by the LZ-SAC, SPAC and SAC agent in the `walker run` task. Action time series of the SPAC agent exhibits a simpler periodic pattern, even outputting a constant value for its third actuator. Actuators were chosen to show qualitatively different behaviors.

6 Robustness to noise

Information-regularized policies tend to show stronger robustness to noisy observations [33, 4]: The less an agent's actions vary systematically with the state, the less will a perturbation to the agent's observation affect its actions. We assessed how observation noise affected the agents' ability to collect rewards. In the following experiments, we added Gaussian noise to the observations the agents used to execute their learned policies, $\mathbf{s}_t \leftarrow \mathbf{s}_t + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(\mathbf{0}, diag(\boldsymbol{\sigma}))$. We tested the agents on a series of noise scales $\sigma_j \in [0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5]$. The effect of noise was probed in all tasks except the `hopper hop` task, since here only the LZ-SAC agent reliably learned a policy that was better than random. Each agent was evaluated using 50 episodes for each noise-level.

We evaluated the agents based on how much reward they could collected given various levels of observation noise. Averaged over the tasks, the LZ-SAC agent showed the best ability to collect rewards when observations were perturbed with Gaussian noise (see the right panel in Fig. 5). The agents that were better at maximizing rewards showed a greater sensitivity to noise: LZ-SAC and SAC dropped to 28% and 30% of their average performance in the noise-free setting, respectively. While the LZ-SAC agent suffered greater percentage drops in return than the MIRACLE and SPAC agents, it still retained the highest performance for all noise levels. In the highest noise settings, SAC is comparable to the MIRACLE and SPAC agents, despite its generally stronger performance in the noise-free setting. This indicates that the LZ-SAC agent performed better in the noisy setting not only because the policy it learned was *generally* better at maximizing rewards, but also because of robustness properties afforded by the sequence prior.

7 Open-loop control

If simple action sequences are pervasive in policies learned with RL, these priors could provide a good starting point for policy search. To further test this claim, we evaluated how well tasks from the DeepMind Control Suite could be solved by autoregressively generated action sequences from the sequence priors themselves. In our experiments, all agents produced the first 15 actions of an episode in a closed-loop manner. We then conditioned the sequence priors with these first 15 actions and sampled actions autoregressively for the remainder of the episode. The priors of the SAC and MIRACLE agents have no autoregressive component, and generated action sequences in a memory-less manner. We approximated samples from the `LZ4` prior by discretizing the action space and sampling the next action proportionally to how low its encoding cost is, given the previous actions.

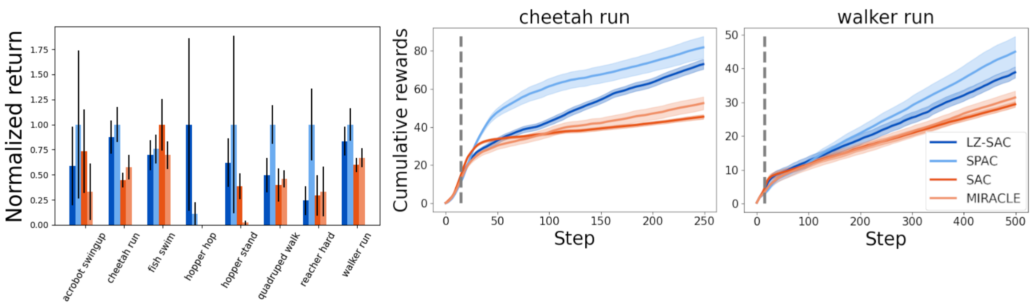

Figure 7: **Left**: Bars represent return attained in the open-loop phase exclusively. Error bars represent the SEM. The sequence prior learned by the transformer generally performs the best. Notably, the `LZ4` prior performs well in tasks solved with periodic action sequences, like `cheetah` and `walker`. **Right**: Average cumulative reward obtained by agents in the `cheetah` and `walker` tasks. Dashed lines indicate where the open-loop controls start.

The adaptive prior implemented as a transformer generally performs the best in the open-loop setting (Fig. 7, left). This is expected, as it was trained to predict behaviors that solve the tasks. In the `fish swim` task a uniform prior collects more rewards in the open-loop phase than the sequences generated by the transformer. However, increasing the number of closed-loop actions used to prompt the transformer to 25 made it surpass the performance of the uniform prior (Appendix E). This points to the importance of providing the sequence models with sufficient context to allow them to accurately predict behavior.

More interesting is the performance of the prior obtained from the `LZ4` algorithm. Not only does it perform better than chance, but even comes close to the performance of the learned sequence prior in tasks like `cheetah run` and `walker run`. By conditioning on only a few actions from the policy, autoregressively approximating samples from `LZ4`'s prior produced behaviors outperforming the non-sequential priors used by SAC and MIRACLE (Fig. 7, right). This vindicates the compressibility prior as a starting point for policy search.

8 Discussion

We have argued that simplicity is a powerful principle to guide policy search in RL tasks. Because control problems are often solved with sequences of actions that contain repeating temporal patterns, we proposed to use simple sequence priors to create effective and robust RL agents. To provide agents with a notion of compressibility, we proposed two models: One where the strategy used for compression was fixed throughout training (LZ-SAC), and one where the strategy itself could change with experience (SPAC). While the LZ-SAC agents either outperformed or matched the performance of state-of-the-art methods like SAC, the SPAC agents learned more compressible strategies, attaining more rewards while using fewer bits of information to make a decision. Furthermore, agents trained with the LZ-SAC algorithm proved to be the most robust to observation noise. Lastly, both the trained transformer model and the prior distilled from the `LZ4` algorithm could autoregressively generate rewarding behaviors in continuous control tasks.

While SPAC showed a better ability to maximize rewards than MIRACLE, returns were lower than SAC and our alternative regularization technique. This is not unexpected. The transformer always required some amount of learning to be able to predict a particular action sequence. The `LZ4` algorithm, on the other hand, could immediately provide feedback about the compressibility of the agent's action sequences without any learning. For SPAC, having to learn a sequence prior induced a stronger bottleneck, resulting in more compressed policies. This is consistent with results reported by Eysenbach et al. [4], where a learned dynamics model was used to compress sequences of states: Here compression with a learned prior led to lower returns, but a higher return per bit rate. Our results suggest that sequence compression based on off-the-shelf compression algorithms like `LZ4` are better for policy search since there is no need for learning a sequence prior from scratch.

Limitations: Action sequence compression requires either an adaptive prior, a neural sequence model, or a pre-programmed compression algorithm. The particular algorithm used for compression adds computational overhead and determines the types of action sequences that will be favored by the agent [27]. Future work should address the ways in which different compression algorithms or sequence priors affect policy regularization. Furthermore, a sufficiently sophisticated sequence model could in principle learn to predict complex action sequences. A possible extension of our work could be to further penalize the description length of the weights of the sequence model, or the compression algorithm, itself [34]. Finally, while we evaluated our algorithm on a large and diverse set of control tasks within the DeepMind Control Suite, the utility of simple sequence priors could be tested on other benchmarks. In discrete action settings, Atari games [35] would be an appropriate benchmark.

Future Directions: A central feature of simple action sequences is that they are predictable. Being able to predict one's future behavior from past behavior could allow agents to simplify and compress their representations of the state of the world [11, 4]: If the point of observing the state is to determine what action to choose, one could discard information about the state of the world simply by considering the actions that were performed previously. This suggests that simple sequence priors could be beneficial for compressing policies and internal representations jointly.

Acknowledgements

We thank the members of the Computational Principles of Intelligence Lab for feedback provided throughout the project, and Can Demircan for feedback on the manuscript. This work was supported by the Max Planck Society, the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A, and funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC2064/1–390727645.

References

[1] Carl Rasmussen and Zoubin Ghahramani. Occam's razor. *Advances in neural information processing systems*, 13, 2000.

[2] Nick Chater and Paul Vitányi. Simplicity: a unifying principle in cognitive science? *Trends in cognitive sciences*, 7(1):19–22, 2003.

[3] Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7(1): 1–22, 1964.

[4] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Robust predictable control. *Advances in Neural Information Processing Systems*, 34:27813–27825, 2021.

[5] Pedro A Ortega and Daniel A Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120683, 2013.

[6] Felix Leibfried and Jordi Grau-Moya. Mutual-information regularization in markov decision processes and actor-critic learning. In *Conference on Robot Learning*, pages 360–373. PMLR, 2020.

[7] 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.

[8] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[11] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[12] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[13] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

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

[15] Naftali Tishby and Daniel Polani. Information theory of decisions and actions. In *Perception-action cycle: Models, architectures, and hardware*, pages 601–636. Springer, 2010.

[16] Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International conference on learning representations*, 2018.

[17] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[18] Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019.

[19] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

[20] Xingyu Lu, Kimin Lee, Pieter Abbeel, and Stas Tiomkin. Dynamics generalization via information bottleneck in deep reinforcement learning. *arXiv preprint arXiv:2008.00614*, 2020.

[21] Lucy Lai and Samuel J Gershman. Policy compression: An information bottleneck in action selection. In *Psychology of Learning and Motivation*, volume 74, pages 195–232. Elsevier, 2021.

[22] Shuchen Wu, Noémi Élteto, Ishita Dasgupta, and Eric Schulz. Learning structure from the ground up—hierarchical representation learning by chunking. *Advances in Neural Information Processing Systems*, 35:36706–36721, 2022.

[23] Noémi Éltető, Dezső Nemeth, Karolina Janacsek, and Peter Dayan. Tracking human skill learning with a hierarchical bayesian sequence model. *PLoS Computational Biology*, 18(11): e1009866, 2022.

[24] Marcel Binz and Eric Schulz. Modeling human exploration through resource-rational reinforcement learning. In *Advances in Neural Information Processing Systems*, 2022.

[25] Sreejan Kumar, Carlos G Correa, Ishita Dasgupta, Raja Marjieh, Michael Y Hu, Robert Hawkins, Jonathan D Cohen, Karthik Narasimhan, Tom Griffiths, et al. Using natural language and program abstractions to instill human inductive biases in machines. *Advances in Neural Information Processing Systems*, 35:167–180, 2022.

[26] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[27] Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.

[28] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[29] 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.

[30] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10674–10681, 2021.

[31] Richard Bellman, Irving Glicksberg, and Oliver Gross. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

[32] Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 34: 27209–27221, 2021.

[33] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

[34] Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

[35] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.