
Scaling Empowered Agents

Catherine Ji

Department of Physics
Princeton University
cj7280@princeton.edu

Vivek Myers

Department of Computer Science
UC Berkeley
vmyers@berkeley.edu

Joshua Lin*

Department of Mathematics
Princeton University

Alexandra Wingate*

Department of Mathematics
Princeton University

Sergey Levine

Department of Computer Science
UC Berkeley

Benjamin Eysenbach

Department of Computer Science
Princeton University

Abstract

Empowerment measures the capacity of an agent to actively control its environment. In this work, we explain the connection between skill learning and empowerment. While skill learning is not empowerment, we introduce a simple scalable recipe for empowerment maximization: empowerment corresponds to the number of executable skills from a state, so maximizing empowerment corresponds to reaching states that are good starting points for many skills. Our recipe applies to arbitrary underlying skill learning methods. Whereas most prior work on empowerment looks at tabular settings, this recipe enables us to study empowerment at a scale where we can realize theoretically-predicted but empirically-elusive properties of empowerment (e.g., gathering resources, environment modification, survival). Experiments show that empowerment maximization unlocks interesting emergent properties, such as building a house in Crafter.¹

1 Introduction

Empowerment, a measure of agency, has been speculated as a unified mathematical explanation for myriad properties of intelligent agents, ranging from building causal models (Abel et al., 2025; Lidayan et al., 2025; Yiu et al., 2026), acquiring resources (Mohamed and Jimenez Rezende, 2015), maintaining homeostasis (Klyubin et al., 2008; Salge et al., 2014b), to explanations of how children explore (Lidayan et al., 2025). Prior work argues that empowerment should be especially important in open-ended environments, but does not maximize empowerment from scratch in such settings (Lidayan et al., 2025; Song et al., 2026). Despite decades of research studying the behavior of empowerment-maximizing agents, there has not yet been empirical demonstrations of algorithmic empowerment maximization in the “big world” (Javed and Sutton, 2024) – challenging, large settings where behaviors like survival and resource collection become tough. It is only in such tasks where we expect to see rich emergent behaviors from empowerment maximization.

*Equal contribution.

¹Website and code at <https://empowerment-geometry.github.io/scaling-empowerment/>.

Agents should successfully interact with an environment that they cannot fully describe. For example, even if a young child cannot fully explain all of the intricacies of a game like Minecraft, the child will still become empowered by defeating enemies and building a house, reaching an empowered state from which they know how to do many things. However, maximizing empowerment is a tricky computational problem. First, empowerment maximization requires solving for empowerment at particular contexts. If the context is a state, this means solving for empowerment in all of reachable state space. Second, solving for the empowerment at a particular context is a challenging problem and requires not only modeling the relationship between actions and outputs (e.g. a transition model), but also finding the actions that maximize control. While tractable using standard algorithms in tabular settings, the combination of these difficulties means canonical methods like Blahut-Arimoto cannot exactly solve this computational problem in high-dimensional settings. In the last decade, methods like VIM (Mohamed and Jimenez Rezende, 2015), variational empowerment (Karl et al., 2017), and LatentGCE (Zhao et al., 2021) aimed to address this computational problem using deep learning, and were the first to demonstrate learning-based empowerment maximization in simple tabular and continuous settings. However, there has not yet been a stable empowerment-maximization method that builds on the foundation of these works and scales to large continuous or discrete settings. This gap is important to resolve, given that large settings are precisely the right environments for empowerment maximization – where an agent must execute intelligent behavior despite having a limited view of a complex world.

In this work, we scale empowerment maximization to a regime where theorized intelligent behaviors can emerge. Our starting point is the observation that empowerment maximization is closely related to unsupervised skill discovery (USD) methods. The goal of empowerment maximization is to reach states from which it is easy to reach a diverse set of states. Relatedly, USD methods aim to learn a diverse set of different “skills”. However, unlike prior empowerment-maximization methods that have struggled to scale, modern skill discovery methods can learn interpretable, diverse behaviors in continuous, high-dimensional tasks. As we will show, a particular class of USD methods called mutual information skill learning (MISL) measures the point-wise empowerment: at any particular state, the number of different skills an agent can execute from that state lower bounds its empowerment. In other words, an agent is empowered at states where it has many options, or skills. Thus, while skill learning methods do not directly learn policies that maximize empowerment, we can use scalable skill learning methods to estimate the empowerment of a particular state; we can then train another policy that ascends the empowerment landscape to get to states where it has many options.

Our work has two key contributions. **First**, we provide a general recipe to convert scalable skill-learning methods into empowerment maximization methods and show our methods scale when baselines cannot. **Second**, we demonstrate our method in challenging, high-dimensional settings where empowerment-maximizing behavior becomes nontrivial, and analyze the behavior of empowered agents including resource gathering, survival, and environment modification. For example, early in training in Crafter, empowerment-maximizing agents learn to build houses at night when enemies appear and make and use tools. **Our work shows that scalable empowerment maximization is possible, demonstrating interesting emergent behaviors in complex, open-ended setting.**

2 Related Work

Intrinsic Motivation Exploration Methods. Intrinsic motivation enables exploration without external information about the Markov decision process (MDP) (Gruaz et al., 2025). Prior intrinsic motivation methods reward curiosity (Pathak et al., 2017), novelty (Bellemare et al., 2016), information gain (Houthoofd et al., 2016), and prediction error (Burda et al., 2019). Empowerment instead rewards agency and control (Mohamed and Jimenez Rezende, 2015), a motivation thought to explain human behavior and desired in complex, open-ended settings (Dai and Gopnik, 2025; Gopnik, 2024; Lidayan et al., 2025; Yang et al., 2026; Yiu et al., 2026). Prior works predicts that empowerment maximization should generate behaviors such as (1) environment modification (Salge et al., 2014a), (2) resource gathering (Mohamed and Jimenez Rezende, 2015), and (3) survival. While these emergent behaviors have

been previously demonstrated in simple experiments, we present scalable empirical evidence in large, open-ended settings.

Potential and Effective Empowerment. As originally introduced, empowerment quantifies the influence an agent’s open-loop actions can exert on future states (Klyubin et al., 2005b; Salge et al., 2014b). Two complementary forms of empowerment are useful (Capdepuy, 2011; Ji et al., 2026). *Potential empowerment* measures the channel capacity between a sequence of actions and future observations conditioned on a starting state, measuring the *maximal possible* control from a particular context. In contrast, *effective empowerment* measures how many bits a particular distribution over open-loop controls communicates to the future. Potential empowerment is a scalar property of the context, while effective empowerment is also a function of the controls.

Prior Deep Learning Methods for Estimating and Maximizing Potential Empowerment. VIM maximizes a variational lower bound on open-loop empowerment by learning a decoder that infers action sequences from policy outcomes (Mohamed and Jimenez Rezende, 2015). Later empowerment maximization works like LatentGCE alternatively learn latent forward models to estimate the same empowerment, but relies on sufficient coverage over actions sequences in high-empowerment regions (Zhao et al., 2021). This requirement becomes increasingly demanding in higher-dimensional tasks. In contrast, we study *closed-loop* empowerment (Gregor et al., 2017; Tiomkin et al., 2025), which is the channel capacity of an information channel from a collection of closed-loop policies to outcomes. Thus, a state is empowering if an agent can execute a diverse repertoire of reactive policies from that state. This closed-loop formulation means empowerment measures what a standard reinforcement learning (RL) agent can actually do and naturally connects with unsupervised skill-discovery methods.

Unsupervised Skill Discovery. MISL methods aim to learn diverse policies that maximize a mutual information (MI) between skill latents $z \in \mathcal{Z}$ and the outcomes of latent-conditioned policies (i.e. $\pi(a \mid s, z)$). Modern MISL methods learn policies that optimize approximations to MI objectives over one-step transitions $I(Z; S, S')$ ((Park et al., 2026, 2024; Zheng et al., 2025), to second order for METRA’s representation objective), long-horizon transitions $I(Z; S_\tau \mid S_0)$ for termination time τ (Gregor et al., 2017), occupancies $I(Z; S)$, (Eysenbach et al., 2019) and the conditional MI $I(Z; S' \mid S)$ (Sharma et al., 2020). Typically, these methods learn a classifier that identifies skills from outcomes and use this classifier as a reward, so that the expected value function is a variational lower bound on the MI. These methods scale to high-dimensional locomotion and pixel-based tasks.

Skill Learning and Effective Empowerment. The MI maximized in MISL methods are related to closed-loop effective empowerment. Skills are *not* potential empowerment-maximizing policies, but they can provide estimates of the potential empowerment. Furthermore, while recent works have called attention to this relationship between skill-learning and empowerment (Levy et al., 2025), to our knowledge, there has not yet been work that exploits this connection to do scalable potential empowerment maximization. We close this gap by adapting modern MISL methods to maximize potential empowerment in large, open-ended settings.

3 Defining Empowerment

Our work focuses on a controlled Markov process (i.e., an MDP without a reward function) with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, and environment dynamics $p(s' \mid s, a)$. Our goal is to learn a *potential empowerment* policy π_{pot} that visits states with high potential empowerment $\mathcal{E}_{\text{pot}}(s)$:

$$\pi_{\text{pot}} \in \arg \max_{\pi \in \Pi} \mathbb{E}_{\pi} \left[\sum_t \gamma^t \mathcal{E}_{\text{pot}}(s_t) \right]. \quad (1)$$

Empowerment $\mathcal{E}_{\text{pot}}(s_0)$ is typically (Salge et al., 2014b) defined as the maximal conditional MI between behavior Z and future states S_+ , given an initial state s_0 : $\mathcal{E}_{\text{pot}}(s_0) = \max_{p(A)} I(A; S_+ \mid S_0 = s_0)$.

However, prior work differs on what exactly constitutes the behavior Z : is it the immediate next action (Salge et al., 2014b, Sec. 4.4.2), a sequence of actions (Klyubin et al., 2005b), or the index of a closed-loop policy (Gregor et al., 2017; Tiomkin et al., 2025)? Prior work also differs in how precisely the future state S_+ is sampled. This paper will adopt an expressive definition of empowerment that measures how behaviors described by closed-loop policies control states *many* steps into the future.

To make the definition of behavior precise, we will consider a z and context-conditioned policy $\pi(a | s, z; s_0) \in \Pi_{\mathcal{Z}}$, with s_0 taken to be the starting context (in our case, a starting state s_0) of the behavior. For a specific “skill vector” z we will refer to this policy as a “skill”. Importantly, the context state s_0 and the current state s are two different objects – the context determines the “start point” of the skill, while current state tells the agent what to do now.

Let $p(z)$ be a distribution over these skills, which we will assume to be uniform (without loss of generality (Ji et al., 2026)). By introducing $p(z)$ as a distribution over skill vectors, we define a distribution over the space of policies. The term “skill” comes from the fact that this distribution over the space of policies may assign zero probability to almost all policies, so the skills are the set of policies with non-zero probability (Eysenbach et al., 2022). For each of these skills $\pi(a | s, z)$, we define the discounted state occupancy measure (DSOM) as:

$$p^\pi(s_+ = s | s_0, z) \triangleq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p^\pi(s_{t+1} = s | s_0, z). \quad (2)$$

Averaging over all skills gives

$$p^\pi(s_+ | s_0) = \mathbb{E}_{p(z)} [p^\pi(s_+ | s_0, z)]. \quad (3)$$

We now define the *potential* empowerment using the DSOM definitions:

Definition 1 (Potential empowerment). *Let $p(z)$ be given, and let $S_+ \sim p^\pi(s_+ | s_0, z)$ be a sample from the DSOM of skill $\pi(a | s, z)$. Then, the potential empowerment of state s is the conditional MI between the skill vector z and the future state S_+ :*

$$\mathcal{E}_{\text{pot}}(s_0) \triangleq \max_{\pi \in \Pi_{\mathcal{Z}}} \underbrace{I^\pi(Z; S_+ | S_0 = s_0)}_{\triangleq \mathcal{E}_{\text{eff}}(\pi, s_0)}. \quad (4)$$

The potential empowerment can be understood as the maximum number of bits that the skill vector Z can broadcast into the future (S_+), starting from state s_0 . Our goal is to learn a policy π_{pot} that uses this empowerment as a reward, effectively navigating to states where it can broadcast many bits into the future (i.e., exert a strong influence on future outcomes).

Estimating potential empowerment requires solving the inner maximization problem over $\Pi_{\mathcal{Z}}$. We will refer to the objective for this inner optimization problem as *effective empowerment*, $\mathcal{E}_{\text{eff}}(\pi, s_0)$. **We observe that this inner optimization problem is similar to that solved in prior works on MISL** (Gregor et al., 2017; Zheng et al., 2025). Whereas potential empowerment measures how good a *state* is, effective empowerment measures how good a *set of policies* is from a particular state. Of course, if this inner optimization problem is only solved approximately, then we obtain a lower bound on the empowerment.

4 Skills Estimate Empowerment (SEE): A Scalable Algorithm for Maximizing Empowerment

In this section, we introduce a scalable, online algorithm (SEE) to maximize empowerment (Eq. 1) by exploiting the connection between skill learning and empowerment. This connection between learning skills (Sec. 4.1) and maximizing potential empowerment (Sec. 3) suggests a two-policy algorithm.

The skill policy π_{eff} estimates a lower bound $\widehat{\mathcal{E}}_{\text{pot}}(s)$ on the state-wise potential empowerment, while the potential policy π_{pot} maximizes this estimate as an intrinsic reward. The effective and potential empowerment policies support one another: the skills π_{eff} measure the controllable futures available from a state, while the potential empowerment policy π_{pot} moves the agent toward states from which the agent can exert control.

4.1 The Inner Optimization: Learning Skills

We modify prior MISL methods (Zheng et al., 2025) to maximize effective empowerment, which lower bounds potential empowerment by Definition 1. Recall that the expectation of a classifier that identifies skills from transitions over the course of the trajectory is a variational lower bound on the MI. This mathematical observation implies that setting the reward to the skill classifier gives a value function that, in expectation, is a lower bound on the MI. SEE builds on contrastive successor features (CSF) (Zheng et al., 2025), a recent state-of-the-art skill-learning method, to approximately lower bound the effective empowerment $\mathcal{E}_{\text{eff}}(\pi, s_0) = I^\pi(Z; S_+ | S_0 = s_0)$ for a fixed starting state s_0 . CSF estimates the one-step MI $I(Z; S, S')$ from a classifier that identifies continuous skills from one-step transitions $s \rightarrow s'$. To capture long-horizon dependence between skills and future states, we instead train the classifier on transitions separated by a geometric random variable lengths of time $\Delta \sim \text{Geom}(1 - \gamma)$:

$$f_\phi(s_0, s_+, z) - \log \mathbb{E}_{\tilde{z} \sim p(z)} \left[e^{f_\phi(s_0, s_+, \tilde{z})} \right] \approx \log \frac{p^\pi(z | s_0, s_+)}{p(z)}. \quad (5)$$

Following representation parameterizations in CSF (Zheng et al., 2025), Metric-aware Abstraction (METRA) (Park et al., 2024), and GenDa (Park et al., 2026), SEE parameterizes the classifier logits f_ϕ with a bilinear form of the learned representations ϕ . This parameterization means that skills are only distinguishable if learned representations linearly separate over the course of skill execution. Prior work find that this parameterization is important to generate sustained, space-filling behaviors (Park et al., 2024). SEE utilizes InfoNCE (van den Oord et al., 2018) to estimate the classifier. Additional discussion on this parameterization and its implications are in Section A.1.

Accumulating this classifier over the trajectory gives, in expectation, a variational lower bound on the MI. For a skill and start-state conditioned policy $\pi(a | s, z; s_0)$, define the effective empowerment Q- and value function as

$$Q_{\text{eff}}(s, a; s_0, z) \triangleq (1 - \gamma) \mathbb{E}_{\pi(a | s, z; s_0)} \left[\sum_{k=0}^{\infty} \gamma^k \log \frac{q(z | s_0, s_{t+k+1})}{p(z)} \middle| s_t = s, a_t = a \right] \quad (6)$$

$$V_{\text{eff}}(s; s_0, z) \triangleq \mathbb{E}_{\pi(a | s, z; s_0)} [Q_{\text{eff}}(s, a; s_0, z)]. \quad (7)$$

Then, in expectation over a uniform skill source $p(z)$ initialized at the start-state s_0 , we have that

$$\mathbb{E}_{p(z)} [V_{\text{eff}}(s_0; s_0, z)] \leq \mathcal{E}_{\text{eff}}(\pi, s_0), \quad (8)$$

so the expected value function lower bounds the effective empowerment. Potential empowerment then uses a Monte-Carlo estimate of $\mathbb{E}_{p(z)} [V_{\text{eff}}(s_0; s_0, z)] \approx \frac{1}{K} \sum_{k=1}^K V_{\text{eff}}(s_0; s_0, z_k)$, with $z_1, \dots, z_K \stackrel{\text{iid}}{\sim} p(z)$, to get an estimate of the pointwise potential empowerment:

$$\hat{\mathcal{E}}_{\text{eff}}(\pi, s_0) = \mathbb{E}_{p(z)} [V_{\text{eff}}(s_0; s_0, z)] \leq I(Z; S_+ | S_0 = s_0) \leq \mathcal{E}_{\text{pot}}(s_0). \quad (9)$$

4.2 Practical Considerations

Conditional and Joint MI Classifiers. While empowerment is theoretically a conditional MI, in practice, we found that training policies with joint MI classifiers that identify skills from a current state s_t and future state s_+ from that state led to stronger performance in continuous benchmarks. Otherwise, the classifier aggressively overfit to the context s_0 . The skill reward still uses the true skill’s starting state s_0 .

We include implementations of SEE using the joint MI classifier and a standard CMI classifier. By default, SEE uses this joint classifier in continuous settings and the conditional classifier in Craftax-Classic (Section A.2).

Limiting empowerment context. As noted in (Abel et al., 2025; Klyubin et al., 2005b), empowerment depends on an agent’s particular perception. Otherwise, an agent may obtain high empowerment by perturbing controllable but meaningless variables. We therefore run experiments that restrict variables over which empowerment is defined. This restriction determines which changes in the state are salient to the skill, without restricting the information available to the policy for control. Thus, the classifier may receive different degrees of freedom from the full-observation policy and critic.

Two-stage rollout. For large state spaces, we cannot simply learn skills for every starting state s_0 and then maximize the potential empowerment over the constructed field. Instead, we consider a two-stage rollout scheme: within each episode of data collection, we first roll out the potential empowerment policy and then roll out one of the effective empowerment policies (i.e., sample a skill z and execute $\pi_{\text{eff}}(a \mid s, z; s_0)$) (Sections A.2 and A.4). We can interpret skills as local probes of empowerment, where the potential empowerment policy ascends the empowerment landscape estimated by these probes.

Method Summary. We adapt CSF, a scalable MISL method, to learn skills, which gives us (a lower bound on) the potential empowerment of states $\mathcal{E}(s)$. Then, we train a second potential empowerment policy using the estimated potential empowerment as a reward. This policy navigates to states *from which* it can exert control.

5 Experiments

We (1) show scalability of the method, (2) carefully ablate and analyze all of the salient design decisions (and provide open-source implementations for future development), and (3) interpret the resulting emergent behaviors. All training setups and hyperparameters are in Sections B.1 and B.2.

Environments. To test whether the method works in large, open-ended settings, we implement our method in Craftax-Classic (Matthews et al., 2024) (JAX-accelerated Crafter, Fig. 1d). Crafter is a large and partially observed setting with a 1345-dimensional symbolic observation and 17 discrete actions consisting of crafting, resource collection, fighting mobs, and eating/drinking. In this environment, we define the empowerment degrees of freedom to be over the agent inventory. The reason for this choice is that we do *not* wish for an agent to be empowered over features like e.g. health, hunger, and thirst, which would imply having a lot of control over these precise status values (for instance, a skill that *decreases* health).

To test whether the method works in continuous settings, we chose two high-DoF benchmarks that capture locomotion and manipulation. The AntSoccer environment consists of an ant and a ball in a closed arena, where an ant can freely kick the ball (Park et al., 2025). Within the AntSoccer environment, we consider the setting where empowerment is defined over all 42 degrees-of-freedom (e.g. joint angles, xy position, ball position, “Ant”) and strictly the ball position (“AntSoccer”). BuilderBench (Ghugare et al., 2026) consists of a gripper with cubes. We consider an environment with 3 cubes in the corner of a bounded play area (Fig. 1a, white corners denote the boxed play area), where empowerment is defined over the 9 degrees-of-freedom that denote the cube positions.

Baselines. We consider two baselines that learn potential empowerment policies: LatentGCE (Zhao et al., 2021) and VIM (Mohamed and Jimenez Rezende, 2015). Both of these methods learn policies that navigate to high open-loop empowerment regions (Section C). We omit standard MISL works because they optimize effective empowerment policies from a fixed initial state distribution and do not learn a potential empowerment policy, though these methods bear similarity to our skill-switching ablation.

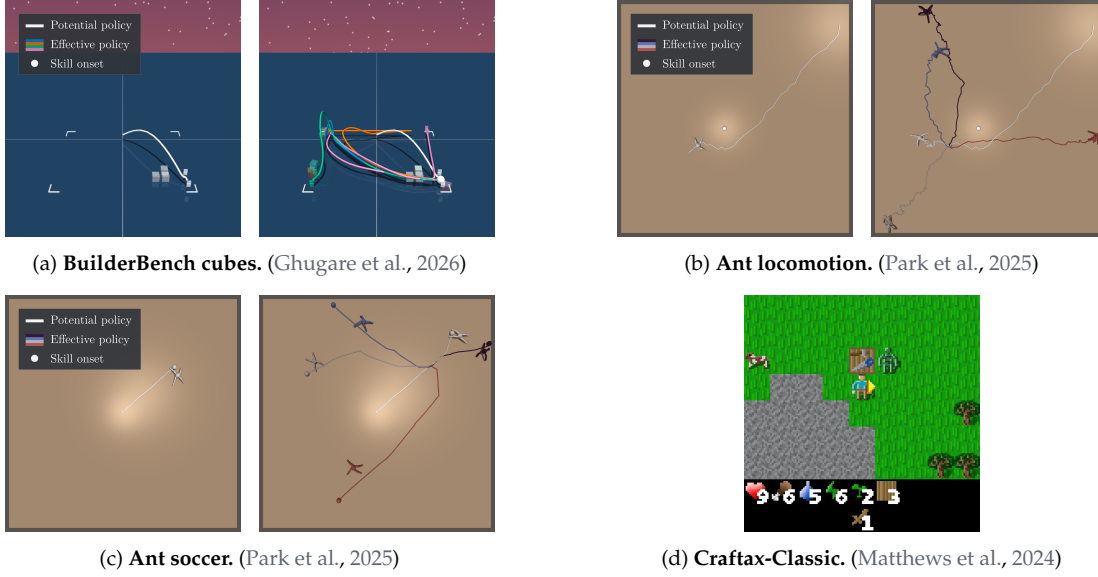


Figure 1: **Empowerment-maximizing behavior across environments.** (a) In the BuilderBench task, a gripper interacts with blocks in an open-ended setting. The potential empowerment policy is visualized with the white rollout, navigating to the blocks and staying there. Then, the effective empowerment policies, or skills, knock the blocks in different directions (colorful lines and blocks). (b) In the first version of the AntSoccer task, an agent can be empowered without interacting with the ball. The potential empowerment policy (white) navigates towards the central area of the arena, while the skills branch out from this point. (c) In the second version of the AntSoccer task, an agent can only be empowered if it controls the ball. The trained potential empowerment policy reaches the soccer ball, while skills kick the soccer ball in different directions. (d) In Craftax-Classic, the potential empowerment agent gathers materials, builds a crafting table, then builds tools using gathered materials.

Ablations. We ablate key design decisions in SEE. While our method builds on MISL, there are fundamental differences between empowerment maximization and skill learning. Skill learning trains effective empowerment policies *usually from a fixed or slightly noised starting state*. Potential empowerment maximization is *not* as simple as running MISL out-of-the-box and deals with the tougher problem of learning skills originating over many regions of state space. Design decisions such as skill-switching and the choice of the underlying MISL parameterization are critical to learn behaviorally interesting policies. We do the following ablations:

- **skill-switching** for learning the potential empowerment policy,
- **joint versus conditional MI**, and
- the **bilinear difference-in-representation encoder versus a more general encoder**.

Metrics. While MI is an important signal for training, achieving high MI does not guarantee meaningful empowerment maximization. In a continuous setting without constraints or significant noise, prior work has shown that trivial behaviors can realize potentially unbounded MI (Ji et al., 2026). Thus, we additionally track environment-specific metrics to judge the quality of the learned policies. In Ant, we measure the distance of the ant to the center of the arena. In AntSoccer, we measure the distance of the ant to the soccer ball. In BuilderBench, we measure the distance of the gripper to the cubes. In Crafter, we measure the achievement count. These external metrics measure the quality of learned behaviors *independent* of the MI quantities themselves.

5.1 Scaling and Baselines

Discrete Setting: Craftax-Classic. SEE scales to high-dimensional, discrete settings. In Craftax-Classic (Fig. 1d), the agent consistently unlocks 7 – 8 out of 22 total achievements per rollout from 1×10^9

environment steps. These metrics exceed the best previously-reported success rates of intrinsic exploration methods in Crafter, which obtains ~ 5.5 achievements with the same training (Mohamed et al., 2026). The estimated empowerment also rises steadily (Fig. 2), tracking the development of interesting behavior. Implementation details and hyperparameters for Craftax-Classic are in Section B.2.

Continuous Settings: Ant & AntSoccer. SEE learns meaningful empowerment behavior in continuous settings that tracks increasing average empowerment estimates (Fig. 8). If the empowerment is defined over all degrees-of-freedom (“Ant”), the policy π_{pot} learns to navigate to the *center region of the arena and stay there* (white, Fig. 1b) while the policies π_{eff} branch and cover the space (colors, Fig. 1b), following intuition that empowerment is high when the Ant can *do more* away from the walls of the arena. The baseline potential empowerment policies, in contrast, do not navigate to the center region and barely move at all. When the empowerment is defined only over the ball xy coordinates (“AntSoccer”), the policy π_{pot} navigates close to the ball (white, Fig. 1c) while the skill policies π_{eff} kick and dribble the ball in different directions (colors, Fig. 1c). The baseline potential empowerment policies, in contrast, do not consistently navigate close to the ball. When LatentGCE does happen to navigate close to the ball and kick the ball, the agents erroneously kick the ball to the very corner of the arena, where empowerment should be low.

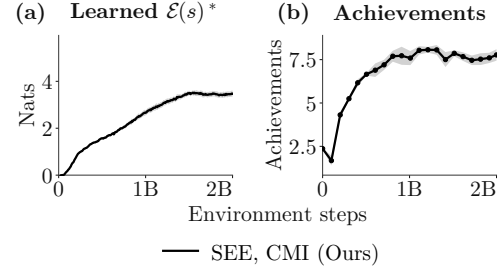


Figure 2: **The potential empowerment estimate increases with control in Crafter.** (a) The learned empowerment estimate increases consistently during training. 5 seeds, p.m. standard error after per-seed EMA ($\alpha = 0.05$). (b) Simultaneously, the agent intrinsically learns to complete achievements in the environment *without access to external rewards*. 5 seeds, evaluated over 128 random environments, p.m. standard error.

Continuous Settings: BuilderBench. Finally, in BuilderBench, the potential empowerment policy sneaks behind the blocks, where it has easy access to the blocks, and stays there (Fig. 3). The effective empowerment policies then knock the blocks in different directions (Fig. 1a), and the arm learns to get closer to the blocks. In contrast, the baseline methods perturb the blocks, but fail to remain close to the blocks at the end of the potential empowerment rollout.

In summary, SEE scales to manipulation, locomotion, and open-ended (e.g. Crafter) environments, whereas prior empowerment-maximization methods struggle.

5.2 Ablating Design Decisions

Policy switching. During data collection, SEE switches from the potential empowerment to the effective empowerment policy. An alternative is to separately train potential empowerment and effective empowerment policies. Despite learning meaningful skills from the starting state, this ablation leads to inconsistent potential empowerment policies that do not reach the center of the arena in Ant nor consistently reach the blocks in BuilderBench. Meanwhile, our base SEE stably learns potential *and* effective empowerment policies in all of these environments.

Conditional MI vs. Joint MI Variants. Klyubin et al. (2005a) define empowerment as a conditional MI conditioned on a starting context state. However, modern MISL methods learn variational lower bounds on joint MIs (e.g. $I(Z; S, S')$) that do *not* condition on the starting context. We compare using conditional versus joint MI parameterizations of empowerment. Further details of how we implement the CMI version of SEE is in the Appendix A.2. Policies trained with the true conditional MI learn empowering behaviors in AntSoccer and BuilderBench. However, when empowerment is defined over many continuous degrees of freedom like in Ant, the CMI falls into the trap where the decoder and policies overfit to small perturbations, leading to high MI values without meaningful behavior. In Crafter, using the CMI leads to the best performance among the ablations and prevents the MI collapse observed when optimizing the joint MI.

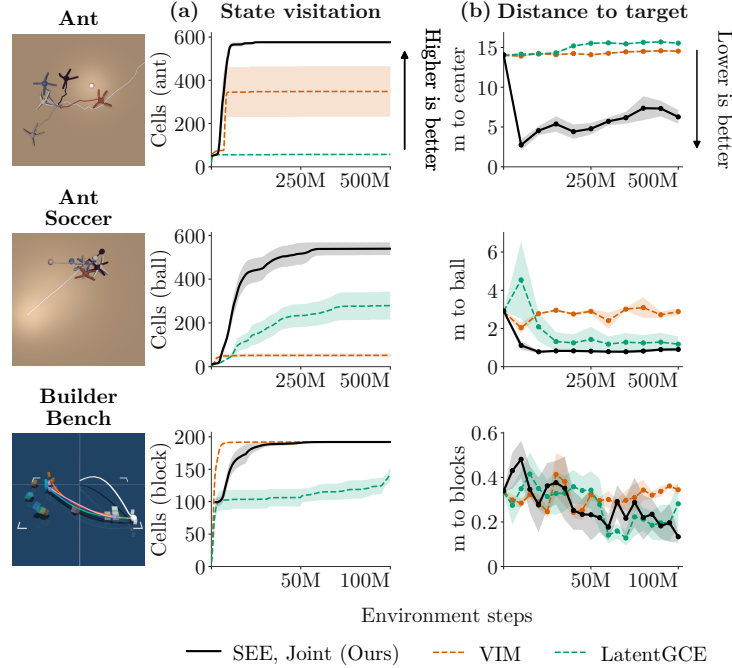


Figure 3: **Potential empowerment and control increases over training (continuous settings).** We compare SEE (Black) and baselines VIM (Orange) and LatentGCE (Green). (a) SEE visits diverse states across all environments, which is necessary to learn correct models and decoders of dynamics and skills. (b) SEE consistently learns potential empowerment policies that navigate towards central regions of the arena in Ant, to the ball in AntSoccer, and to the blocks in BuilderBench. VIM is unstable and largely stationary in Ant and AntSoccer. LatentGCE is stationary in Ant. Five seeds for SEE and LatentGCE and three for VIM, p.m. standard error. Eval metrics averaged over 128 randomized environments.

Bilinear Difference in Representations vs. General Parameterization. Recent MISL works such as Park et al. (2026, 2024); Zheng et al. (2025) utilize differences in representations $z^\top(\phi(s') - \phi(s))$ to parameterize classifiers (or similar objects) for skill learning. Replacing this parameterization with a more expressive logit significantly worsens the stability of training and the learned behaviors across environments, as the classifier can closely fit collected data and the policies “overfit” to sensitive classifiers.

5.3 Emergent Behaviors

Accruing and spending power. Across all environments, the learned switching policies accrue empowerment while executing the potential empowerment policy (Fig. 4, white traces). The policy gathers empowerment by walking to the center of the arena in Ant, getting close to the soccer ball in AntSoccer, getting close to the blocks in BuilderBench, and gathering resources and tools in Crafter. Then, the policies spend the empowerment when executing skills (Fig. 4, colorful traces), spreading out in the arena (Ant), kicking the ball (AntSoccer), and moving the blocks (BuilderBench).

Gathering resources and tool use. Resource-gathering and tool use enable empowerment-maximizing agents to exert control over the environment. In the Crafter environment, the trained potential empowerment agent learns to acquire many diverse materials (e.g. wood, saplings), such that its potential empowerment increases over the course of the episode. From that potentially-empowered state, the agent executes diverse skills such as building houses, mining, and using the gathered resources to craft new tools (Fig. 5).

Modifying the environment. Empowered agents learn to perturb their environment to maximize future control. This phenomena has been documented in prior works (Salge et al., 2014a) in simple tabular settings. In the scaled up Crafter setting, we observe more complex “building” behaviors, such as learning to construct a house when defending against enemies (Fig. 5).

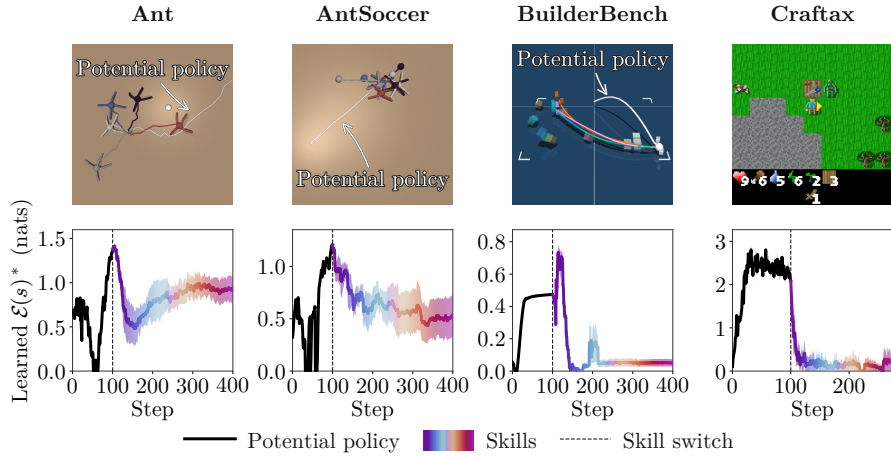


Figure 4: **Potential empowerment policies accrue empowerment, while skills spend it.** (Top row) Over the course of the trajectory, the potential empowerment policies (white lines) “gather” empowerment, reaching empowered states. Correspondingly, the empowerment estimates increase over the course of the trajectory. (Bottom row) At the gray skill switch mark, executing a skill (colorful lines) from the reached state “spends” the empowerment, reflecting the intuition that potential empowerment policies collect power and effective empowerment spends power. Traces use one seed and four skill branches per environment.

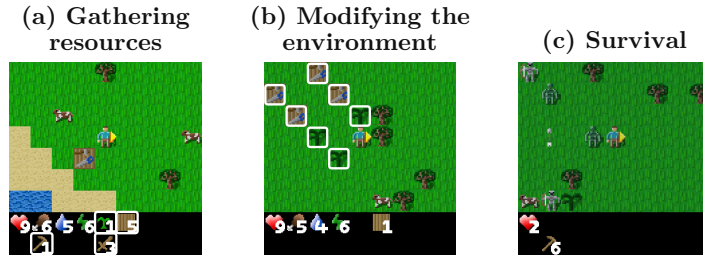


Figure 5: **Empowerment maximization has interesting emergent behaviors.** (a) In Crafter, the empowered agent quickly accumulates many resources and uses a tool (crafting table, wooden icon with saw) to craft other tools in the game. Resources boxed in white. (b) Early in training, the agent learns to build a “house” consisting of plants and crafting tables. Built tiles boxed in white. (c) The agent also naturally learns to survive, avoiding and killing enemies like zombies and skeletons. One seed per example; further qualitative examples across seeds in Section E.

Survival. In death, an agent no longer has any control and, thus, has no empowerment. Therefore, agents maximizing empowerment should learn survival skills. In Crafter, we see that empowerment-maximizing agents naturally learn survival skills like killing and dodging enemies. Importantly, the empowerment-maximizing agent learns behavior *distinct* from simply training a survival reward (e.g. $r = 1$ for every step, $r = 0$ upon termination), which does not learn more complex crafting behaviors and *only* prioritizes survival skills.

6 Discussion

Despite the conceptual appeal of empowerment as a tool for understanding natural agents and building artificial agents, its seeming computational intractability has caused prior work to resort to approximations that do not learn one policy that “keeps its options open” (Klyubin et al., 2008). In providing a conceptual link between skill learning methods and empowerment, we provide a method that maximizes potential and effective empowerment, two policies that measure controllability in distinct ways. We present key practical considerations to learn empowerment maximization policies in high-DoF settings. Future work should investigate simplified parameterizations of empowerment maximization that can resolve practical considerations detailed within our method, at scale. Furthermore, our work highlights

the sensitivity of empowerment maximization to representations of the world, with open questions on the interplay between representations of the world and empowerment maximization (Dai et al., 2026).

Acknowledgments. We would like to thank David Abel, Elom Amematsro, Mahsa Bastankhah, Caleb Chuck, Max Kleiman-Weiner, Andrew Levy, and anonymous reviewers for their insight and helpful feedback. This material is based upon work supported by Google and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2444107 and Award No. 2441665. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

AI use statement

We used generative AI tools to assist with implementing experiment code, preparing and revising figures, minor writing edits, assembling appendix details, and manuscript formatting. We did not use AI for the methods development, theoretical results, or any major writing. The authors take responsibility for the final content of this work.

Reproducibility statement

We provide algorithm pseudocode and implementation details in the appendix. Code, videos, and additional figures are available at <https://empowerment-geometry.github.io/scaling-empowerment/>.

References

- David Abel, Michael Bowling, André Barreto, Will Dabney, Shi Dong, Steven Hansen, Anna Harutyunyan, Khimya Khetarpal, Clare Lyle, Razvan Pascanu, et al. Plasticity as the Mirror of Empowerment. *Neural Information Processing Systems*, 38:164340–164374, 2025.
- Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. *Advances in Neural Information Processing Systems*, volume 30, 2017.
- David Barber and Felix Agakov. The IM Algorithm: A Variational Approach to Information Maximization. *Neural Information Processing Systems*, volume 16, 2003.
- Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *Neural Information Processing Systems*, 29, 2016.
- Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by Random Network Distillation. *International Conference on Learning Representations*, 2019.
- Philippe Capdepuy. *Informational Principles of Perception-Action Loops and Collective Behaviours*. Ph.D. thesis, University of Hertfordshire, 2011.
- Fei Dai and Alison Gopnik. Understand Intrinsic Motivation in Causal Learning Through Exploration. *Annual Meeting of the Cognitive Science Society*, volume 47, 2025.
- Fei Dai, Hanqi Zhou, Alison Gopnik, and Charley Wu. What Is Worth Representing? Representational Empowerment for Continual Model Construction. arXiv:2609.02322, 2026.
- Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity Is All You Need: Learning Skills Without a Reward Function. *International Conference on Learning Representations*, 2019.
- Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The Information Geometry of Unsupervised Reinforcement Learning. *International Conference on Learning Representations*, 2022.
- Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Foerster, and Mario Martin. Simplifying Deep Temporal Difference Learning. *International Conference on Learning Representations*, 2025.

- Raj Ghugare, Roger Creus Castanyer, Catherine Ji, Kathryn Wantlin, Jin Schofield, Karthik Narasimhan, and Benjamin Eysenbach. BuilderBench: The Building Blocks of Intelligent Agents. arXiv:2510.06288, 2026.
- Alison Gopnik. Empowerment as Causal Learning, Causal Learning as Empowerment: A Bridge Between Bayesian Causal Hypothesis Testing and Reinforcement Learning. PhilSci-Archive, 2024.
- Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational Intrinsic Control. *International Conference on Learning Representations, Workshop Track*, 2017.
- Lucas Gruaz, Alireza Modirshanechi, Sophia Becker, and Johanni Brea. Merits of Curiosity: A Simulation Study. *Open Mind*, 9:1037–1065, 2025.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *International Conference on Machine Learning*, pp. 1861–1870, 2018.
- Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. VIME: Variational Information Maximizing Exploration. *Neural Information Processing Systems*, 29, 2016.
- Khurram Javed and Richard S Sutton. The Big World Hypothesis and Its Ramifications for Artificial Intelligence. *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.
- Catherine Ji, Vivek Myers, Sergey Levine, and Benjamin Eysenbach. The Geometry of Empowerment. arXiv:2610.07796, 2026.
- Maximilian Karl, Maximilian Soelch, Philip Becker-Ehmck, Djalel Benbouzid, Patrick van der Smagt, and Justin Bayer. Unsupervised Real-Time Control through Variational Empowerment. arXiv:1710.05101, 2017.
- Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. All Else Being Equal Be Empowered. *European Conference on Artificial Life*, pp. 744–753, 2005a.
- Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Keep Your Options Open: An Information-Based Driving Principle for Sensorimotor Systems. *PloS One*, 3(12):e4018, 2008.
- A.S. Klyubin, D. Polani, and C.L. Nehaniv. Empowerment: A Universal Agent-Centric Measure of Control. *IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135 Vol.1, 2005b.
- Andrew Levy, Alessandro G Allievi, and George Konidaris. Representation Learning and Skill Discovery With Empowerment. *Reinforcement Learning Conference*, 2025.
- Aly Lidayan, Yuqing Du, Eliza Kosoy, Maria Rufova, Pieter Abbeel, and Alison Gopnik. Intrinsically-Motivated Humans and Agents in Open-World Exploration. arXiv:2503.23631, 2025.
- Zhuang Ma and Michael Collins. Noise Contrastive Estimation and Negative Sampling for Conditional Models: Consistency and Statistical Efficiency. *Empirical Methods in Natural Language Processing*, pp. 3698–3707, 2018.
- Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A Lightning-Fast Benchmark for Open-Ended Reinforcement Learning. arXiv:2402.16801, 2024.
- Faisal Mohamed, Catherine Ji, Benjamin Eysenbach, and Glen Berseth. Temporal Representations for Exploration: Learning Complex Exploratory Behavior Without Extrinsic Rewards. *International Conference on Learning Representations*, 2026.
- Shakir Mohamed and Danilo Jimenez Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. *Neural Information Processing Systems*, 28, 2015.
- Jongchan Park, Seungjun Oh, Seungho Baek, and Yusung Kim. Learning Generalizable Skill Policy With Data-Efficient Unsupervised RL. arXiv:2607.00392, 2026.
- Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking Offline Goal-Conditioned RL. *International Conference on Learning Representations*, 2025.
- Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable Unsupervised RL With Metric-Aware Abstraction. *International Conference on Learning Representations*, 2024.

- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-Driven Exploration by Self-Supervised Prediction. *International Conference on Machine Learning*, pp. 2778–2787, 2017.
- Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.
- Christoph Salge, Cornelius Glackin, and Daniel Polani. Changing the Environment Based on Empowerment as Intrinsic Motivation. *Entropy*, 16(5):2789–2819, 2014a.
- Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment—an Introduction. *Guided Self-Organization: Inception*, pp. 67–114. Springer, 2014b.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.
- Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-Aware Unsupervised Discovery of Skills. *International Conference on Learning Representations*, 2020.
- Jinyeop Song, Jeff Gore, and Max Kleiman-Weiner. Estimating the Empowerment of Language Model Agents. arXiv:2509.22504, 2026.
- Stas Tiomkin, Christoph Salge, and Daniel Polani. Process Empowerment for Robust Intrinsic Motivation. *Journal of Physics: Complexity*, 6(3):035011, 2025.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning With Contrastive Predictive Coding. arXiv:1807.03748, 2018.
- Claire Yang, Claire Jie Zhang, Maya Cakmak, and Max Kleiman-Weiner. When Assisting One Disempowers Another. arXiv:2511.04177, 2026.
- Eunice Yiu, Kelsey Allen, Shiry Ginosar, and Alison Gopnik. Empowerment Gain and Causal Model Construction: Children and Adults Are Sensitive to Controllability and Variability in Their Causal Interventions. arXiv:2512.08230, 2026.
- Ruihan Zhao, Kevin Lu, Pieter Abbeel, and Stas Tiomkin. Efficient Empowerment Estimation for Unsupervised Stabilization. arXiv:2007.07356, 2021.
- Chongyi Zheng, Jens Tuyts, Joanne Peng, and Benjamin Eysenbach. Can a MISL Fly? Analysis and Ingredients for Mutual Information Skill Learning. *ICLR 2025*, 2025.

A Algorithm and Implementation Details

Below, we include key algorithm and implementation details. While we describe the Joint MI classifier form of the SEE for simplicity, we detail the Conditional MI variant of SEE later in Section A.2. When the skill is independent of the starting state (e.g. drawn from the same prior distribution at a starting state), the joint $(I(Z; S_+, S_0))$ and conditional MI $(I(Z; S_+ | S_0))$ are equivalent, as $p(z | s_0) = p(z)$. Practically, our Joint MI classifier breaks this independence assumption by sampling a state *within a trajectory* as a starting point, then sampling a geometric future state from that point. Further discussion is in Section A.2.

A.1 Training the classifier

SEE utilizes MISL methods to lower bound the pointwise potential empowerment. We take an approach similar to CSF (Zheng et al., 2025), employing noise contrastive estimation to learn the skill classifier, which provides a variational lower-bound on the MI.

Without loss of generality, assume that the skills z are drawn from a uniform distribution on a sphere $p(z) = \text{Unif}(\mathbb{S}^{d-1})$. Let $f_\phi(s, s_+, z)$ denote a parameterized logit over a state s , future state s_+ , and skill z , constrained to take the form

$$f_\phi(s, s_+, z) = \frac{1}{\tau} z^\top (\phi(s_+) - \phi(s))$$

which we refer to as a bilinear difference-of-representations parameterization, with temperature $\tau = 0.1$. This parameterization of the logit appears in recent SOTA MISL literature and has been shown to train persistent skill behavior that spread over latent space; we refer readers to these references for more information (Park et al., 2026, 2024; Zheng et al., 2025).

Then, given access to a dataset $\mathcal{D}$ of starting states, skills, and outcomes (s, s_+, z) , where outcomes $s_+ \sim p^\pi(\cdot | s, z)$ are drawn from the DSOM, InfoNCE trains a classifier to distinguish whether skills are drawn from the joint distribution $p(z, s, s_+)$ or the product of marginals $p(z)p(s, s_+)$. For a batch of size B with one positive pair $(z, (s, s_+))$, where $z \sim p^\pi(\cdot | s, s_+)$, and $B - 1$ negative pairs $(z, (s, s_+))$, where $z \sim p(z)$, we have:

$$\text{InfoNCE Loss: } \mathcal{L}(\phi) = \frac{1}{B} \sum_{i=1}^B \left[\log \sum_{j=1}^B \exp f_\phi(s_i, s_{+,i}, z_j) - f_\phi(s_i, s_{+,i}, z_i) \right].$$

At optimum, the logit recovers the true classifier

$$f_\phi^*(s, s_+, z) = \log \frac{p(z | s, s_+)}{p(z)} + c_\phi(s, s_+)$$

where $c_\phi(s, s_+)$ denotes the log partition function

$$c_\phi(s, s_+) \triangleq \log \mathbb{E}_{p(z)} [\exp(f_\phi(s, s_+, z))].$$

The learned logit gives a variational lower bound on the MI $I(Z; S, S_+)$:

$$\mathbb{E}_{p(z, s, s_+)} [f_\phi(s, s_+, z) - c_\phi(s, s_+)] \leq I(Z; S, S_+),$$

which becomes tight at the optimum (Ma and Collins, 2018).

Adding an Information Bottleneck. High MI *does not directly imply interesting behavior in continuous settings*. Prior works note that without any type of constraint on the information maximization problem, empowerment is ill-defined in continuous settings and can be infinite (Salge et al., 2014b). In our case, a

fully-general skill decoder trained over the transitions can give arbitrarily large empowerment estimates, even for skills that barely change the state.

For the inner skill optimization problem, it is critical to apply an information bottleneck from $S_0, S_+ \rightarrow Z$ when fitting the encoder in the continuous setting. For simplicity, we uniformly add Gaussian noise to the state when training the classifier. This noising forces the classifier to identify skills from perturbations in the state beyond this noise scale. Concretely, we train $f_\phi(s + \epsilon, s_+ + \epsilon_+, z)$, redrawing the noise $\epsilon, \epsilon_+ \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2 I)$ at each encoder update. SEE does not noise states during policy and value optimization and solely applies the bottleneck during classifier training. Noise scales for each environment are given in Section B. Craftax-Classic results use no noise.

A.2 Data Collection & Conditional MI

During data collection, SEE switches from the potential empowerment at a switch state s_0 to the effective empowerment policy. The purpose of this switch is to ensure that the method learns skills precisely at reachable locations. During rollout, we take the skill distribution $p(z | s_0) = p(z)$ to be uniform. Then, classifying skill z from the switch state s_0 and the reached future states gives, in expectation, a variational lower bound on the empowerment (Section F). In practice, however, training an encoder over this data can lead to the classifier severe overfitting on the context s_0 , particularly in continuous settings.

We can address this overfitting in two ways. The first is to instead learn a joint MI encoder, and fit an encoder to a sampled state in a trajectory (not necessarily the switch point) and a discounted future state, which creates a larger diversity of samples; we call this approach SEE with a Joint MI classifier. The second is to generate additional trajectories from diverse switch points across the collected trajectories, preserving the conditional MI definition of empowerment; we call this approach SEE with a Conditional MI (CMI) classifier. *The difference between the conditional MI versus the joint MI classifier variants of SEE happens at the level of data collection and encoder training.*

Why the CMI variant estimates conditional MI If skills are sampled uniformly-at-random from the initial state s_0 , the MI $I(Z; S_0)$ equals 0. Thus, a simple way to optimize the conditional MI during the inner MISL optimization is to roll out skills sampled from the uniform marginal $p(z)$ and train the classifier as usual. Concretely, we copy a switch state s_0 then roll out fresh skill(s) from this switch state (i.e. without resets). The classifier only trains on these “branching” rollouts. Then, the distributions $p(z | s_0)$ and $p(z)$ are equivalent, and the joint MI and conditional MI are equivalent:

$$I(Z; S_0, S_+) = I(Z; S_0) + I(Z; S_+ | S_0) = I(Z; S_+ | S_0).$$

An alternative approach is to train an estimator of $p(z | s)$; we did not pursue this approach to keep the method simple.

A.3 Training empowerment value functions and policies

A.3.1 Inner Optimization: Skill Learning/Effective Empowerment

We consider continuous skill spaces for all of our learned skills.

Markovian Skill Policies & Critics in Continuous Settings. The skill policies and value functions for the inner optimization problem condition on the start state, the current state, and a skill (Section 4.1). As a result, the skill policy is a function of (s, s_0, z) and the skill critic is a function of (s, s_0, a, z) . Meanwhile, the potential policy simply uses the pointwise estimate of the potential empowerment as a state-wise reward, so the potential policy and critics take as input s and (s, a) , respectively. Each SAC critic consists of two independent multilayer perceptrons, each with two 512-unit hidden layers, GELU activations, layer normalization, and a scalar Q-value output.

Nonmarkovian Skill Policies & Critics in Crafter. In the case of a nonmarkovian policy as used in the Crafter environment, which is partially-observed, the policies and PQN (Gallici et al., 2025) critics

take in an additional recurrent state. Following (Gallici et al., 2025), both of the potential empowerment and skill learning Q-networks use a 512-unit fully connected layer with layer normalization and ReLU, followed by a 512-unit LSTM and a linear output giving one Q-value per action. The LSTM receives the full symbolic observations, the previous action, and the encoded inputs. The hidden and cell states carry information from earlier steps.

In addition the recurrent state, the skill Q-network also receives the observation at the skill switch/initiation of the skill and the skill latent z . The recurrent state resets at the start of each episode and at the skill switch, so each skill starts from a fresh memory. An alternative option is to carry through the recurrent state through the skill switch, where empowerment is now defined at the recurrent state rather than the observation. Carrying the recurrent state through the skill switch did not substantially change the results.

Skill Rewards. As mentioned in Section 4.1, setting the reward to the normalized classifier score provides a lower bound on the MI in expectation, which we show later in Theorem 3. In practice, for the inner optimization problem, MISL methods like CSF (Zheng et al., 2025) utilize the *unnormalized* classifier as the reward, because the latter partition term has a skill-agnostic, anti-exploration effect. In our case, while we desire the exploration effects of the unnormalized reward, for the goal of potential empowerment maximization, we need the inner optimization to actually lower bound the pointwise potential empowerment. Thus, at the start of training, we begin with the unnormalized reward to get exploration ($\lambda = 0$), then gradually turn on the anti-exploration term ($\lambda = 1$) so the method estimates pointwise empowerment by the end of training: The effective-policy reward is

$$r_{\text{eff}}(s, a, s'; s_0, z) = f_{\phi}(s_0, s', z) - \lambda \log \frac{1}{K} \left(e^{f_{\phi}(s_0, s', z)} + \sum_{j=1}^{K-1} e^{f_{\phi}(s_0, s', z_j)} \right), \quad z_j \stackrel{\text{iid}}{\sim} \text{Unif}(\mathbb{S}^{d-1}),$$

with $K = 1024$, where the negative skills z_j are fresh draws from the uniform skill prior $p(z)$. Specifically, SEE sets $\lambda = 0$ for the first 30% of training, increases it linearly to 1 over the next 50%, and keeps it at 1 for the final 20%. In experiments, we see that this annealing does not significantly change training in continuous environments. In Craftax-Classic, annealing gives the largest empowerment estimate (Fig. 12). Meanwhile, using the normalized reward throughout training ($\lambda = 1$) learns far fewer achievements. Using the unnormalized reward $\lambda = 0$ leads to the highest achievement count (> 9 achievements), but the value functions are no longer estimates of a lower bound on empowerment by the end of training and are unnormalized objects.

Increasing sample efficiency with HER. The effective empowerment value functions and policies are heavily-parameterized and depend on the current state, start-state, and the skill. To improve sample efficiency and collect more training examples from the same set of the data, the base SEE employs Hindsight Experience Relabeling (HER) for the start-state and skill during value function and policy training: while, during rollout, there is a true start-state, SEE relabels the start-state by sampling geometrically backwards-in-time, then greedily relabels the executed skill. For example, say that the agent sampled a skill from $p(z)$ at start-state s_0 , and the entire trajectory enters a replay buffer. During policy and value learning using a particular transition $(s_0, s, a, s'z)$, SEE resamples an effective s'_0 backwards-in-time then sets $z \rightarrow z'$, where $z' = \arg \max_{z \in \mathcal{Z}} G(s'_0, s, a, z)$ and $G(s'_0, s, a, z)$ is a Monte-Carlo estimate of the cost-to-go. While we acknowledge and observe that this relabeling introduces bias and leads to some skill collapse, this relabeling helps scale the method.

A.3.2 Outer Optimization: Potential Policy and Critic Optimization

In the ideal pointwise setting, the potential empowerment critic simply takes the Monte-Carlo estimate of the expected skill critic over 8 uniformly-sampled skills as the reward:

$$r_{\text{pot}} = \frac{1}{8} \sum_{k=1}^8 Q(s, a_k; s, z_k), \quad z_k \sim p, \quad a_k \sim \pi_{\text{eff}}(\cdot \mid s, z_k; s).$$

Under the stated assumptions, in expectation, this estimate is a lower bound on the pointwise potential empowerment (see Eq. (9)).

Separating Actor and Critic Buffers from the Classifier Buffer. All the critics used in this method can theoretically work in off-policy settings. However, the skill classifier used in the inner optimization cannot train on an extensively off-policy replay buffer to improve the *current* policy’s effective empowerment. Levy et al. (2025) discuss the importance of training a classifier on the current policy. Thus, we maintain separate buffers for actor-critic training and classifier training, where the classifier buffer contains only recent experience and is generally much smaller than that of the actor-critic buffers. Exact parameters are in Section B.

A.4 Empowerment maximization pseudocode

Algorithm 1: SEE

Require: Skill prior $p(z)$, episode length T , switch time M , iterations N

- 1: Initialize potential policy π_{pot} , skill policy π_{eff} , their critics, classifier f_ϕ , and buffers.
 - 2: **for** $n = 1, \dots, N$ **do**
 - 3: **Collect:** Run π_{pot} for M steps from reset; record the switch state s_0 (Section 4.2).
 - 4: Draw $z \sim p(z)$ and run $\pi_{\text{eff}}(\cdot \mid s, z; s_0)$ for $T - M$ steps, holding z, s_0 fixed.
 - 5: Store the two-stage rollout for policy and critic updates.
 - 6: **Fit classifier:** Train f_ϕ with InfoNCE on states and discounted future states
 from skill execution (Joint MI), or branches with fresh skills $z \sim p(z)$
 from copied starting states (CMI; Section A.2).
 - 7: **HER (SAC only):** Replace the anchor and skill of replayed skill transitions in hindsight (Section A.3.1).
 - 8: **Inner optimization:** Update π_{eff} and its critics using r_{eff} with the current normalization weight $\lambda(n)$ (Sections A.3 and 4.1).
 - 9: **Estimate empowerment:** Average skill values over eight draws $z \sim p(z)$ (Eq. (9)).
 - 10: **Outer optimization:** Use this estimate as r_{pot} to update π_{pot} and its critic (Section 4).
 SAC uses replay; PQN uses the two-stage rollout; CMI branches train only f_ϕ .
 - 11: **return** $\pi_{\text{eff}}, \pi_{\text{pot}}$
-

B Experimental setup and evaluation

Continuous Env Evaluation. Like in the data collection during training, the evaluation rollouts run the potential empowerment policies for the first 100 steps. Then, the rollout draws a skill $z \sim p(\cdot)$ and executes the effective empowerment policy until termination or truncation for the next 300 steps. Evaluations average over 128 episodes in 128 new, sampled environments.

The potential empowerment target distance is the xy distance *at the switch point*, which is the 100-step potential empowerment phase of the rollout. In Ant, this is the distance from the Ant to the center of the arena; In AntSoccer, this is the distance away from the soccer ball; In BuilderBench, this is the distance to any of the blocks. Meanwhile, the skill displacements in Fig. 9 measure the displacement from the switch point to the end of the skill rollout (next 300 steps). In Ant, this is the displacement of the ant; In AntSoccer, this is the displacement of the ball; In BuilderBench, this is the displacement of the blocks.

Craftax Classic Evaluation. We use the externally-defined achievement count for evaluation in Craftax-Classic, measuring the total achievements accrued over the course of the two-phase rollout in 128 randomized evaluation environments.

Curves show the mean $\pm$ standard error across training seeds: five for SEE and LatentGCE, and three for VIM and each ablation. The learned empowerment metrics are plotted after per-seed EMA ($\alpha = 0.05$) before averaging. The achievement metrics are unsmoothed.

B.1 Continuous Experiment Details

In the figures, Ant denotes the full-state classifier (42 features), while AntSoccer denotes the classifier using only ball x, y ; BuilderBench uses the three blocks' positions. Visitation counts the cumulative number of distinct planar cells seen during training. In Ant and AntSoccer, the cells are $1\text{ m} \times 1\text{ m}$ xy tiles of either the ant (Ant) or ball (AntSoccer) location, with a total of 576 cells. Note that SEE can quickly get complete or close-to-complete coverage in both Ant and AntSoccer. In BuilderBench, the cells are $5\text{ cm} \times 5\text{ cm}$ xy tiles pooled across the three blocks, totaling 192 cells. For SEE and ablations, we record and plot the visitation metrics for the skill phase of the rollouts. Because the baseline LatentGCE and VIM methods do not have a skill phase, we instead plot the visitation metrics over the entirety of the training rollouts.

B.1.1 Continuous hyperparameters

Hyperparameters are in Table 1. Training curves are in Fig. 3 and learned MI curves are in Fig. 8.

Both Ant and AntSoccer use the same arena, but use different encoder observations when training the classifier. In Ant, the ant starts in a top-right start cell with a uniform, random offset in $[-1, 1]^2\text{ m}$. The ball sits 1.5 m past the center. In AntSoccer, the ant starts 80% of the way along a line from the ant start to the center of the arena, where the ball starts at the center. BuilderBench sets the three cubes in a deterministic position in the corner. The gripper initialization is also deterministic at the center-edge of the play area, far from the blocks.

Table 1: Hyperparameters for SEE with joint MI in continuous environments.

Parameter	Ant	AntSoccer	BuilderBench
Encoder observations	Full state (42)	Ball x, y (2)	Block positions (9)
Observation noise std	0.05	0.05	0.01
Skill dimension	2	2	9
Online env steps per seed	500M	500M	100M
Replay capacity (transitions)	10M	10M	6M
Batch size	1,024	1,024	256
Target entropy	0	0	-12.5
λ annealing (env steps)	150M-400M	150M-400M	30M-80M
Shared settings			
Learner	SAC		
Training seeds	5		
Skill prior	Uniform unit sphere		
Potential / skill phase	100 / 300 steps		
Reward partition weight λ	0 before the interval, linear increase to 1 from 30 – 80% of training, 1 thereafter		
Encoder buffer	3,334 episodes		
Learning rate	3×10^{-4}		
SAC / encoder samples per collected transition	4 / 1		
Critic target update rate	0.02		
Policy / future-sampling discount	0.99 / 0.99		
Contrastive temperature	0.1		
Entropy temperature	Learned, initialized to 1		
Actor / critic / encoder hidden layers	[512, 512]		
Layer norm	Critic and encoder only		
Potential-value skill samples	8		

B.2 Discrete Experiment Details

B.2.1 Craftax-Classic

Hyperparameters are in Table 2. Fig. 6 shows one two-phase episode. We use PQN (Gallici et al., 2025) rather than SAC to utilize the GPU acceleration in the Craftax environment.

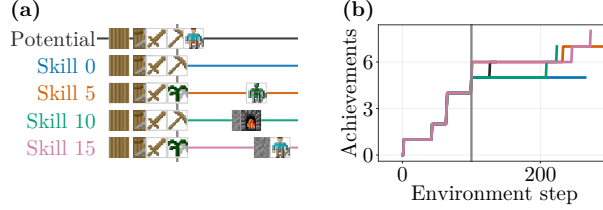


Figure 6: **Achievements in a single Craftax-Classic two-phase episode.** Icons indicate when achievements first occur. (a) The black line shows the achievements of the shared potential phase up to the skill switch (vertical gray line). In this rollout, these achievements include collecting wood, building a crafting table, then crafting a wooden sword. The colorful skill rows show the achievements rolled out from the switch point sharing the same starting state, including sleeping/waking up, building a pickaxe, planting a tree, collecting stone, building a furnace, and defeating an enemy. (b) The lines indicate the cumulative achievement counts over the rollout, achieved without the use of any extrinsic reward. One seed.

Table 2: Hyperparameters in Craftax-Classic (SEE, CMI (Ours); quantitative curves).

Parameters	Value
Encoder observations	inventory (12 features)
Encoder data	CMI branches from copied states
Observation noise std (classifier bottleneck)	0
Future-sampling discount	0.95
Logit temperature	0.1 (fixed)
Encoder learning rate	1e-3
Encoder batch size	4,096
Encoder hidden dims	[512, 512]
Encoder Buffer	10,000,000 branch transitions
Skill dimension	16
Skill prior	uniform unit sphere
Potential / skill phase	100 / 900 steps
Training seeds	5
Encoder branch horizon	900
Extra simulator steps per live step	0.9 from encoder branches, excluded from the live-step budget
Online env steps per seed (live)	2,000,000,000
Potential-value skill samples	8
Skill / anchor relabeling	Disabled
Reward partition weight λ	0 before 600M live steps, linear increase to 1 at 1.6B, 1 thereafter
Learning rate	3e-4 (linear decay to 0)
Minibatches per rollout	4, each with 256 sequences of 128 steps
Update epochs	4
Encoder updates per rollout batch	4
Exploration	ϵ -greedy: $1 \rightarrow 0.005$ over 100M live env steps
Discount γ	0.95
PQN λ	0.5
Max gradient norm	0.5
Q-network input normalization	Batch renormalization

C Baselines

Unlike our method, both VIM (Mohamed and Jimenez Rezende, 2015) and LatentGCE (Zhao et al., 2021) consider open-loop definitions of empowerment, where A_0^{H-1} is shorthand for the action sequence $(a_0, \dots, a_{H-1})$:

$$\mathcal{E}_{\text{VIM}}(s_0) \triangleq \max_{\pi \in \Pi_{\mathcal{Z}}} I^\pi(A_0^{H-1}; S_H \mid S_0 = s_0). \quad (10)$$

VIM. VIM utilizes a variational lower bound on the empowerment with a learned decoder $q_\theta(a_1^H \mid s_0, s_H)$, a generative model for the action sequence. VIM fits this decoder using a standard cross-entropy loss. Then, by definition, learning a source distribution $\omega(a_0^{H-1} \mid s_0)$ that maximizes this variational lower bound gives a lower bound on the true open-loop potential empowerment. We follow the autoregressive source and decoder in Mohamed and Jimenez Rezende (2015, Sec. 4.2), using tanh-Gaussians for our continuous actions.

Fixing the estimated decoder $q_\theta(a_1^H \mid s_0, s_H)$, at optimum, the source distribution $\omega(a_0^{H-1} \mid s_0)$ takes the form

$$\omega^*(a_0^{H-1} \mid s_0) \propto \exp(\mathbb{E}_{p(s_H \mid s_0, a_0^{H-1})} [\log q_\theta(a_0^{H-1} \mid s_0, s_H)]).$$

However, computing the partition function/normalizer is generally intractable. Instead, VIM regresses a function $f_\phi(s_0, a_0^{H-1}) \triangleq \log h_\phi(a_0^{H-1} \mid s) + \psi_\phi(s_0)$ onto the energy $\mathbb{E}_{p(s_H \mid s_0, a_0^{H-1})} [\log q_\theta(a_0^{H-1} \mid s_0, s_H)]$, where h_ϕ is normalized. At optimum, the function $\psi_\phi(s_0)$ converges to an approximation of the open-loop potential empowerment by the standard variational Gibbs identity. We refer the reader to Mohamed and Jimenez Rezende (2015) for more details.

LatentGCE. The next baseline we consider is LatentGCE (Zhao et al., 2021). Unlike VIM, LatentGCE learns a generative, forward-in-time model to estimate empowerment. LatentGCE learns a latent model of the environment that is a linear Gaussian channel, which automatically gives a closed-form expression for the open-loop potential empowerment (under power constraints). This approach sidesteps the challenges of optimizing a source distribution by directly learning a generative model with a linear Gaussian form.

C.1 Validating Empowerment Fields in Simple Continuous Benchmarks

We compare the learned empowerment fields in two simple, didactic continuous environments of a pendulum and ball-in-box discussed in LatentGCE. The learned empowerment fields are in Fig. 7. Our JAX implementations of LatentGCE (based on the original LatentGCE codebase) and VIM qualitatively reproduce the behavior reported in Zhao et al. (2021), where LatentGCE succeeds to find the high empowerment region while VIM does not.

C.2 Implementation in the main evaluation environments

We then apply VIM and LatentGCE to Ant, AntSoccer, and BuilderBench following the referenced manuscript methodology. Unfortunately, neither VIM nor LatentGCE provide full public codes of their methods. In particular, VIM has no publicised codebase and the public LatentGCE codebase does not implement the version of their method that learns latents. As such, we adapted the methodology discussed in the LatentGCE paper (originally applied to pixel-based tasks) in addition to their partial codebase to our scaled continuous settings (Zhao et al., 2021). We implemented VIM following the algorithm block included in the paper and include these implementations within our public codebase (Mohamed and Jimenez Rezende, 2015). Key hyperparameters are included in Tables 3 and 4.

Both methods estimate the empowerment over the same degrees of freedom as SEE: full state in Ant, ball in AntSoccer, and block positions in BuilderBench. In LatentGCE, we replace the convolutional encoders originally designed for the pixel-based settings with standard MLPs. Both methods optimize the potential empowerment policies using PPO. Since both methods estimate open-loop empowerment,

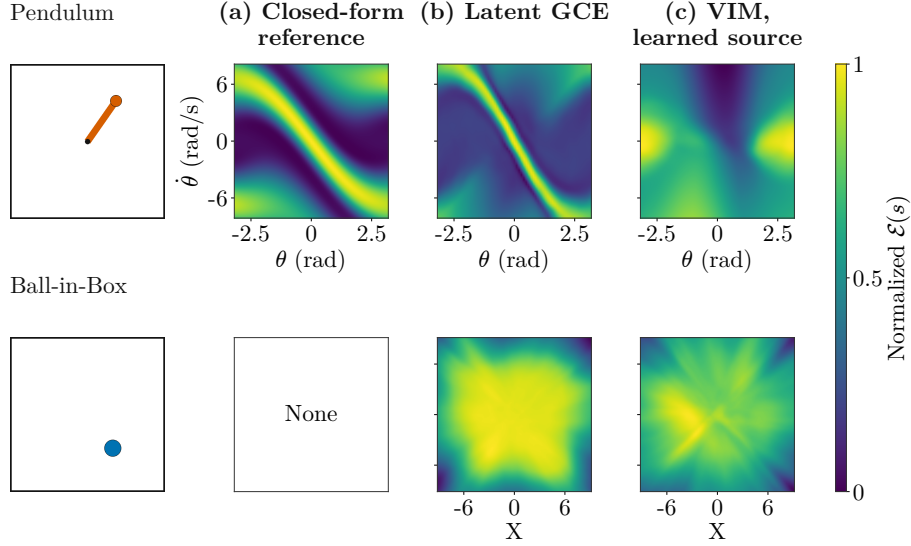


Figure 7: **Empowerment fields on low-dimensional benchmarks.** For the canonical environments of a pendulum and a ball in a box with hard walls, various empowerment maximization methods predict (normalized) empowerment fields over state space. (a) A linearized reference reveals that the potential empowerment is maximized when the pendulum points straight up. For the Ball in Box, the potential empowerment is maximal at the center of the box. Thus, a potential empowerment policy over a long effective horizon should seek a straight-up state in the pendulum and the central state for Ball in Box. (b) The empowerment field predicted by LatentGCE method correctly identifies these potential empowerment states. (c) VIM does not recover these high empowerment regions. One seed per learned field.

we consider empowerment over open-loop action sequences of length $H = 30$ for VIM and $H = 100$ for LatentGCE. We chose a shorter sequence length for VIM due to the observed instability of the algorithm. We chose $H = 100$ for LatentGCE to match the effective trajectory length scale for the rollouts elsewhere in the paper, corresponding to a discount factor $\gamma = 0.99$. Tables 3 and 4 list the key hyperparameters, with full implementation provided in the linked code. In general, we find that VIM exhibits unstable training. In VIM, defining an energy for the source action sequence distribution with the learned decoders led to unstable source updates, which then negatively impacted future data collection. In earlier attempts at running LatentGCE, exponentiating the empowerment estimate produced large rewards (Zhao et al., 2021). Thus, we instead set the reward as the empowerment, following the released LatentGCE pseudocode and the methodology in the rest of this paper.

D Ablation Details

Here, we include additional ablations and ablation details. The corresponding figures are in Fig. 9 and Fig. 10. Default SEE utilizes the joint MI classifier in the continuous settings and the CMI classifier in the Crafter setting.

D.1 Policy switching.

The first design decision we ablate is policy switching. An alternative is to separately train potential empowerment and effective empowerment policies:

- Ours:** Policy switching from potential to effective during data collection.
- Ablation:** Training skills and potential policy from the the same starting state, without skill switching. To allocate similar training time to the potential and skill phases as in the base SEE, we roll out using the potential policy 25% of the time, and the skill phase the other 75%.

Table 3: VIM hyperparameters for Ant, AntSoccer, and BuilderBench. Each environment uses 3 training seeds.

Parameter	Value
Training budget (Ant / AntSoccer / BuilderBench)	500M / 500M / 100M
Episode length / probe horizon H	400 / 30 steps
Parallel environments / rollout length	256 / 80 steps
Probe starts per update	256, one uniformly sampled rollout time per environment
PPO controller	
Actor / critic hidden layers	[256, 256], tanh
Optimizer / learning rate	Adam / 3×10^{-4} , linear decay
Discount γ / GAE λ	0.99 / 0.95
Update epochs / minibatches	4 / 4
PPO policy and value clipping / max. gradient norm	0.2 / 0.5
Entropy / value loss coefficients	0.01 / 0.5
Action distribution	Tanh-Gaussian, actions in $[-1, 1]^{d_a}$
Observation normalization	Running mean and variance
Empowerment estimator	
Estimator replay capacity / minibatch	10,000 tuples / 1,024
Estimator passes over replay per update	10
Learning rate	3×10^{-4}
Hidden layers	Source/decoder [512, 512, 512]
Additional hidden layers	Empowerment head [256, 256, 256]
Activation	ReLU
VIM energy coefficient	1
Controller reward	Estimated empowerment, clipped below at zero (due to instability)

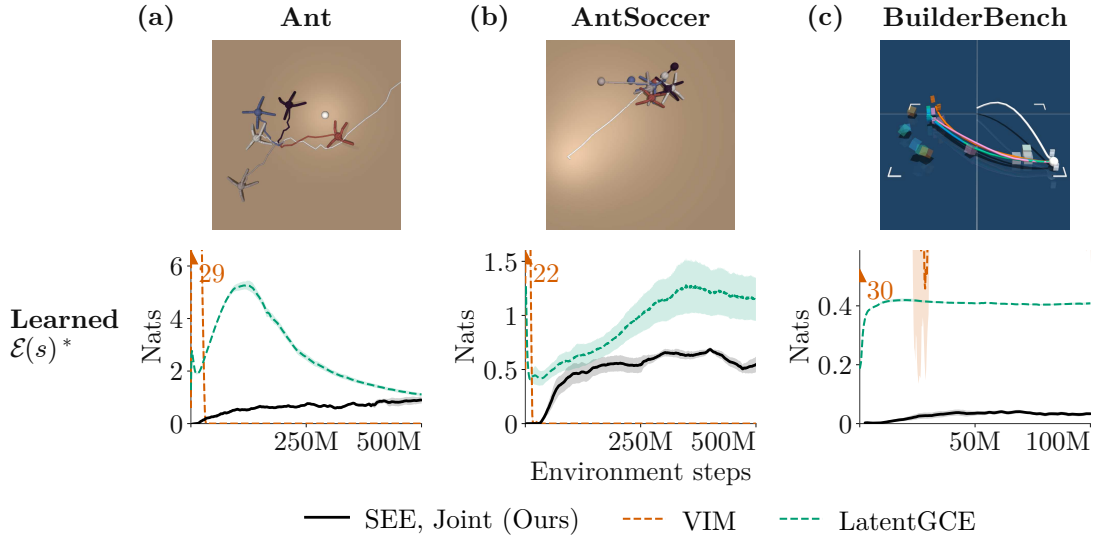


Figure 8: **Learned empowerment in continuous settings.** SEE (Black) empowerment values gradually increase over time. Meanwhile, VIM (Orange) empowerment values are unstable and are very large at the start of training (Orange arrows point beyond the displayed y-axis). The LatentGCE (Green) empowerment values do not monotonically increase in time. *As a note, learned estimates use different objectives and regularization, so their magnitudes are not directly comparable across methods. Five seeds for SEE and LatentGCE and three for VIM, p.m. standard error.

Table 4: LatentGCE hyperparameters for Ant, AntSoccer, and BuilderBench. Each environment uses 5 training seeds.

Parameter	Value
Training budget (Ant / AntSoccer / BuilderBench)	500M / 500M / 100M
Episode length / probe horizon H	400 / 100 steps
Parallel environments / rollout length	256 / 80 steps
Probe starts per update	256, one uniformly sampled rollout time per environment
PPO controller	
Actor / critic hidden layers	[256, 256], tanh
Optimizer / learning rate	Adam / 3×10^{-4} , linear decay
Discount γ / GAE λ	0.99 / 0.95
Update epochs / minibatches	4 / 4
PPO policy and value clipping / max. gradient norm	0.2 / 0.5
Entropy / value loss coefficients	0.01 / 0.5
Action distribution	Tanh-Gaussian, actions in $[-1, 1]^{d_a}$
Observation normalization	Running mean and variance
Empowerment estimator	
Estimator replay capacity / minibatch	10,000 tuples / 1,024
Estimator passes over replay per update	10
Learning rate	10^{-4}
Hidden layers	Encoders/channel [512, 512, 512]
Additional hidden layers	Reconstruction decoders [512, 512]
Activation	ReLU
State / action latent dimension	32 / 32
Channel power budget	1
Latent prediction / state reconstruction loss weights	1 / 500
Controller reward	Open-loop empowerment estimate in bits
Action reconstruction output	Tanh in $[-1, 1]$
Action reconstruction / latent norm loss weights	10 / 1

Without switching, the method cannot consistently learn meaningful potential empowerment policies in Ant and BuilderBench. The potential policy ends up far away from the center of the arena/ball/blocks in Ant/AntSoccer/BuilderBench and does not accumulate many achievements in Crafter-Classic.

D.2 Conditional versus full MI.

CMI uses the branching rollout scheme detailed in Section A.2. All other training stays fixed. We observe that using CMI worsens training in the high-dimensional, continuous setting of Ant where it is easy for a properly context-conditioned classifier to overfit. The potential policy does not move much. Meanwhile, using the Joint MI in Crafter-Classic leads to collapsed empowerment values despite high achievement scores. In BuilderBench, CMI and SEE perform similarly. In AntSoccer, SEE kicks the ball further in the skill phase, though both methods learn potential policies that approach the ball.

D.3 Pair-decoder parameterization.

We compare two logits:

$$\text{Difference encoder: } f_\phi(s, s_+, z) = z^\top [\phi(s_+) - \phi(s)] / \tau,$$

$$\text{Pair decoder: } f_\psi(s, s_+, z) = z^\top g_\psi([s, s_+]) / \tau.$$

We use the same network architecture for both logit parameterizations. All other experiment details are identical to base SEE. The pair decoder worsens visitation, target distance, and skill displacement in Ant and AntSoccer, despite having higher learned empowerment estimates in Ant. For example, in AntSoccer, the pair decoder visits 280 cells versus 539 for SEE. The learned skills barely move the

ball. This ablation highlights the importance of regularizing the classifier when doing empowerment maximization in scaled settings.

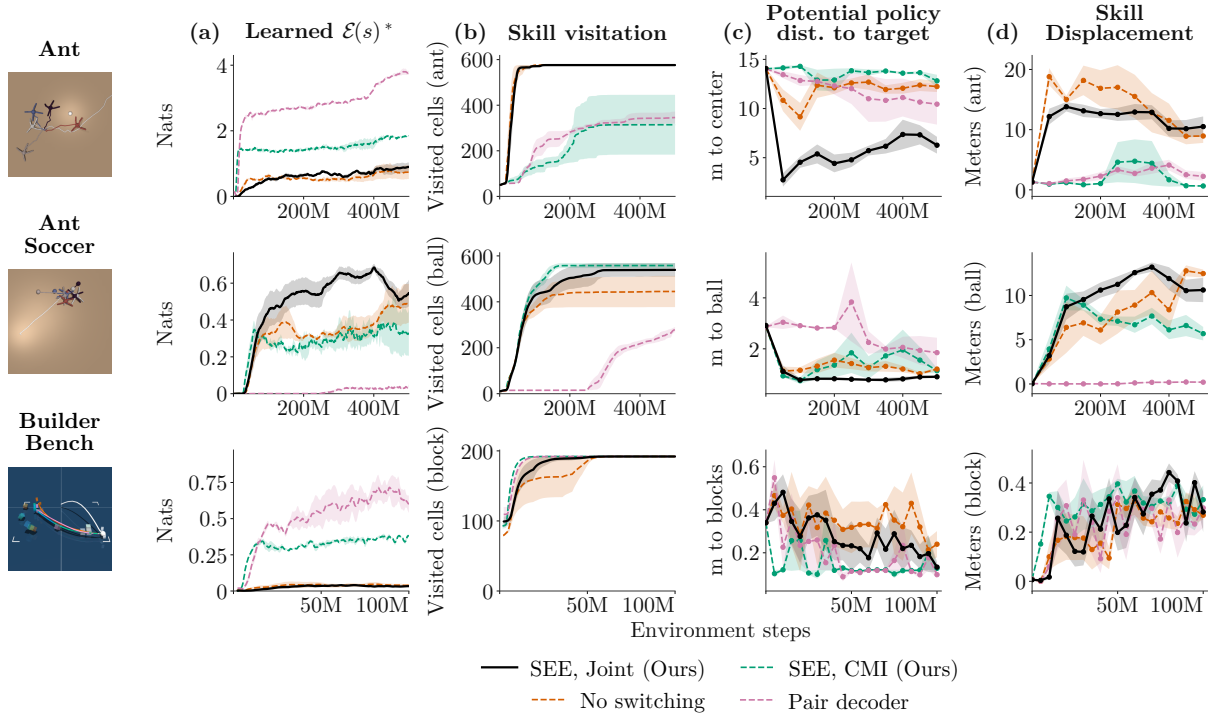


Figure 9: We compare the full method with variants that remove the two-phase rollout (no switching, orange), replace joint MI with conditional MI (green), or replace the bilinear logit with a general MLP (pair decoder, purple). The solid black curve shows SEE, and the colored dashed curves show the ablations. (a) For all methods, average empowerment increases over training. * Learned empowerment estimates use different objectives and regularization, so their magnitudes are not directly comparable across methods. (b) The skill visitation of SEE quickly saturates for these environments. Meanwhile, the ablations struggle to get visitation in a subset of the different environments. (c) SEE consistently learns potential policies that reach the center of the arena (Ant), approach the ball (AntSoccer), and reach the blocks (BuilderBench). In BuilderBench, conditional MI and the pair decoder potential empowerment rollouts reach the blocks, while no switching ends far away. In Ant, all ablations end far from the center of the arena. In AntSoccer, the ablations end up further from the ball than SEE. (d) SEE consistently learns effective policies with large displacements. Five seeds for SEE; three per ablation.

D.4 Reward normalization

We compare the normalization schedule in Section A.3 with fixed $\lambda = 0$ and $\lambda = 1$ (Figs. 11 and 12). Overall, we find that the normalization schedule does not significantly change the training metrics in continuous settings (Fig. 11). However, in Craftax-Classic, normalization ($\lambda = 1$) throughout training significantly hurt the learned average empowerment values (derived from the classifier). In this setting, we find that the normalization schedule is critical to learn value functions that estimate lower bounds of the potential empowerment (Fig. 12).

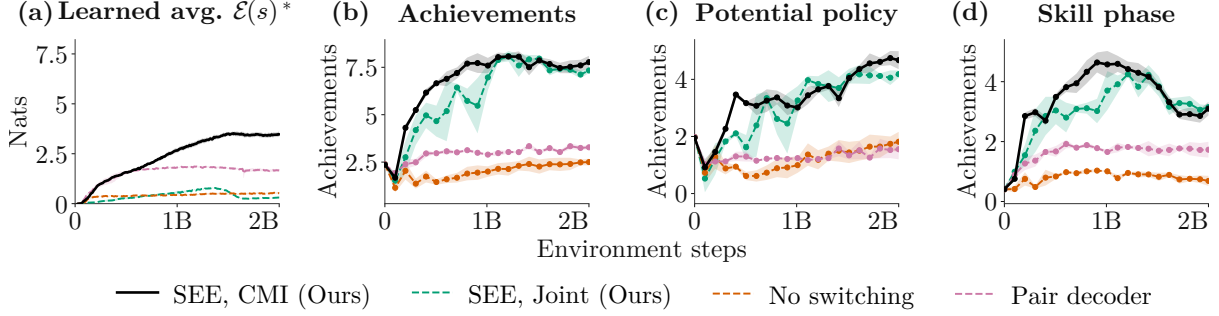


Figure 10: Crafttax-Classic ablations (CMI SEE: 5 seeds; each ablation: 3). The black curve is the SEE with the CMI classifier (Ours). The ablations are SEE with Joint MI (Ours), no switching (25% potential-policy episodes during training), and pair decoder. (a) Learned empowerment estimates rise over training for the base method. (b) Total achievements per rollout. (c) Achievements in the first 100 potential-policy actions. (d) New achievements in the subsequent skill phase of up to 900 actions. All methods use 128 fresh worlds and skills per seed and evaluation checkpoint. Curves show mean $\pm$ standard error across training seeds, with per-seed EMA ($\alpha = 0.05$) only in (a). Achievement panels are unsmoothed and display every tenth evaluation checkpoint plus the final checkpoint.

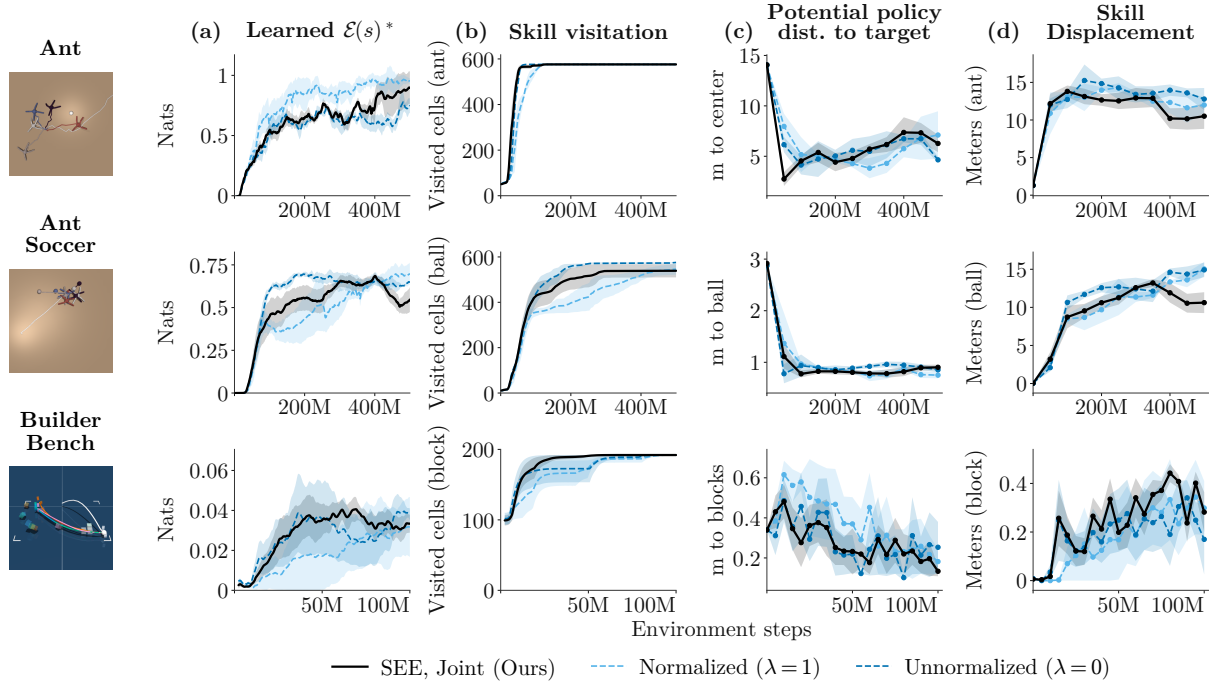


Figure 11: **Reward normalization in continuous environments.** We compare the annealed schedule with normalized ($\lambda = 1$) and unnormalized ($\lambda = 0$) rewards. Columns show learned empowerment, training skill visitation, target distance after 100 potential-policy actions, and displacement during the subsequent 300 skill actions. Learned estimates use different objectives and are not directly comparable across settings. Five seeds for SEE; three per ablation. Curves show mean $\pm$ standard error across seeds. Training curves use per-seed EMA ($\alpha = 0.05$); evaluation curves are unsmoothed.

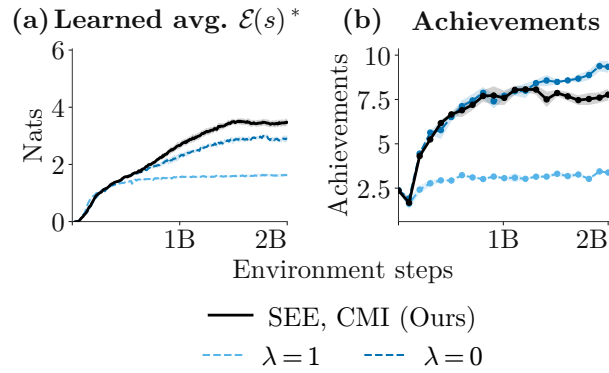


Figure 12: **Reward normalization in Craftax-Classic.** We compare CMI SEE with the annealed schedule against fixed $\lambda = 0$ and $\lambda = 1$. Five seeds for SEE; three per ablation. Curves show mean $\pm$ standard error across seeds. The learned-estimate panel uses per-seed EMA ($\alpha = 0.05$). Achievements use 128 fresh worlds and skills per seed and checkpoint, with 100 potential-policy actions followed by up to 900 skill actions. Achievement evaluations are unsmoothed; every tenth checkpoint and the final checkpoint are shown.

E Additional qualitative examples

We include additional examples of emergent environment modification across all seeds.

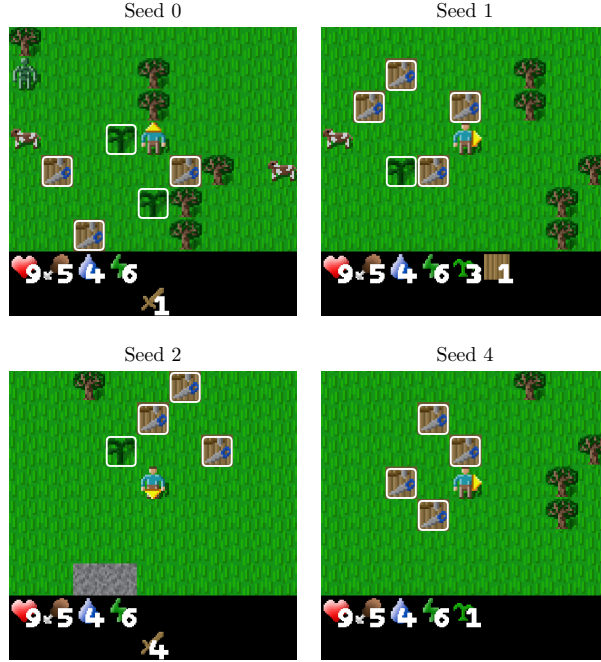


Figure 13: **Additional environment-modification behaviors.** Stills of environment modifications over the remaining 4 seeds in Craftax-Classic. Each panel shows a selected rollout. White outlines box placed tiles.

F Theory

Below, we include the key statement and proof that justifies the empowerment-maximization method. While the particular parameterizations of the underlying MISL methods to maximize potential empowerment are novel, we explicitly note that prior works such as VIC (Gregor et al., 2017) have already shown versions of the same theoretical results. We include these statements for completeness and emphasize that the primary contribution of our work is an empirical demonstration of tractable empowerment maximization, rather than novel theoretical results.

F.1 Policy improvement of the potential empowerment lower-bound

The crux of (most) modern MISL methods is to learn a classifier that identifies skills from states, transitions, or trajectories, then use that classifier as a reward. In expectation, the cumulative reward estimates/lower bounds an MI (see Sections A.1 and 4.1). Policies that maximize this MI become interpretable skills. In our method, we use the resulting MI estimate as a reward, training a **potential empowerment policy** where agents navigate to states where they have many options. We emphasize that this result is a modification of the result proved in Gregor et al. (2017). We do not claim that this is a significant theoretical contribution, and include these results for completeness to show that, under assumptions, our method gives policy improvement. In practical settings with function approximation, these assumptions do not hold.

Here, we prove that alternating (1) fitting an on-policy skill classifier and (2) running policy improvement using the trained classifier as a reward improves a lower bound on the potential empowerment. The result requires two preliminaries. The first preliminary is the Barber-Agakov bound (Barber and Agakov, 2003) which is a variational bound on MI that comes from the positivity of the Kullback-Liebler

Divergence; we have already invoked this bound when introducing the classifier rewards. The second preliminary is the statement that a standard MDP admits greedy policy improvement (Puterman, 2014).

The proofs use the induced posterior q_ϕ , where we drop the ϕ subscript for notational simplicity. Note that we can always recover an effective posterior from the learned classifier, since we know the baseline $p(z)$ exactly.

We fix a start state s_0 and a skill prior $p(z)$ shared across iterations. The discounted future-state distribution is

$$p^\pi(s_+ | s_0, z) = (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k \Pr_\pi(S_{k+1} = s_+ | S_0 = s_0, Z = z).$$

For continuous states, the probabilities are replaced by densities.

Theorem 1 (Policy improvement on a lower bound of potential empowerment). *Let $0 < \gamma < 1$ be the discount factor and let $\pi^{(n)}(a | s, z; s_0)$ be the current skill-conditioned policy at iteration n , with $z \sim p(z)$ sampled at s_0 and held fixed. Fit an on-policy classifier $q^{(n)}(z | s_0, s_+)$ to $p^{\pi^{(n)}}(z | s_0, s_+)$ and define the $(1 - \gamma)$ -normalized action value*

$$\begin{aligned} Q_z^{(n)}(s, a; s_0) &\triangleq Q_z^{\pi^{(n)}, q^{(n)}}(s, a; s_0) \\ &\triangleq (1 - \gamma) \mathbb{E}_{\pi^{(n)}(a | s, z; s_0)} \left[\sum_{k=0}^{\infty} \gamma^k \log \frac{q^{(n)}(z | s_0, s_{t+k+1})}{p(z)} \middle| s_t = s, a_t = a \right]. \end{aligned}$$

Let $\pi^{(n+1)}$ be the result of a greedy policy-improvement step with respect to $Q_z^{(n)}(s, a; s_0)$ for every skill z and start-state s_0 . If the classifier is perfectly fit on-policy at each iteration, $q^{(n)}(z | s_0, s_+) = p^{\pi^{(n)}}(z | s_0, s_+)$, then

$$I^{\pi^{(n+1)}}(Z; S_+ | S_0 = s_0) \geq I^{\pi^{(n)}}(Z; S_+ | S_0 = s_0) \quad \forall n$$

which proves policy improvement under the assumptions of a perfectly-fit classifier.

Improving the estimated potential empowerment-seeking behavior reduces to standard RL with the same classifier intrinsic reward. Let $Q_z^{\pi, q}$ and $V_z^{\pi, q}$ denote values under policy π and classifier q where n indexes training iterations, and $V_z^{\pi, q}(s; s_0) = \mathbb{E}_{a \sim \pi(\cdot | s, z; s_0)}[Q_z^{\pi, q}(s, a; s_0)]$. Then,

$$\widehat{\mathcal{E}}_{\text{pot}}^{(n)}(s) \approx \mathbb{E}_{p(z)} \left[V_z^{\pi^{(n)}, q^{(n)}}(s; s) \right] \tag{11}$$

$$\leq I^{\pi^{(n)}}(Z; S_+ | S_0 = s) \tag{12}$$

$$\leq \mathcal{E}_{\text{pot}}(s) \tag{13}$$

$$= \max_{\pi \in \Pi_Z} I^\pi(Z; S_+ | S_0 = s), \tag{14}$$

which, under the γ -discounted sum and these assumptions, is an unbiased Monte-Carlo estimate of a lower-bound on the cumulative potential empowerment objective:

$$\mathbb{E}_{\pi_{\text{pot}}} \left[\sum_{t \geq 0} \gamma^t \mathbb{E}_{z \sim p} \left[V_z^{\pi^{(n)}, q^{(n)}}(s_t; s_t) \right] \right] \leq \mathbb{E}_{\pi_{\text{pot}}} \left[\sum_t \gamma^t \mathcal{E}_{\text{pot}}(s_t) \right].$$

Proposition 2 (A fixed classifier reward satisfies standard MDP conditions). *Fix a classifier $q(z | s_0, s_+)$ and define the normalized intrinsic reward*

$$r_q(s_0, z, s') \triangleq (1 - \gamma) \log \frac{q(z | s_0, s')}{p(z)}.$$

For any policy π , define

$$Q_z^{\pi,q}(s, a; s_0) \triangleq \mathbb{E}_{\pi(a|s,z;s_0)} \left[\sum_{k \geq 0} \gamma^k r_q(s_0, z, s_{t+k+1}) \mid s_t = s, a_t = a \right].$$

Then $Q_z^{\pi,q}$ satisfies the Bellman equation

$$\begin{aligned} Q_z^{\pi,q}(s, a; s_0) &= (1 - \gamma) \mathbb{E}_{p(s'|s,a)} \left[\log \frac{q(z \mid s_0, s')}{p(z)} \right] \\ &\quad + \gamma \mathbb{E}_{p(s'|s,a)} [V_z^{\pi,q}(s'; s_0)], \end{aligned}$$

where $V_z^{\pi,q}(s; s_0) = \mathbb{E}_{\pi(a|s,z;s_0)} Q_z^{\pi,q}(s, a; s_0)$.

Proof. Re-express the discounted sum in terms of the reward and remaining summation. The classifier is fixed, so r_q is a stationary reward for the MDP conditioned on skill and start-state (z, s_0) . $\square$

Proposition 3 (On-policy classifier gives tight lower bound on MI). *For any policy π and any distribution $q(z \mid s_0, s_+)$,*

$$\begin{aligned} \mathcal{L}(\pi, q) &\triangleq \mathbb{E}_{p(z) p_+^{\pi}(s_+|s_0)} \left[\log \frac{q(z \mid s_0, s_+)}{p(z)} \right] \\ &\leq I^{\pi}(Z; S_+ \mid S_0 = s_0). \end{aligned}$$

Equality holds if $q(z \mid s_0, s_+) = p^{\pi}(z \mid s_0, s_+)$.

Proof. Follows from B-A bound. $\square$

Proof of Theorem 1. At iteration n , fit the classifier $q^{(n)}$ to the on-policy posterior:

$$\mathcal{L}(\pi^{(n)}, q^{(n)}) = I^{\pi^{(n)}}(Z; S_+ \mid S_0 = s_0).$$

Now hold $q^{(n)}$ fixed. By Theorem 2, this is an ordinary MDP with stationary intrinsic reward. A greedy policy-improvement step therefore gives

$$\mathcal{L}(\pi^{(n+1)}, q^{(n)}) \geq \mathcal{L}(\pi^{(n)}, q^{(n)}).$$

Then, refit the classifier to the new on-policy posterior. By Theorem 3, the refitted classifier tightens the lower bound which can only increase the MI lower bound value:

$$\begin{aligned} I^{\pi^{(n+1)}}(Z; S_+ \mid S_0 = s_0) &= \mathcal{L}(\pi^{(n+1)}, q^{(n+1)}) \\ &\geq \mathcal{L}(\pi^{(n+1)}, q^{(n)}) \\ &\geq \mathcal{L}(\pi^{(n)}, q^{(n)}) \\ &= I^{\pi^{(n)}}(Z; S_+ \mid S_0 = s_0). \end{aligned}$$

Thus exact on-policy classifier fitting plus greedy improvement monotonically improves the true MI. $\square$

Importantly, this theoretical result assumes an exact classifier and greedy policy updates at fixed s_0 , which does not happen in practice with function approximation.