Entropic Desired Dynamics for Intrinsic Control

Anonymous Author(s)
Affiliation
Address
email

Abstract

An agent might be said, informally, to have mastery of its environment when it has maximised the effective number of states it can reliably reach. In practice, this often means maximizing the number of latent codes that can be discriminated from future states under some short time horizon (e.g. [14]). By situating these latent codes in a globally consistent coordinate system, we show that agents can reliably reach more states in the long term while still optimizing a local objective. A simple instantiation of this idea, **E**ntropic **D**esired **D**ynamics for **I**ntrinsic **C**on**T**rol (EDDICT), assumes fixed additive latent dynamics, which results in tractable learning and an interpretable latent space. Compared to prior methods, EDDICT's globally consistent codes allow it to be far more exploratory, as demonstrated by improved state coverage and increased unsupervised performance on hard exploration games such as Montezuma's Revenge.

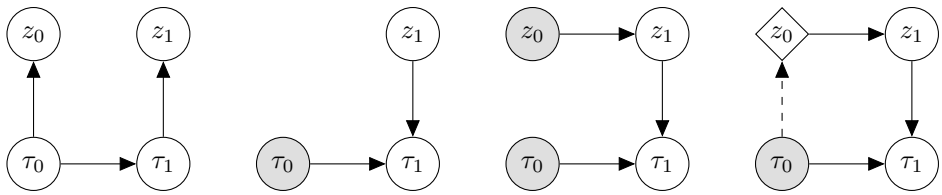
1 Introduction

Endowing reinforcement learning agents with the ability to learn effectively from unsupervised interaction with the environment, i.e. without access to an extrinsic reward signal, has the potential to make reinforcement learning practical in settings where the tasks the agent will face are initially unknown or where task feedback is expensive. The natural question is: what should the agent learn in the absence of extrinsic rewards? One appealing guiding principle is maximizing the number of states the agent can reach and to which it can reliably return.

Intrinsic control methods have shown promise in this direction. By maximizing the mutual information between a latent code z and future states reached by a policy conditioned on this code, intrinsic control methods learn to map latent codes to behaviors from which the code can be inferred. One major limitation of such approaches is that the latent codes z are usually sampled from a fixed prior distribution $p(z)$. Using a fixed prior means that such approaches are unable to learn codes that correspond to states that cannot be reached in the time horizon T, since any code can be sampled in any state. Simply increasing the time horizon T does not solve the problem since it leads to a sparser learning signal. Learning a state-dependent prior has proven to be difficult and has been shown to lead to fewer learned codes/goal states [14]. This inability to learn how to reach distant states limits the usefulness of such intrinsic control approaches.

We propose to sidestep this limitation by replacing the fixed code distribution $p(z)$ with a fixed dynamics model over codes $p(z_t|z_{t-1})$. Our algorithm, **E**ntropic **D**esired **D**ynamics for **I**ntrinsic **C**on**T**rol (EDDICT), learns to map sequences of latent codes sampled from this dynamics model to behaviors for which the state transition dynamics in the environment match the latent code dynamics. EDDICT learns to map each z_t to a state that is reachable from the state corresponding to z_{t-1}, allowing it to reach states much farther than the time horizon T using sequences of codes z. We show that even highly constrained latent dynamics (i.e. additive noise) are sufficient to both interpret latent codes in terms of their corresponding locations in state space, and encourage exploratory behavior to a far greater extent when compared to prior methods.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) (b) (c) (d)

Figure 1: (a) Prior over a particular trajectory consisting of two sub-trajectories $\{\tau_0, \tau_1\}$, and auxiliary variables $\{z_0, z_1\}$. (b) Posterior inference with independent codes, as in prior work. (c) Naive posterior inference for the sub-trajectory $\{z_1, \tau_1\}$, conditioned on the past. (d) Posterior inference with hindsight. Despite z_0 being observed, we infer z_1 based on the most likely code z_0 to have generated τ_0, using the variational reverse predictor (dashed line).

2 Notation

Our environment is a special case of a Markov Decision Process (MDP) without rewards or terminal signals: $\mathcal{M} : (S, A, P, \mu)$. S is the state space, A is the action space, $P(s_{t+1} \mid s_t, a_t)$ the conditional distribution representing the state transition dynamics when taking action $a_t \in A$ from state $s_t \in S$ and $\mu(s_0)$ the initial state distribution. For simplicity, we present our method in the episodic setting with episodes of length $T = MK$, but relax this assumption in practice. Agents interact with the environment according to policy $\pi(a \mid s; \theta)$ parameterized by θ, yielding trajectories $\tau_\pi = \{s_0, a_0, s_1, a_1, \cdots s_T\}$, and distributed as $p^\pi(\tau) = \mu(s_0) \prod_{t=0}^{T-1} P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t)$.

It will be useful for us to segment a given trajectory τ into sub-trajectories of length K, with $\tau = \{s_0, \tau_0, \tau_1, \cdots\}$, and $\tau_i = \{a_{iK}, s_{iK+1}, a_{iK+1}, \cdots a_{(i+1)K-1}, s_{(i+1)K}\}$. Note that τ_i is defined to include $s_{(i+1)K}$, but *not* s_{iK} from which a_{iK} was sampled. With a slight abuse of notation, we rewrite $p^\pi(\tau) = \mu(s_0) \prod_{i=0}^{M-1} p^\pi(\tau_{i+1} \mid \tau_i)$, with M the number of sub-trajectories per episode. [1]

Central to our method are hierarchical agents which sample a high-level goal or latent variable $z \sim p(z)$ every K steps, and interact with the environment via a conditional low-level policy $\pi_z(a \mid s) := \pi(a \mid s, z)$. Consequently, π_z can be thought of as a fixed duration option [41]. Composing our policy over options, conditional policy over actions π_z, and the transition dynamics, P, yields an augmented trajectory $\Lambda = \{s_0, z_0, \tau_0, z_1, \tau_1, \cdots\}$, with τ_i the sub-trajectory generated by unrolling $\pi(a \mid s; z = z_i)$ for K-steps. [2]

To simplify exposition, we index sequences at the timescale of sub-trajectories using index i, e.g. $\{z_i, \tau_i, z_{i+1}, \tau_{i+1}, \cdots\}$ and reserve index t for indexing sequences at the granular timescale of actions, e.g. $\{s_t, a_t, s_{t+1}, a_{t+1}, \cdots\}$. Concretely, indexing by i should be interpreted as iK, as in $s_i := s_{iK}$. For a general sequence $x = \{x_0, x_1, \cdots\}$, we define $x_{<t} := \{x_0, \cdots, x_{t-1}\}$ and by extension $\Lambda_{<i} := \{s_0, z_0, \tau_0, \cdots, z_{i-1}, \tau_{i-1}\}$.

3 Method

We would like to learn goal directed agents, which are capable of reaching any state $s \in S$, given a goal or state embedding $z \in Z$. Extending earlier work on empowerment [39, 33], Variational Intrinsic Control (VIC) [18] and related methods (e.g. [14]) propose to achieve this by learning a conditional policy π_z which maximizes $\mathcal{I}(z; \tau)$, the mutual information between the latent code z and (possibly a subset of) the resulting trajectory obtained by following π_z. Unfortunately, this objective can be difficult to train in practice as we scale both the number of options and the horizon over which the code is executed [1, 14]. Our method addresses both of these issues in a principled manner by introducing temporal dependencies between a sequence of latent codes, evolving under simple linear dynamics, which allows to break the full objective into a sum of local mutual information objectives without loss of coherence of the global code.

[1] $p^\pi(\tau_i \mid \tau_{i-1}) = \prod_{t=iK}^{(i+1)K-1} P(s_{t+1} \mid s_t, a_t)\pi(a_t \mid s_t)$, the dependence on τ_{i-1} is due to $s_{iK} \in \tau_{i-1}$.

[2] We avoid introducing new notation for the corresponding semi-MDP, as our present notation allows us to reason about sub-trajectories, for both standard policies $\pi(a \mid s)$ and conditional policies $\pi_z(a \mid s)$.

Entropic Desired Dynamics for Instrinsic Control (EDDICT) can be understood from the perspective of divergence minimization [19, 12, 26]. Concretely, we can define a prior policy $\pi_\emptyset$ which induces a distribution $p^{\pi_\emptyset}(\Lambda)$ over the space of augmented trajectories. We then learn a (posterior) policy π by minimizing the KL-divergence between $p^{\pi}(\Lambda)$ and $p^{\pi_\emptyset}(\Lambda)$.

3.1 Preliminaries: VIC as Divergence Minimization

Given a prior policy $\pi_\emptyset$, we construct a prior distribution over an augmented space (τ, z), with auxiliary variables[3] $z \in Z$, as $p^{\pi_\emptyset}(z, \tau) = p^{\pi_\emptyset}(\tau) q_w(z \mid \tau)$. The conditional $q_w(z \mid \tau)$ is a learnt predictor, with parameters $w \in \Omega$, which aims to predict z from the underlying trajectory.

We can show that a regularized version of VIC is equivalent to minimizing $\mathrm{KL}[p(z)p^{\pi_z}(\tau) \| p^{\pi_\emptyset}(z, \tau)]$ wrt. the parameters of π_z and q_w, with $p(z)$ a fixed or learnt distribution over options. This is equivalent to maximizing:

$$\mathcal{O}_{\text{ent-VIC}}(\theta, w) = \mathbb{E}_{\substack{z \sim \pi \\ \tau \sim p^{\pi_z}}} \log \frac{q_w(z \mid \tau)}{p(z)} \frac{p^{\pi_\emptyset}(\tau)}{p^{\pi_z}(\tau)} = \mathbb{E}\Big[\underbrace{\log q_w(z \mid \tau) - \log p(z)}_{ⓐ\ \mathcal{I}_q(z;\tau)} - \sum_{t=0}^{T-1} \underbrace{\log \frac{\pi_z(a_t \mid s_t)}{\pi_\emptyset(a_t \mid s_t)}}_{ⓑ\ \text{regularizer}} \Big] \tag{1}$$

In the above, $\mathcal{I}_q(z; \tau)$ refers to the variational lower-bound [2, 33] to the mutual information $\mathcal{I}(z; \tau) = \mathbb{E}[\log \frac{p^{\pi_z}(z|\tau)}{p(z)}]$, using reverse predictor q trained to approximate the true posterior $p^{\pi_z}(z \mid \tau)$. In expectation under π_z, the regularization terms correspond to a sum of KL-divergences between $\pi_z(a_t \mid s_t)$ and our prior $\pi_\emptyset(a_t \mid s_t)$[3]. The equivalence to VIC is obtained by choosing the reverse predictor $q_w(z \mid \tau) = q_w(z \mid s_0, s_T)$, which predicts z from the first and last states of the trajectory.

Since we focus on discrete action spaces, we set $\pi_\emptyset$ to a uniform distribution over actions, causing ⓑ to revert to standard entropy rewards [47]. In practice, we optimize the above objective using a value-based reinforcement learning algorithm and ϵ-greedy policies (in lieu of a Boltzmann policy), and thus drop these terms altogether. Note that the auxiliary variable perspective of VIC can also be found in Hausman et al. [22].

3.2 Incorporating Temporal Dynamics

Instead of sampling a single goal to be reached within the duration of the episode, it may be preferable to sample a sequence of codes either as *relative (or local) goals*, parameterized relative to the agent's current position, or as *way points*, a sequence of global goal coordinates which the agent should visit in sequence.

Relative vs Global Codes. Local goals can be implemented for VIC by resampling a latent code every K steps and maximizing a sum of local objectives of the type $\mathcal{I}(z_i; s_{i+1} \mid s_i)$, with option z_i initiated from state s_i. We describe these codes as having *local semantics*, as an option z_i should only be decodable in the context of the relationship between its initiation state s_i and final state s_{i+1}. In essence, each z_i represents a local displacement which the low-level policy should execute. In contrast, the strategy of sampling way points in some global frame of reference would require maximizing $\mathcal{I}(z_i; s_{i+1})$. Unfortunately, this would seem to require learning a state-dependent high-level policy $\pi(z \mid s)$, which would give higher probability to goals z_i which are reachable (in K steps) from s_i.

Ours is a hybrid of these two approaches: by specifying goals relative to previously sampled codes, in the form of a Markov chain with simple linear dynamics, EDDICT can recover codes with global semantics while avoiding the need to explicitly train a high-level policy.

EDDICT Prior. As in Section 3.1, we start by specifying a joint distribution over the set of sub-trajectories $\{\tau_i\}$ and auxiliary variables $\{z_i\}$, $_{i \in [0, M[}$. Defining $\tau_{-1} := s_0$, our prior is given by:

$$p^{\pi_\emptyset}(\Lambda \mid s_0) = \prod_{i=0}^{M-1} p^{\pi_\emptyset}(\tau_i \mid \tau_{i-1}) q_w(z_i \mid \tau_i),$$

[3]The regularization terms emerge from the fact that the transition dynamics, shared by both $p^{\pi_\emptyset}(\tau)$ and $p^{\pi_z}(\tau)$, cancel out in the above quotient.

again with $\pi_\emptyset$ a uniform distribution over actions. As we shall see, making the *a priori* assertion that z_i is conditionally independent of τ_{i-1} given τ_i will ensure that our objective breaks down as a sum of local objectives, ammenable to greedy optimization. Predicting z_i from s_{i+1} alone will then induce the appropriate global code semantics. This prior is illustrated in Fig. 1a.

EDDICT Posterior. We structure our posterior around goal-conditioned policies $\pi_z(a \mid s; \theta)$, but modified to account for the temporal structure of our prior. We incorporate temporal dependencies between the latent codes in the form of a Markov chain $p(z_i \mid z_{i-1})$ with initial distribution $p(z_0)$. Defining $p(z_0 \mid z_{-1}) \coloneqq p(z_0)$, we write:

$$p^\pi(\Lambda \mid s_0) = \prod_{i=0}^{M-1} p(z_i \mid z_{i-1}) p^{\pi_{z_i}}(\tau_i \mid \tau_{i-1})$$

and expand the negative KL-divergence. $\mathcal{O}$ is then obtained by dropping action entropy terms:

$$-\mathrm{KL}\left[p^\pi(\Lambda \mid s_0) \| p^{\pi_\emptyset}(\Lambda \mid s_0)\right] = \mathbb{E}_{p^\pi(\Lambda|s_0)} \left\{ \sum_{i=0}^{M-1} \log \left[\frac{q_w(z_i \mid \tau_i)}{p(z_i \mid z_{i-1})} \frac{p^{\pi_\emptyset}(\tau_i \mid \tau_{i-1})}{p^{\pi_{z_i}}(\tau_i \mid \tau_{i-1})} \right] \right\}$$

$$\mathcal{O}(\theta, w) = \sum_{i=0}^{M-1} \mathbb{E}_{p^\pi(\Lambda_{<i})} \underbrace{\mathbb{E}_{p^\pi(z_i, \tau_i | \Lambda_{<i})} \left[\overbrace{\log q_w(z_i \mid \tau_i) - \log p(z_i \mid z_{i-1})}^{ⓒ} \right]}_{O^{(i)}(\theta, w; z_{i-1}, \tau_{i-1})} \tag{2}$$

The objective thus breaks down as a sum of M terms, defined[4] as $\mathcal{O}^{(i)}(\theta, w) = \mathbb{E}[O^{(i)}(\theta, w; z_{i-1}, \tau_{i-1})]$. It is interesting to compare the above to $\mathcal{O}_{\text{VIC}}$. While $\mathcal{O}^{(0)}$ includes mutual information based rewards $\mathcal{I}_q(z_0; \tau_0)$ – by virtue of $p(z_0 \mid z_{-1}) \coloneqq p(z_0)$, the equivalent terms for subsequent option periods do not cleanly map onto mutual information, since entropy is computed wrt. the conditional distribution $p(z_i \mid z_{i-1})$ but z_{i-1} is missing from the reverse predictor. While ⓒ constitutes a valid lower-bound to this conditional mutual information, by virtue of the data-processing inequality, this interpretation would be misleading as the omission of z_{i-1} from the predictor was by design, in order to impose global semantics to our latent code.

3.3 EDDICT Objective

We obtain EDDICT by incorporating (i) greedy optimization, (ii) hindsight correction, and (iii) linear dynamics into the objective of Equation 2. In order to give our codes *latent goal-state semantics*, we choose to condition the reverse predictor on the last state s_{i+1} of each sub-trajectory τ_i.

Greedy Optimization. Define the *effective entropy* as the difference in log-probabilities given by the reverse predictor and the high-level policy over options (cf. ⓒ). As written, the objective aims to maximize the long term sum of effective entropies: concretely, each code z_i should seek to be entropic and discernible from s_{i+1} but also lead to states from which future options are themselves discernible. The variance of any return estimator will thus increase with the number of option periods. To avoid this issue, EDDICT optimizes Eq. 2 in a greedy-manner as:

$$\mathcal{O}_{\text{Greedy-OL}}(\theta, w) = \sum_{i=0}^{M-1} \mathbb{E}\left[O^{(i)}(\theta, w; z_{i-1}, \tau_{i-1}) \right], \tag{3}$$

where the expectation is wrt. the fixed distribution $p^\pi(\Lambda_{<i})$.

Hindsight Correction. Unfortunately, the above objective is rather brittle as the distribution over z_i is conditioned solely on z_{i-1}, and ignores the underlying state in which the code is sampled. We can improve on this open-loop formulation by reasoning in hindsight. From Eq. 3, $O^{(i)}$ is computed in expectation under $p^\pi(\Lambda_{<i})$ which includes the joint $p^\pi(z_{i-1}, \tau_{i-1} \mid \Lambda_{<i-1})$. We rewrite this joint

[4]Note that we use $O^{(i)}$ to refer to the i-th term of Eq. 2, which is a function of a particular value of z_{i-1} and τ_{i-1}. $\mathcal{O}^{(i)}$ is reserved for the expected value of $O^{(i)}$ under $p^\pi(\Lambda_{<i})$.

as $p^\pi(\tau_{i-1} \mid \Lambda_{<i-1})p^\pi(z_{i-1} \mid \tau_{i-1}) \approx p^\pi(\tau_{i-1} \mid \Lambda_{<i-1})q_w(z_{i-1} \mid s_i)$, since q_w is a variational approximation to the true posterior by construction. Incorporating this approximation to Eq. 3 yields the final objective:

$$\mathcal{O}_{\text{EDDICT}}(\theta, w) = \sum_{i=0}^{M-1} \mathbb{E}_{p^\pi(\Lambda_{<i})}\mathbb{E}_{q_w(z_{i-1}|s_i)} \left[O^{(i)}(\theta, w; z_{i-1}, \tau_{i-1}) \right]. \tag{4}$$

Concretely, when sampling $z_i \sim p(z_i \mid z_{i-1})$, we thus condition on the code most likely to have yielded state s_i, under the reverse predictor. Importantly, this objective induces a cross-entropy term between the target distribution $q_w(z_{i-1} \mid s_i)p(z_i \mid z_{i-1})$ and $q_w(z_i \mid s_{i+1})$: this ensures that predictions made from s_{i+1} are consistent with those from s_i, under our latent state dynamics.

Linear Dynamics The final piece of the puzzle concerns the choice of code distribution. We cannot employ the VIC strategy of a fixed entropic distribution, since our codes form a Markov chain. We would further like to avoid the full HRL problem, which would require us to have a parameterized high-level policy over options. Choosing an AR(1) process as the conditional code distribution satisfies both of these requirements and we thus set $p(z_i \mid z_{i-1}) = z_{i-1} + \Delta_i$, with Δ_i sampled from either an isotropic Gaussian or a uniform distribution on the disc. Another useful property of the AR(1) process is that it ensures that the marginal code entropy increases monotonically with each option period (more states visited) while the conditional entropy remains constant (same number of states reachable from any given state), as shown in Fig. 4b. Finally, hard coding the dynamics to be linear, versus learning a parametric policy over codes, naturally imposes an interpretable Euclidian topology in code space, as shown in Fig. 4a.

3.4 Algorithm

We now provide a more mechanistic view of EDDICT, as presented in Algorithm 1, specifically the concrete instantiation of the method used in experiments.

We optimize our objective using a distributed deep reinforcement learning system [13], based on Peng's Q(λ) [36] and ϵ-greedy policies. The system consists of a centralized learner, a replay buffer [31], and a set of distributed workers each interfacing with their own copy of the environment. Given the latest parameter values and current state of the environment s_i (local to each worker), actors sample z_i and generate sub-trajectory τ_i by executing π_z for K steps in the environment. The resulting (s_i, Δ_i, τ_i) is then fed back to the replay buffer, from which the learner consumes data to perform off-policy updates. Storing the initiation state s_i and offset Δ_i, instead of z_i, allows the learner to recompute the code as required using the most up-to-date version of the reverse predictor. Intrinsic rewards derived from the reverse predictor are similarly computed on the learner.

In practice, the learner maximizes $\mathcal{O}_{\text{EDDICT}}$ by summing two losses. The first implements policy iteration by minimizing the mean-squared error between a target return, computed by Peng's Q(λ) under a target network [31], and the current Q-value estimates. Our greedy optimization procedure yields a single non-zero reward, $-\log q_w(z_i \mid s_{i+1})$, which is received upon option termination. The second corresponds to the cross-entropy loss of the reverse predictor found in Eq. 4. With $q_w(z \mid s) = \mathcal{N}(f(s), 1)$ for some function f, this amounts to minimizing $\|\Delta_i - (f(s_{i+1}) - f(s_i))\|_2^2$. This loss is extremely intuitive: we train the reverse predictor such that the inferred latent state from s_i, matches the inferred state from s_{i+1} under our latent dynamics. As in [40], we found that an uninformative prior performed best in practice (despite our choice of isotropic Gaussian for the predictor), and thus sample Δ_i from a uniform distribution on the disc [5].

Concretely, we parameterize $Q_\theta(s, a, z)$ as an MLP operating on state embeddings, derived from a ResNet [23], and linear action and code embeddings. In our experiments, the reverse predictor q_w operates on the same state embeddings as the Q-function, with gradients from both objectives being backpropagated into the ResNet. Complete details of the architecture can be found in the Appendix.

[5] Our variational bound is looser as a result, since our variational posterior is not matched to the prior. Improving the modeling assumptions of the reverse predictor, e.g. by using a truncated Gaussian, is left for future work.

Algorithm 1: EDDICT

Input : Environment dynamics P, initial state s_0, behavior policy π, code predictor q_w, option period K, discount γ, code dimension d.

repeat

- $\Delta_z \sim U(\mathbb{D}^d)$
- $z \leftarrow \Delta_z + \mu_w(s_0)$ /* $q_w(Z \mid s) \sim \mathcal{N}(\mu_w(s), 1)$ */
- **for** $t \leftarrow 0 \ldots K-1$ **do**
  - $a_t \sim \pi(a|s_t, z; \theta)$
  - $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
  - $\gamma^{t+1} \leftarrow 0$ if $t = K-1$ else γ
  - $r^{t+1} \leftarrow \log q_w(z \mid s_K)$ if $t = K-1$ else 0
- Update θ with an off-policy reinforcement learning algorithm on $(a_0^{K-1}, s_0^K, r_1^K, \gamma_1^K)$,
- Update w by ascending $\nabla_w \log q_w(z \mid s_K)$
- $s_0 \leftarrow s_K$

4 Related Work

Intrinsic Control and Empowerment. EDDICT can best be thought as incorporating temporal structure into intrinsic control algorithms [18, 14, 1, 20], which build on empowerment [25, 33]. Relative Variational Intrinsic Control (RVIC) [7] also extends an intrinsic control objective, but does so by penalizing codes predictable from a single state, leading to codes representing state-agnostic behaviors. In contrast, the parameterization of our reverse predictor, along with a fixed high-level policy over options, ensure that EDDICT's codes are reachable from the states in which they are sampled while preserving global state semantics.

It is well known that the VIC objective is difficult to train when the code space is large [1]. At a high-level, EDDICT tackles this issue by breaking down this single goal into a sequence of sub-goals. This is orthogonal to the approach of Achiam et al. [1], which increases the number of available options over time. HIDIO [50] proposes an objective similar to ours (discriminator rewards over sub-trajectories, greedy-optimization), but sample options using a state-dependent high-level policy trained to maximize extrinsic rewards over the semi-MDP induced by π_z.

Skill Discovery and HRL. The notion of reusable behavior and hierarchy has a long history in the RL literature [e.g. 41]. In comparison to EDDICT existing work can be broadly categorized with respect to the signal that is used for behavior induction and the nature of the learned representation. EDDICT bears similarity to *unsupervised* skill discovery methods that induce behavior in the absence of external rewards usually for the use in downstream tasks, including [15, 35]. Other approaches learn skills or behavior representations from demonstrations provided by humans or expert policies [e.g. 16, 30, 37], while optimizing the reward for one or multiple tasks [e.g. 6, 24, 22, 43, 48, 28, 17], or via subgoals that are associated with explicit rewards in a predefined [e.g. 27, 34] or learned space [e.g. 45, 46].

Methodology. Auxiliary variables have a long history in variational inference [3, 29], as a way to obtain more expressive posteriors and serve a similar purpose in the context of EDDICT/VIC. [44] incorporates an AR(1) process in the context of HRL and skill transfer, but did so within the prior which served to regularize a set of task-specific high-level policies. In contrast, the AR(1) process in EDDICT ensures that the high-level policy samples goals which are reachable from the current state. AR(1) processes over latent temporal sequences have also been used to prevent "posterior collapse" in VAEs with powerful autoregressive decoders [38], an analogous phenomenon to "option collapse" in HRL. Hindsight reasoning has seen a multitude of applications in reinforcement learning, improving credit assignment [21], training of goal-conditioned policies in sparse reward settings [5] and off-policy learning of options in HRL [49].

5 Experiments

Here we evaluate EDDICT's learned representations and behavior, and contrast them to prior work in the space of intrinsic control (or skill discovery) methods. We assess the learned representations

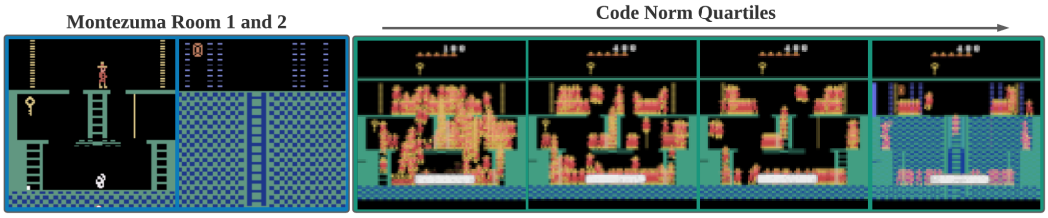


Figure 2

Figure 3: **Montezuma's revenge. (left)** Typical observations from the first (left) and second (right) rooms. **(right)** Observations from a trained EDDICT agent, sorted by $L2$-norm of corresponding code and aggregated into quartiles. Images are generated by taking the pixel-wise maximum

qualitatively by looking at how well they correspond to privileged information known to be relevant to down stream tasks. Namely, the state dimensions given in the DeepMind Control Suite [42] and the avatar coordinates in the Atari Learning Environment (ALE) [8]. We stress that this privileged information is not used during training in any way, with reverse predictors operating on the same input as the Q-function.

The quality of learned behaviors is measured in terms of exploration; we posit that EDDICT explores in the space of controllable outcomes, and that this style of exploration results in reaching many states of interest. To assess this quantitatively, we compare unsupervised behavior policies in terms of reward achievement on the Atari game Montezuma's Revenge, which is known to require sophisticated exploration in order to progress. Additionally, we look at the number of unique states visited per episode using privileged environment information (i.e. the underlying RAM states in ALE), as this is a proxy for state coverage that is agnostic to the specific reward function of the game [4]. To look specifically at the claim that EDDICT explores the *controllable* states, we also measure an estimate of the mutual information between the marginal code distribution and the marginal state distribution.

We consider the following baselines for evaluation: VIC [18], RVIC [7] and an ablated version of EDDICT. VIC refers to a scalable variant introduced in [7], that uses a fixed Categorical distribution over 50 outcomes. In the EDDICT ablation (`EDDICT-`Δ) the code proposal mechanism is simplified by substituting $\dot{z}_{i+1} := \Delta_i$ for EDDICT's $z_{i+1} := z_i + \Delta_i$. Note that the reverse predictor remains unchanged, and thus tries to predict Δ_i directly from s_{i+1}. All algorithms were implemented in the same codebase and thus share the same network architecture and reinforcement learning method.

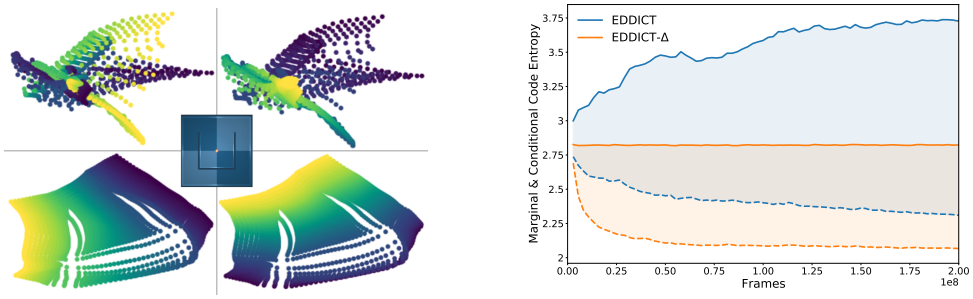
For the results on Montezuma's Revenge, we further include results for a $Q(\lambda)$ agent trained to maximize the game score (which other methods do not have access to), again matched in terms of network architecture.

5.1 Codes as Representations of State

Codes as state representations To illustrate EDDICT's ability to map codes to the controllable aspects of the environment, we have trained EDDICT on a simple continuous control task from the DeepMind Control Suite [42]. This environment comes with a set of ground truth state dimensions required to compute the dynamics, a subset of which are under the agent's control. We can thus allow EDDICT to train from raw pixels, and then evaluate the correspondence between the code values and ground truth state dimensions.

Shown in Figure 4a, is the state representation learned on a version of the Control Suite point mass domain which is modified to include 3 obstacle walls in the shape of a "U" to increase the difficulty and exploration requirements of the domain. EDDICT successfully recovers the ground truth coordinates of the point mass position (under the agent's control), but not the target position (randomly set per episode and not under the agent's control). This property of not representing what can not be controllable effectively solves the well known "noisy TV problem" [9].

Standard intrinsic control methods also have this property of only representing the controllable, but they lack any incentive to represent states unreachable in a single unroll as being distinct. This is clearly demonstrated by our ablation's performance, wherein the relationship between nearby states is much more tenuous.



(a) (b)

Figure 4: **(a) Point Mass.** 2D codes colored by ground truth x (left) and y (right) coordinates of the point-mass on a version of the environment with a 'U' shaped wall. (top) VIC (bottom) EDDICT. **(b) Montezuma's Revenge**. Estimated marginal code entropy $\mathcal{H}[z_i]$ (solid) and conditional entropy $\mathcal{H}[z_i \mid s_{i+1}]$ (dashed). Despite codes being less predictable, EDDICT achieves higher mutual information $\mathcal{I}_p(z_i; s_{i+1})$, as measured under the marginal code distribution (shaded).

Code norm as difficulty Since our desired latent dynamics consist of a sum of zero centered IID variables, the marginal code distribution will be also centered at zero, with the probability density dissipating as a function of the code norm. Assuming EDDICT manages to form a mapping between latent codes and states, this implies that less frequently visited states will have a higher code norm.

We test this hypothesis empirically, by training EDDICT on Montezuma's Revenge. In this game, the agent is represented by an avatar who can move locally around the screen, but who can easily die. When this occurs, the avatar is reborn on a platform in the middle of the screen. This is means that by construction, states where the avatar is near the center of the screen as easier to reach than those farther out. Our hypothesis would thus suggest that EDDICT should assign center states with low norm codes and more peripheral states with high norm codes. As Figure 2 shows, this is exactly what happens in practice. One interesting subtlety is that the game actually contains several rooms, each with a different background. These are exceedingly hard to reach and, as expected, this results in EDDICT assigns these states the highest code norms of all.

5.2 Control to Explore

EDDICT's representations only tell half of the story. Since codes represent states, and the marginal code entropy tries to increase monotonically, this suggests EDDICT's fixed high-level policy should result in exploratory behavior. But unlike most traditional work on the exploration problem, EDDICT focuses its exploration only on what it can control.

Measuring control Intrinsic control methods can measure their control over the environment in terms of the mutual information between a code and its downstream effects of the environment. For episodic or resampled but independent codes, this can be estimated straightforwardly as the effective code entropy (cf. ©, Eq. 2) averaged over a mini-batch. EDDICT however defines a Markov chain over codes, and thus requires us to compute entropy over the *marginal code distribution*. To do so, we fit an isotropic Gaussian to all of the codes in the batch, yielding mean and variance estimates $\hat{\mu}$ and $\hat{\sigma}^2$. Our final estimate of $\mathcal{I}_{q_w}(z_i; s_{i+1})$ is then $\mathcal{H}\big[\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)\big]$, plus the average log-prediction reward $\log q_w(z_i \mid s_{i+1})$ over the batch.

Since this is a lower bound, it is not an unbiased estimate, but the relative values should still be meaningful when comparing models of the same architecture. This metric thus allows us to answer the question: does EDDICT control the environment to a greater degree than its fixed code distribution equivalent? As Figure 4b shows, this is very much the case. Interestingly, this is true despite the codes in general being less predictable; the extra entropy from the sequential sampling more than makes up for it.

Exploring what matters In order to evaluate EDDICT's behavior policy, we must designate a proxy metric for exploration quality. For Atari 2600, game score is an obvious candidate. But since the methods under consideration learn without access to score or episode boundaries, this metric is

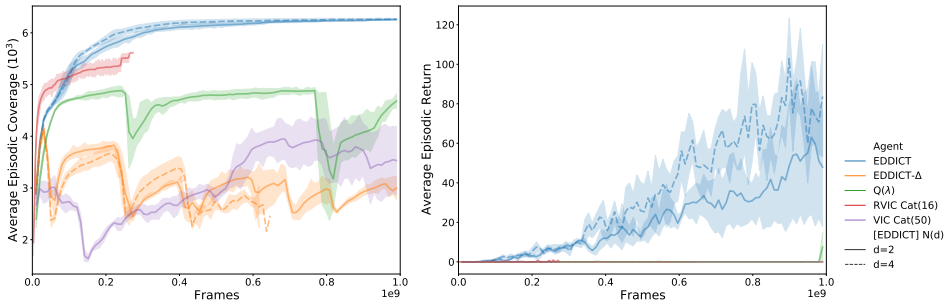


Figure 5: Comparing exploratory behavior in Montezuma's Revenge. **(right)** Game score. All methods except $Q(\lambda)$ did not have access to this during training. **(left)** Number of unique avatar positions visited. This is a proxy for coverage of the controllable states.

sometimes quite noisy. To give a more complete view of exploratory behavior, we also include two 'coverage' metrics which counts the number of unique RAM states the agent visits per episode and over its lifetime. Using the information given in [4], we only count the RAM states corresponding to the controllable avatar. We evaluate EDDICT on these 3 metrics across 6 amenable games, and as shown in Table 1, we find that in the majority of cases EDDICT outperforms RVIC, VIC and its ablation on one or more metrics. RVIC was the most competitive method, and suggests that global codes are not the only way for intrinsic control methods to yield exploratory behavior.

Of particular interest is Montezuma's Revenge, one of the hardest exploration games, as attested by the numerous reinforcement learning papers that fail to receive non-zero scores (e.g. [32, 13]). As shown in Figure 5, EDDICT outperforms other intrinsic control methods by a wide margin[6].

Game	VIC Cat(50)	RVIC Cat(16)	EDDICT	EDDICT-Δ
Berzerk	0.15, 11.5, 341	0.13, 22.6, 133	0.34, **33.5**, 427	**0.43**, 10.2, **500**
Hero	1.03, 18.4, 176	**2.78**, **29.8**, **1.4k**	1.62, 26.8, 649	0.82, 19.2, 214
Montezuma	0.39, 4, 0	0.48, 5.37, 0	**0.75**, **6.11**, **5.49**	0.59, 4.44, 0
Ms. Pacman	0.32, 1.72, 490	0.31, 1.72, 453	0.32, 1.72, 503	0.32, 1.72, **518**
Private Eye	3.95, 72.8, 345	4.85, 79.3, -560	**4.99**, **85.7**, **1.03k**	4.31, 70.8, -54.3
Seaquest	1.1, 10.9, 152	0.87, 10.9, 107	**1.46**, 10.9, **264**	1.32, 10.9, 203

Table 1: Results on a broader set of 6 Atari games at 250M frames. Each tuple A,B,C represents: (A) Episodic Coverage ($\cdot 10^3$) (B) Lifetime Coverage ($\cdot 10^3$) (C) Average return. For EDDICT-based agents, we pick the best metric across code sizes. Metrics which are best across agents, based on mean performance over 3 seeds, are shown in bold.

6 Limitations and Discussion

Endowing agents with the ability to master the environment is an important step towards more general purpose agents, as it allows learning in any circumstance without any requirement of a task specific reward function. EDDICT can both explore and control the environment by learning latent codes that make sense of states in a globally consistent coordinate system. But in terms of pure exploration, EDDICT falls short of state of the art methods that learn a single policy (e.g. [11, 10]). Understanding what these advancements mean for learning code-conditional policies is a promising future direction.

However, additive dynamics can not capture important aspects of some environments that we might wish for our agents to represent, such as dynamics that are irreversible or state-dependent. Ideally, general purpose function approximators (e.g. neural networks) could be used to specify more general dynamics, but makes learning tractable remains an important open question.

In addition to its inherent merits for environment exploration and manipulation, EDDICT's novel state to code mapping and code transition function could be used to aid local planning, or could serve as a compact representation on top of which to learn policies, or a good state similarity metric for goal-based RL, or aid in many other unlisted tasks.

[6] RVIC did not get to 10^9 frames in time for submission, this result will be amended in the final version

References

[1] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[2] D. B. F. Agakov. The im algorithm: a variational approach to information maximization. *Advances in Neural Information Processing Systems*, 16:201, 2004.

[3] F. V. Agakov and D. Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

[4] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm. Unsupervised state representation learning in atari. *arXiv preprint arXiv:1906.08226*, 2019.

[5] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[6] P. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1726–1734, 2017.

[7] K. Baumli, D. Warde-Farley, S. Hansen, and V. Mnih. Relative variational intrinsic control, 2020.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 06 2013.

[9] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[10] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[11] V. Campos, P. Sprechmann, S. Hansen, A. Barreto, S. Kapturowski, A. Vitvitskyi, A. P. Badia, and C. Blundell. Coverage as a principle for discovering transferable behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021.

[12] M. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics.

[13] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[14] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function, 2018. URL `http://arxiv.org/abs/1802.06070`.

[15] C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[16] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg. Multi-level discovery of deep options, 2017.

[17] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta learning shared hierarchies. *CoRR*, abs/1710.09767, 2017. URL `http://arxiv.org/abs/1710.09767`.

[18] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *CoRR*, abs/1611.07507, 2016. URL `http://arxiv.org/abs/1611.07507`.

[19] D. Hafner, P. A. Ortega, J. Ba, T. Parr, K. Friston, and N. Heess. Action and perception as divergence minimization, 2020.

[20] S. Hansen, W. Dabney, A. Barreto, T. Van de Wiele, D. Warde-Farley, and V. Mnih. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019.

[21] A. Harutyunyan, W. Dabney, T. Mesnard, M. Gheshlaghi Azar, B. Piot, N. Heess, H. P. van Hasselt, G. Wayne, S. Singh, D. Precup, and R. Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[22] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rk07ZXZRb`.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[24] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[25] A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

[26] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018.

[27] A. Levy, R. Platt, and K. Saenko. Hierarchical reinforcement learning with hindsight. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryzECoAcY7.

[28] A. Li, C. Florensa, I. Clavera, and P. Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByeWogStDS.

[29] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

[30] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJl6TjRcY7.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)—Proceedings of the deep learning workshop*, 2013.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[33] S. Mohamed and D. J. Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[34] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

[35] O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1emus0qF7.

[36] J. Peng and R. J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22:283–290, 1996.

[37] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *arXiv preprint arXiv:1905.09808*, 2019.

[38] A. Razavi, A. van den Oord, B. Poole, and O. Vinyals. Preventing posterior collapse with delta-VAEs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJe0Gn0cY7.

[39] C. Salge, C. Glackin, and D. Polani. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

[40] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

[41] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1), 1999.

[42] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[43] Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4499–4509, 2017.

[44] D. Tirumala, H. Noh, A. Galashov, L. Hasenclever, A. Ahuja, G. Wayne, R. Pascanu, Y. W. Teh, and N. Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl, 2020.

[45] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3540–3549, 2017.

[46] D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[47] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[48] M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, T. Hertweck, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning, 2020.

[49] M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, and M. Riedmiller. Data-efficient hindsight off-policy option learning, 2020.

[50] J. Zhang, H. Yu, and W. Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=r-gPPHEjpmw`.

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]
   (b) Did you describe the limitations of your work? [Yes] These are addressed in general in the text throughout the paper, and with a summarizing focus in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Given the foundational nature of this work, we do not believe there are any obvious negative potential social impacts – beyond those of researching AI in general, which we believe offers potential social benefits that significantly outweigh the risks.
   (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? [N/A] Our main results are not theoretical. To the extent that our method is theoretically motivated, we present derivations and layout our assumptions in the main text body and in particular in Section 2 and Section 3
   (b) Did you include complete proofs of all theoretical results? [N/A]
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)? [No] The full framework code necessary to reproduce our experiments is currently proprietary, but the key contributions should be straightforward to implement given the algorithm description in Algorithm 1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include full training and architecture details in the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] These are partially provided in the main text (and the important details live there). However, due to space constraints we provide the full and complete details regarding some evaluation error-estimates only in the Appendix.
   (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] The compute cluster we performed experiments on is rather heterogenous, and has features such as host-sharing, adaptive load-balancing, etc. It is therefore hard to give precise details regarding compute resources – however, we provide best-guess estimates in 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] Included as standard citations where relevant.
   (b) Did you mention the license of the assets? [No] This is implied and detailed in the references that we cite.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (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]