The World Is Bigger: A Computationally-Embedded Perspective on the Big World Hypothesis

Alex Lewandowski^{1,2,*} Aditya A. Ramesh³ Edan Meyer^{1,2}
Dale Schuurmans^{1,2,4,5} Marlos C. Machado^{1,2,4}

¹University of Alberta ²Amii ³The Swiss AI Lab IDSIA, USI & SUPSI

⁴Canada CIFAR AI Chair ⁵Google DeepMind

Abstract

Continual learning is often motivated by the idea, known as the big world hypothesis, that the "world is bigger" than the agent. Recent problem formulations capture this idea by explicitly constraining an agent relative to the environment. These constraints lead to solutions in which the agent continually adapts to best use its limited capacity, rather than converging to a fixed solution. However, explicit constraints can be ad hoc, difficult to incorporate, and limiting to the effectiveness of scaling up the agent's capacity. In this paper, we characterize a problem setting in which an agent, regardless of its capacity, is implicitly constrained by being embedded in the environment. In particular, we introduce a *computationally-embedded* perspective that represents an embedded agent as an automaton simulated within a universal (formal) computer. We prove that such an automaton is implicitly constrained and that it is equivalent to an agent that interacts with a partially observable Markov decision process over a countably infinite state-space. We then propose an objective for this setting, which we call *interactivity*, that measures an agent's ability to continually adapt its behaviour and to continually learn new predictions. We develop a reinforcement learning algorithm for maximizing interactivity and a synthetic evaluation task to support experimentation on continual learning. Our results indicate that deep nonlinear networks struggle to sustain interactivity whereas deep linear networks can achieve higher interactivity as capacity increases.

1 Introduction

The goal of this paper is to characterize a general problem setting in which the best use of an agent's limited capacity is to continually adapt (Abel et al., 2023). Our approach is motivated by the idea, known as the big world hypothesis, that the "world is bigger" than the agent (Javed and Sutton, 2024). That is, an agent in a big world may lack the capacity to learn the fixed optimal solution, and should instead continually adapt by updating its approximate solution (*i.e.*, by tracking, Sutton et al., 2007). However, formalizing the relationship between the agent and the environment presents a challenge, because they are typically treated as separate entities in reinforcement learning (see Figures 1b and 1c). We address this challenge by defining a general environment in which an agent can be embedded, and derive a problem setting in which any such agent is (i) implicitly constrained by its capacity, and (ii) suboptimal if it stops learning.

Explicit constraints on the agent have been previously considered in continual learning as a means of capturing the big world hypothesis. For example, in continual learning experiments, it is common practice to constrain what the agent can store (Prabhu et al., 2020), or the capacity of its function approximator (Meyer et al., 2024). Other more general constraints on the agent have also been

^{*}Correspondence to: Alex Lewandowski <lewandowski@ualberta.ca>.

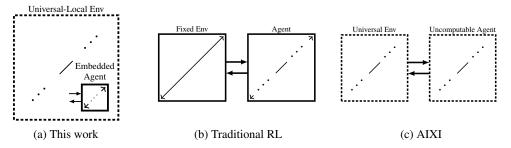


Figure 1: Comparing the agent-environment relationship in different problem formulations. Each problem formulation imposes different constraints on the agent and environment. This work: A universal-local environment embeds the agent by simulating it on its state-space. The agent is implicitly constrained because the environment necessarily has greater complexity than the agent it contains. Traditional RL: A given environment and agents of varying complexity. Agents are unconstrained in principle—they can always be scaled to exceed the environment's complexity. AIXI: A computationally universal environment and an uncomputable agent, neither is constrained.

considered. Such constraints include hardware limits on the agent's compute (see discussion on measuring compute in Section 4.1, Verwimp et al., 2024) and on the agent's energy use (Javed and Sutton, 2024). One recent characterization of continual learning uses an explicit constraint on the agent's information-theoretic capacity (Kumar et al., 2023, 2024). However, beyond analytically tractable agents and environments, this information-theoretic constraint is difficult to measure and verify, limiting its utility for evaluating continual learning agents. In addition, an explicit constraint undermines the effectiveness of scaling up the agent's capacity beyond that constraint, which has been a major source of progress in machine learning more broadly (Hestness et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022). These limitations suggest that explicit constraints may not be an effective way of capturing the big world hypothesis.

In contrast to explicit constraints, our approach considers the implicit constraint that arises from an agent embedded in an environment (see Figure 1a). Specifically, an embedded agent is, in principle, fully defined by the environment's state and dynamics. The embedded nature of intelligent systems—that they exist within and as part of their environment—is typically treated as outside the scope of the problem formulation (Demski and Garrabrant, 2019). However, the physical world is a clear example of a world bigger than any agent, suggesting that a formalization of embedded agency could also provide a natural framework for formalizing the big world hypothesis.

To formalize an agent embedded in an environment, we define a universal-local environment that is capable of simulating an agent on its state-space with its transition dynamics. This environment is a Markov process that is computationally universal, meaning that it is capable of simulating any algorithm (Church, 1936; Turing, 1937). Locality further ensures that the Markov process can be decomposed into a collection of boundaried Markov processes which depend on a finite boundary-space, thus bounding the complexity of the environment's transition dynamics. We use this formalism to show that an embedded automaton can be simulated in the environment as a boundaried Markov process. When the automaton's boundary-space consists of only its input and output, we prove that the automaton interacts with a partially observable Markov decision process by receiving inputs (observations), updating its internal finite-state, and producing outputs (actions). We then propose interactivity, defined through algorithmic complexity (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966), to measure an automaton's ability to adapt its future input-output behaviour sequence, given its past experience. An interactivity-seeking agent produces structured behaviour that balances complexity and predictability, rather than behaviour that is either simple or unpredictable. Crucially, we prove that interactivity-seeking agents are implicitly constrained: an agent with bounded resources has bounded interactivity, but the agent could use additional resources to increase the complexity of its behaviour and improve its predictions.

Universal artificial intelligence similarly considers universal environments (Hutter, 2000, 2005), in which the uncomputable AIXI agent was extended to define an embedded agent (Orseau and Ring, 2012). However, these works did not consider how such an agent could be designed to learn given its limited resources. By considering a universal environment that is also local, we are able to define an interactivity-seeking agent that is implicitly constrained: maximizing interactivity requires continually adapting to experience, regardless of the agent's capacity. Interactivity is similar to

previously considered intrinsic motivation objectives (Chentanez et al., 2004; Schmidhuber, 2010), and specifically predictive information (Bialek et al., 2001; Still and Precup, 2012). However, interactivity uses algorithmic information rather than Shannon information, which can operate directly on individual sequences rather than requiring probability distributions. This sequence-based formulation provides a natural framework for continual adaptation, in which an agent's behaviour is treated as an individual sequence. Furthermore, by tying the agent's objective to its computational capacity, and by bounding that capacity relative to the environment, our formalism captures the big world hypothesis.

Lastly, we demonstrate how the computationally-embedded perspective can be applied in a practical setting by developing a reinforcement learning algorithm to maximize interactivity and a task to evaluate continual adaptation in learning algorithms. Specifically, we recast algorithmic complexity in terms of the prediction error incurred by the agent and view interactivity as the reduction in prediction error due to continual learning, relative to a baseline that stops learning. Interactivity-seeking behaviour thus involves learning a policy to direct the agent's future to new experiences that would have high prediction error without learning, but which are learnable for the agent. We show that interactivity-seeking agents create their own non-stationarity and face the common desideratum of the continual learning problem, in which any agent that stops learning is suboptimal. Our results indicate that deep nonlinear networks struggle to sustain interactivity whereas deep linear networks can achieve higher interactivity as capacity increases.

2 Background

To formalize an embedded agent, we use a computational perspective in which we view the environment as a computation that also simulates the agent. Specifically, we consider a computationally universal environment that can simulate any algorithm through state transitions that implement computational steps. Our approach is general, by making use of the Church-Turing thesis, which states that all computationally universal systems are equivalent in what they can simulate, and that any such system can simulate another (Church, 1936; Turing, 1937). This allows us to adopt a particular model of computation (*e.g.*, Turing machines) while retaining a general class of environments that are capable of simulating an embedded agent.

We characterize the capabilities of an agent, relative to its environment, in terms of its input-output behaviour as a finite sequence (*i.e.*, a string). In particular, we use the algorithmic complexity of a string, which is the length of the shortest program that computes it and halts (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966).

Definition 1 (Algorithmic Complexity). Given strings $x, y \in \Sigma^*$, where Σ is a finite symbol-set and Σ^* is the set of strings, the conditional algorithmic complexity is the length of the shortest program, |c|, that halts and outputs x given y as input,

$$\mathbb{K}(x|y) := \min\{|c| : \mathcal{U}(c,y) = x\},\$$

where \mathcal{U} is a reference universal machine. The unconditional algorithmic complexity is given by $\mathbb{K}(x) := \mathbb{K}(x \mid \epsilon)$, where ϵ is the empty string.

While algorithmic complexity depends on the choice of a reference universal machine, any specific choice affects the algorithmic complexity by, at most, an additive constant independent of the specific string (Li and Vitányi, 2019). Moreover, in this work, we will consider the given computationally universal environment as the canonical universal machine.

3 A Universal-Local Environment

We begin by defining a general class of environments—*universal-local environments*—in which an agent can be embedded. Specifically, such an environment is both computationally universal—capable of simulating any algorithm on its state-space (Section 3.1)—and local—ensuring that any bounded computation is confined to a portion of the environment's state-space (Section 3.2). These two properties will be used to define an embedded agent as an automaton simulated on the state-space of this environment, implicitly constraining the agent's computational capacity (Section 4).

3.1 Markov Representation of a Computationally Universal Environment

We first connect computational processes with Markov processes defined over a countable state-space with a transition function that is computable in polynomial-time with respect to its input size.

Definition 2. An algorithmic Markov process, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$, is a discrete process defined on a countable state-space $\Omega := \{\omega : \Lambda \to \Sigma : |\omega| < \infty\}$ where Σ is a finite symbol set with distinguished blank symbol $\square \in \Sigma$, Λ is a countable set used for indexing, and $|\omega| := |\{\lambda \in \Lambda : \omega(\lambda) \neq \square\}|$ counts the number of non-blank symbols. Given an initial state $\omega \in \Omega$ with $|\omega| < \infty$, the process produces the next state $\omega' = \mathbb{T}(\omega)$, such that $|\omega'| < \infty$ and where the transition function, $\mathbb{T} : \Omega \to \Omega$, is computable in time $O(\operatorname{poly}(|\omega|))$.

The significance of this formulation is that any Turing machine can be represented as an algorithmic Markov process. Specifically, the Markov state represents the entire configuration of the Turing machine, including its head position, current tape contents, and control state. The Markov transition function represents the Turing machine's transition function, which is a lookup table that can be applied to the Markov state in polynomial-time.

Proposition 1 (Turing machines as Markov processes). *The computational process followed by a Turing machine can be represented as an algorithmic Markov process.*

All proofs of propositions and theorems can be found in Section A of the Appendix. One consequence of Proposition 1 is that there exists a universal Markov process—an algorithmic Markov process corresponding to a universal Turing machine. This result highlights how an algorithmic Markov process is more general than the Markov processes typically considered in reinforcement learning. In particular, a universal Markov process is capable of simulating any algorithm, which is crucial for defining an embedded agent in Section 4.

3.2 Defining Locality with Boundaried Markov Processes

Intuitively, locality means that the transition function can be restricted to a bounded portion of the state-space, which we define as a *substate-space*, corresponding to a finite index-set.

Definition 3. Given an algorithmic Markov process, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$, a substate-space, $\Omega|_F \hookrightarrow \Omega$ (where \hookrightarrow denotes an embedding), is defined as a restriction of the state-space to a finite index-set, $F \in \mathcal{F}(\Lambda) := \{I : I \subset \Lambda, |I| < \infty\}$, such that $\Omega|_F = \{\omega|_F : \omega \in \Omega\}$ where $\omega|_F = \{\omega(\lambda)\}_{\lambda \in F}$

We now consider the transition function restricted to a substate-space, $\Omega|_F \hookrightarrow \Omega$. In particular, we define a boundaried Markov process in which the transition dynamics on the substate-space, $\mathbb{T}|_F$, depends on another substate-space, $\Omega|_{b(F)} \hookrightarrow \Omega$, referred to as the boundary-space of $\Omega|_F$.

Definition 4. Given an algorithmic Markov process, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$, a substate-space, $\Omega|_F$, admits a k-horizon boundaried Markov process if there exists a boundary-space, $\Omega|_{b^k(F)} \hookrightarrow \Omega$ and a restricted transition function $\mathbb{T}|_F^k: \Omega|_F \times \Omega|_{b^k(F)} \to \Omega|_F$ that is equivalent to the k-step transition function on the substate-space: $\mathbb{T}|_F^k(\omega|_F, \omega|_{b^k(F)}) = \mathbb{T}^{(k)}(\omega)|_F$ for all $\omega \in \Omega$. When such a boundary exists, we denote the k-horizon boundaried Markov process as $\mathcal{E}|_F^k = (\Omega|_F, \Omega|_{b^k(F)}, \mathbb{T}|_F^k)$.

Typically, the size of the boundary-space increases as the transition horizon k becomes larger. This is because the current substate, $\omega|_F \in \Omega|_F$, and the current boundary, $\omega|_{b(F)} \in \Omega|_{b(F)}$, only define the next substate, $\omega|_F' \in \Omega|_F$, and not the next boundary. We now characterize an algorithmic Markov process as uniformly local if it can be factored into 1-horizon boundaried Markov processes.

Definition 5 (Uniform Locality). An algorithmic Markov process, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$, is uniformly local if there exists a polynomial-time boundary function, $b: \mathcal{F}(\Lambda) \to \mathcal{F}(\Lambda)$ and a reference $\lambda_{\star} \in \Lambda$, such that every $\lambda \in \Lambda$ admits an isomorphic boundaried Markov process: there exist bijections $\alpha_{\lambda}: \Omega|_{\{\lambda_{\lambda}\}} \to \Omega|_{\{\lambda_{\star}\}}$ and $\beta_{\lambda}: \Omega|_{b(\{\lambda_{\lambda}\})} \to \Omega|_{b(\{\lambda_{\star}\})}$ satisfying, for all $\omega \in \Omega$,

$$\alpha_{\lambda}\left(\mathbb{T}|_{\{\lambda\}}(\omega|_{\{\lambda\}},\omega|_{b(\{\lambda\})})\right) = \mathbb{T}|_{\{\lambda_{\star}\}}(\alpha_{\lambda}(\omega|_{\{\lambda\}}),\beta_{\lambda}(\omega|_{b(\{\lambda\})})).$$

Uniform locality ensures that the transition function is structurally similar across the state-space, that each singleton boundary is constant, $|b(\{\lambda\})| = c$, for constant c, and that every substate-space admits a boundaried Markov process, using the boundary function $b(F) = \bigcup_{\lambda \in F} b(\{\lambda\})$.

Thus, we use the term *universal-local environment* for a universal Markov process that is also uniformly local. As we will show, an embedded agent can be simulated as a boundaried Markov process within the universal-local environment. The environment terminology reflects its role as the setting in which an embedded agent operates.

In this paper, we consider deterministic transitions. For non-deterministic transitions, \mathbb{T} becomes a relation where multiple next states $\omega' \in \mathbb{T}(\omega)$ are possible.

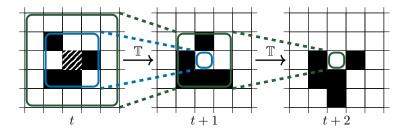


Figure 2: Conway's Game of Life is a cellular automaton and an example of a universal-local environment. The state-space is an infinite 2D grid, $\Lambda = \mathbb{Z}^2$, where each cell takes on one of two values, $\Sigma = \{\text{alive, dead}\}$. Each cell has a local transition function in which it lives (black) with 2 or 3 neighbours, but dies (white) otherwise, and dead cells with 3 neighbours become alive. The blue and green borders (*left*) correspond to neighbourhoods that determine the middle cell at time-steps t+1 (*middle*) and t+2 (*right*). Longer-term transition dynamics depend on larger neighbourhoods.

3.3 Example of a Universal-Local Environment: Conway's Game of Life

Conway's Game of Life is an example of a universal-local environment (Conway, 1970). This environment is computationally universal because it can simulate a universal Turing machine (Berlekamp et al., 1982; Rendell, 2011). A substate-space in Conway's Game of Life is a finite subset of locations on the grid, specifying the possible values taken by the cells at those locations. Conway's game of life is local because the one-step transition dynamics for each individual cell depend on only the adjacent neighbourhood of that cell, which defines the boundary-space (see Figure 2).

While Conway's Game of Life has the potential to simulate any algorithm using its local dynamics, we are not suggesting to program an agent within it. We only point out Conway's Game of Life as a proof-of-existence for universal-local environments. Instead, we consider and formalize the implicit constraints that would be faced by an agent if it were embedded in any such environment.

4 A Computationally-Embedded Agent

We formalize an embedded agent as an automaton simulated within the univeral-local environment as a boundaried Markov process. The boundary-space acts as an interface that separates the automaton from the rest of the universal-local environment (Jiang, 2019; Harutyunyan, 2020). When this boundary-space consists of only the automaton's input and output space, we prove that the automaton interacts with a partially observable Markov decision process. We then propose *interactivity* to measure the automaton's capability for continual adaptation, and prove that an interactivity-seeking automaton is implicitly constrained: the maximum achievable interactivity is upper bounded by the automaton's finite capacity.

4.1 Embedding an Automaton in a Universal-Local Environment

A universal-local environment can simulate any algorithm; this property enables us to define an automaton, \mathcal{A} , on the environment's state-space, Ω such that its computations are performed through the environment's transition dynamics, \mathbb{T} . Moreover, locality ensures that the automaton can be represented as a boundaried Markov process on a substate-space (see Figure 3, left).

Definition 6. Given a universal-local environment, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$, an embedded automaton is defined by $\mathcal{A} := (\Omega|_X, \Omega|_Y, \Omega|_Z, u, \pi)$, where $\Omega|_Z \hookrightarrow \Omega$ is the internal substate-space of the automaton, $\Omega|_X, \Omega|_Y \hookrightarrow \Omega$ are input and output spaces, $u : \Omega|_Z \times \Omega|_X \times \Omega|_Y \to \Omega|_Z$ is the internal substate update function, and $\pi : \Omega|_Z \times \Omega|_X \times \Omega|_Y \to \Omega|_Y$ is the output function. The automaton iteratively receives an input from the environment, x, and produces its next internal substate, z' = u(z, x, y), and next output, $y' = \pi(z, x, y)$.

Relating this to an agent in reinforcement learning, we may think of the inputs as observations, the internal substate as the parameters of a function approximator, the outputs as actions, the substate update function as a learning rule, and the output function as a policy.

The input-space may also provide an external reward to the automaton, but this need not be the case.

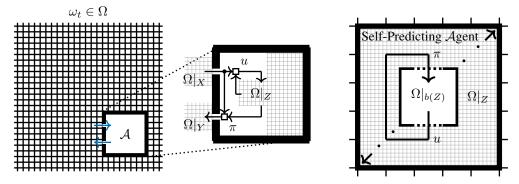


Figure 3: An illustrative depiction of a computationally-embedded agent interacting with its environment. An embedded automaton, $\mathcal{A}=(\Omega|_X,\Omega|_Y,\Omega|_Z,u,\pi)$, simulates an agent on a substate-space of the universal-local environment. Left: At each step, an automaton with subtate z and with previous output y iteratively reads from its input-space, $x\in\Omega|_X\hookrightarrow\Omega$, updates its internal substate, $z'=u(z,x,y)\in\Omega|_Z\hookrightarrow\Omega$, and writes to its output-space, $y'=\pi(z,x,y)\in\Omega|_Y\hookrightarrow\Omega$. Right: We will also consider an idealized setting in which a self-predicting agent exerts full control over its experience by reading and writing to an internal boundary-space, $\Omega|_{b(Z)}$.

Proposition 2 (Automaton-Environment Relationship). Given an embedded automaton, $\mathcal{A} = (\Omega|_X, \Omega|_Y, \Omega|_Z, u, \pi)$, if there exists a horizon k such that $b^k(Z) = X \cup Y$, then:

- 1. The automaton is equivalent to a k-horizon boundaried Markov process
- 2. The automaton's environment is a partially observable Markov decision process.
- 3. The automaton's interaction is equivalent to a stateful policy acting on the environment

Now that we have defined both the embedded automaton and its environment within the same universal-local environment, we can show that any such automaton is constrained by the size of its internal substate, which determines its memory and computational capacity.

Proposition 3 (Implicitly Constrained). In a universal-local environment, any embedded automaton is constrained by the size of its internal substate-space: there exists an input-output behavioural sequence that the automaton cannot realize.

While every embedded automaton is constrained by its internal substate-space, which is finite, simple behavioural sequences can be realized within this constraint. For example, any automaton could produce a constant output sequence without being constrained by its substate-space. We will show, however, that every embedded automaton is constrained by the finiteness of its internal substate-space when adapting to its past input-output experience.

4.2 Interactivity as a Computational Measure of Adaptivity

An agent's capability for learning can be characterized by its ability to adapt its future behaviour using its past experience. We propose *interactivity* to measure this capability directly in terms of algorithmic complexity. Specifically, interactivity measures the difference between the algorithmic complexity of future behaviour with and without conditioning on past experience.

Following Proposition 2, we represent an embedded agent as an embedded automaton $\mathcal A$ where its input and output spaces constitute its boundary-space, $\Omega|_{X\cup Y}=\Omega|_{b^k(Z)}$. Thus, the behaviour of the agent is determined by the values taken on the boundary-space, $b_t\in\Omega|_{b^k(Z)}$ which combines the input and the corresponding output, $b_t=(x_t,y_t)$. At time t, the behaviour can be separated into finite sequences: the past, $b_{0:t-1}:=b_0b_1\cdots b_{t-1}$, and the T-horizon future, $b_{t:t+T-1}:=b_tb_{t+1}\cdots b_{t+T-1}$.

Definition 7 (Interactivity). Given an embedded automaton, $\mathcal{A} := (\Omega|_X, \Omega|_Y, \Omega|_Z, u, \pi)$, we define its T-horizon interactivity as the difference between the algorithmic complexity of its future behaviour with and without conditioning on its past behaviour,

$$\mathbb{I}_T(\mathcal{A} \mid x_t, b_{0:t-1}) := \mathbb{K}(b_{t:t+T-1} \mid \epsilon) - \mathbb{K}(b_{t:t+T-1} \mid b_{0:t-1}),$$

where ϵ is the empty string, $y_t = \pi(b_{t-1}, z_{t-1})$ is the action, $z_t = u(b_{t-1}, z_{t-1})$ is the internal substate, x_t is the current observation, and $b_t = (x_t, y_t)$ is the next behaviour tuple.

That is, interactivity measures the predictable complexity of an agent's future behaviour, given its past behaviour. Interactivity is high if both (i) the future behaviour, $b_{t+1:T}$, has high unconditional algorithmic complexity, and (ii) the past behaviour, $b_{0:t}$, is predictive of this future behaviour, thereby yielding a low conditional algorithmic complexity. However, interactivity is low if the future behaviour has low algorithmic complexity, or if the past behaviour is not sufficiently predictive.

4.3 An Interactivity-Seeking Agent Faces a Big World

The interactivity of any embedded agent is always constrained by its internal substate-space. That is, an embedded agent that seeks to maximize its interactivity can only sustain a given level of interactivity with a given internal substate-space. However, an embedded agent with a larger internal substate-space could use the additional resources to achieve higher interactivity.

Theorem 1. Given an embedded agent, $A = (\Omega|_X, \Omega|_Y, \Omega|_Z, u, \pi)$, its maximum interactivity is upper bounded by a quantity depending on the size of its internal substate-space, |Z|.

The goal of an interactivity-seeking agent can be understood as balancing its future complexity with its future predictability given its past behaviour. Seeking to achieve this goal requires that the agent continually adapt to its experience so that it learns to track its environment as it changes, and to discover its computational limitations. This suggests the following interactivity thesis:

Interactivity measures a relative capability for continual adaptation.

We refer to this as the interactivity thesis, rather than a hypothesis, to reflect its speculative and philosophical nature. With low interactivity, an agent's capability for continual adaptation is limited because its future behaviour is either: i) simple, or ii) complex and unpredictable. In either case, the thesis stresses the relative notion of capabilities. A simple agent has a limited range of possible behaviours and thus has a relatively lower capability for adaptation. A complex agent could have a relatively high capability for adaptation, but only if its future behaviour is influenced by, and can be predicted from, past experience. Embracing the interactivity thesis naturally leads to a spectrum of adaptive capability.

5 Maximizing Agent-Relative Interactivity with Reinforcement Learning

Maximizing interactivity poses a fundamental computational challenge: it depends on algorithmic complexity, which is generally uncomputable. While algorithmic complexity can be computed for finite automata through exhaustive program enumeration (Li and Vitányi, 2019), this requires computational resources that exceed the automaton's own capacity. Consequently, an interactivity-seeking agent must approximate interactivity rather than compute it exactly.

To approximate interactivity, we take a distortion-rate view of algorithmic complexity, which measures complexity in terms of prediction error under a constrained reference machine, rather than under an unconstrained universal Turing machine (Vereshchagin and Vitányi, 2010). The embedded agent, being an automaton, provides a naturally constrained reference machine. We then augment this agent to include a predictor function on its finite substate-space, and measure agent-relative complexity by the incurred temporal difference errors under this predictor.

Definition 8. Given an embedded agent $A = (\Omega|_X, \Omega|_Y, \Omega|_Z, u, \pi, v, \gamma)$ with a predictor $v : \Omega|_{X \cup Y} \times \Omega|_Z \to \Omega|_{X \cup Y}$ and $\gamma \in [0, 1]$, the agent-relative complexity of future behaviour $b_{t:t+T-1}$, conditioned on past behaviour $b_{0:t-1}$, is the sum of future temporal difference errors,

$$\mathbb{K}_{\mathcal{A}}(b_{t:t+T-1}|b_{0:t-1}) := \sum_{k=0}^{T-1} \left(b_{t+k} + \gamma v(b_{t+k}, z_{t+k-1}) - v(b_{t+k-1}, z_{t+k-1})\right)^{2},$$

where z_{t+k-1} is the agent's internal substate after processing $b_{0:t-1}$ followed by $b_{t:t+k-1}$ through repeated application of the substate update function u.

Complexity, measured in this way, is relative to the agent's capabilities. If the future prediction errors are large (small), then the future behaviour is relatively complex (simple). These future prediction errors depend crucially on the agent through its predictor, its current internal substate, its learning algorithm, its policy, and its observations from the environment. With this prediction error formulation, we now consider the agent-relative interactivity,

$$\hat{\mathbb{I}}_T(\mathcal{A} \mid x_t, b_{0:t-1}) := \mathbb{K}_{\mathcal{A}}(b_{t+1:t+T} \mid \epsilon) - \mathbb{K}_{\mathcal{A}}(b_{t+1:t+T} \mid b_{0:t-1}).$$

5.1 Learning to Maximize Agent-Relative Interactivity

We now develop a reinforcement learning algorithm for maximizing agent-relative interactivity. Our algorithm is summarized in the following three steps: (i) learning a prediction of the discounted future behaviour using a value function, (ii) computing the agent-relative interactivity, defined as the difference between static prediction errors (unconditional complexity) and dynamic prediction errors (conditional complexity), and (iii) meta-learning a policy to maximize agent-relative interactivity.

The first step involves learning a prediction of the future input-output behaviour, in which we consider the deterministic setting and omit the expectation over the policy and environment. This value function predicts both future observations and actions, generalizing successor features (Barreto et al., 2017) and the successor representation (Dayan, 1993). Specifically, with temporal difference learning (Sutton, 1984, 1988), we train a value function to predict the discounted sum of future input-output behaviour,

$$v(b_t, z_t) \approx \sum_{k=0}^{\infty} \gamma^k b_{t+k+1}, \quad \delta_{t+k,\gamma}(z_{t+k}) = b_{t+k} + \gamma v(b_{t+k}, z_{t+k-1}) - v(b_{t+k-1}, z_{t+k-1}),$$

where each z_{t+k} can be understood as dynamic parameters which are updated using semi-gradient TD(0): $z_{t+k+1} = z_{t+k} + \eta \delta_{t+k,\gamma}(z_{t+k}) \nabla_z v(b_{t+k},z) \big|_{z=z_{t+k}}$, where η is a step-size.

For the second step, we measure the conditional and unconditional complexity terms using dynamic and static temporal difference errors, respectively. From the first step, we can readily compute the conditional complexity using the dynamic temporal difference errors just described,

$$\mathbb{K}_{\mathcal{A}}(b_{t+1:t+T}|b_{0:t-1}) = \sum_{k=1}^{T} \delta_{t+k,\gamma}^{2}(z_{t+k}).$$

However, conditional complexity alone is not sufficient for maximizing interactivity, as it would otherwise result in reaching a fixed point in which the behaviour is perfectly predicted by the value function. In contrast, the definition of unconditional complexity does not involve previous experience, meaning that it cannot be computed using the dynamic evolution from the current substate, z_t . Instead, computing the unconditional complexity involves a static and unchanging reference state, z_{ref} . The most convenient choice for which is the current state, $z_{ref} := z_t$, which yields the following form for agent-relative interactivity,

$$\mathbb{I}(\mathcal{A}_t) = \sum_{k=t}^{T} \left(\underbrace{\delta_{t+k}^2(z_t)}_{\text{static}} - \underbrace{\delta_{t+k}^2(z_{t+k})}_{\text{dynamic}} \right).$$

Lastly, we outline how a policy can be trained to maximize interactivity. To obtain the future prediction errors, we assume access to an environment model that we can use to roll-out a sequence of observations and actions from the current policy. Model-based reinforcement learning can then be used to maximize interactivity: the policy is updated using a gradient of the cumulative prediction error differences computed from the roll-out, thus estimating the agent-relative interactivity. Interactivity-seeking behaviour is directed towards experience which can be predicted by the dynamic value function, but which cannot be predicted by any static value function. Thus, to sustain interactivity, an agent would need to continually adapt both its policy and its value function.

5.2 Maximizing Interactivity as a Continual Learning Problem

We now show that an interactivity-seeking reinforcement learning agent faces a continual learning problem. Similar to our previous result on embedded automata (Theorem 1), we show that, as the agent's capacity increases, so too does its maximum possible interactivity. Moreover, interactivity-seeking agents are suboptimal if they stop learning, such as when the parameters of their policy and value stop changing.

Theorem 2 (Big World). An agent that seeks to maximize its agent-relative interactivity is (i) limited by its finite capacity and, (ii) suboptimal if it stops learning.

This optimization problem involves meta-learning because the dynamic prediction errors depend on the value function's learning process.

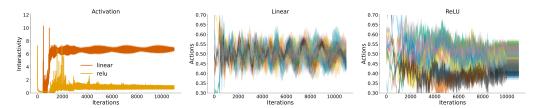


Figure 4: **Deep** ReLU **networks fail to sustain performance on the interactivity evaluation task.** Interactivity maximization is a big world simulator that directly evaluates a learning algorithm's capability for continual adaptation. **Right:** The deep linear network sustains interactivity, whereas the deep ReLU network performs progressively worse. **Middle:** Each colour corresponds to an action index. The deep linear policy learns to produce actions following a non-stationary wave, which can be locally predicted by a static linear function and globally predicted by a dynamic linear function. **Left:** The deep ReLU policy fails to produce actions with any predictable structure, thus highlighting the unique challenge provided by the interactivity evaluation task.

The desiderata of Theorem 2 were previously described as conditions for a big world simulator (Kumar et al., 2024). Thus, interactivity maximization appears to be a general problem setting which captures the big world hypothesis: the best use of an agent's limited resources is to continually adapt.

6 Evaluating Continual Adaptation With Agent-Relative Interactivity

We now pose interactivity maximization as a behavioural self-prediction task, and use it as a synthetic task to evaluate algorithms for continual learning. In particular, we consider a setting in which the agent only observes its own actions (see the self-predicting agent in Figure 3, right). Even though such an agent has full control over its experience stream, it is still implicitly constrained. That is, the interactivity objective depends on the parameters and learning algorithm of the value function, which the agent does not directly observe. The advantage of this type of evaluation task is that it does not require an external environment, or any collected data. Instead, algorithms are directly evaluated by continually learning from their own online experience in a manner similar to self-play.

We instantiate the reinforcement learning agent outlined in Section 5 with a linear parameterization of the value function, $v(b_t, \mathbf{W}_t) := \mathbf{W}_t b_t \approx \sum_{k=0}^{\infty} \gamma^k b_{t+k+1}$. Linearity provides stability for learning online with TD(0), unlike the instability of temporal difference learning with deep nonlinear networks. For the policy parameterization, we consider a deep network architecture (using either linear or ReLU activations), where we normalize the output to ensure that the output has bounded range, $b_{t+1} := \text{RMSNorm}\left(\pi(b_t; \theta_t)\right)$. This policy is optimized using the model-based approach described in Section 5. Optimizing the agent-relative interactivity is a bi-level optimization problem: the dynamic prediction errors depend on the value function's learning process. That is, gradient-based optimization, using auto-differentiation, implements an online version of model-agnostic metalearning (Finn et al., 2017), similar to the cross-prop algorithm (Veeriah et al., 2017). For both the policy and the value function, we found RMSProp (Hinton et al., 2012) to balance performance and stability better than either Adam (Kingma and Ba, 2015), or conventional gradient descent.

Our results demonstrate that, across a variety of network sizes, the deep nonlinear policy was unable to sustain interactivity with a horizon of T=10 (see Figure 4, right). That is, the deep nonlinear policy is unable to plan an action sequence for which the dynamic value function has low prediction error, but for which the current static value function has high prediction error. In contrast, we find that a deep linear policy is able to sustain interactivity. This finding suggests that interactivity-seeking agents produce non-stationarity that can lead to apparent loss of plasticity, which linear methods have been shown to avoid (Lewandowski et al., 2025b; Dohare et al., 2024). Observing the actions chosen by each policy, we found that the deep linear policy learned to produce structured actions, resembling a non-stationary wave (see Figure 4, middle). These actions can be locally predicted by a linear function, but global prediction is impossible with a fixed linear function. In comparison, the deep nonlinear policy learned failed to produce actions with any predictable structure (see Figure 4, left). Furthermore, in Figure 5, we found that deep linear networks are also capable of increasing their interactivity with more resources, in the form of deeper or wider networks. These findings provide empirical support that the interactivity task is useful for evaluating continual learning algorithms as a big world simulator; even given a linear predictor, the policy is empirically: (i) limited by its finite capacity and, (ii) suboptimal if it stops learning.

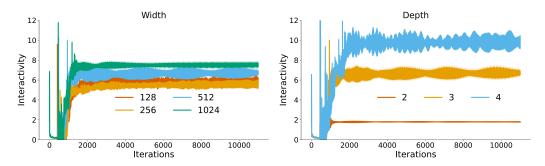


Figure 5: Deep linear networks are capable of sustaining higher interactivity with more computational resources. By increasing the width and depth of the deep linear network, we increase the network's computational resources, which allows it more quickly adapt its linear function approximator. Left: Increasing width marginally increases the sustained interactivity. Right: Increasing depth results in a large increase in interactivity, as well as more oscillatory behaviour.

7 Discussion

In this paper, we introduced a computationally-embedded perspective on the big world hypothesis, which considers the implicit constraint faced by an embedded agent. Our work (i) characterizes the implicit constraint faced by an embedded agent, (ii) proposes interactivity as a computational measure of adaptivity, and (iii) develops a reinforcement learning algorithm for maximizing interactivity. We show that interactivity maximization leads to the common desideratum of the continual learning problem in which any agent that stops learning is suboptimal.

Our work departs from dogmas common in reinforcement learning (Abel et al., 2024) by formalizing the agent as embedded in the environment, and by proposing interactivity which depends both on the environment and on the agent's internals. This perspective has a critical consequence: an interactivity-seeking agent must continually adapt rather than converging to a fixed point. Such an agent is implicitly constrained because interactivity depends on components, such as parameters, that are not directly observable but evolve through learning. Thus, our work formally captures the big world hypothesis: no matter the agent's capacity, optimal behaviour requires continual adaptation.

Building on the computationally-embedded perspective, we develop a synthetic task that evaluates an algorithm's capability for continual learning in a big world setting. This evaluation task is derived from maximizing interactivity without an external environment. Unlike other tasks in continual learning, the non-stationarity is autonomously produced by the learning algorithm, rather than manually produced by the researcher. Moreover, the complexity of the task scales with the complexity of the learning algorithm. Our results show that typical deep nonlinear networks are unable to sustain interactivity, whereas deep linear networks can sustain interactivity and improve with more resources. Further empirical work is needed to design algorithms that combine the representational capacity of deep nonlinear networks with the stability of deep linear networks, such as using recurrent networks to summarize past behaviour, and applying regularizers designed for continual learning (Lewandowski et al., 2025a). Interactivity maximization thus provides a unique evaluation task that integrates the central continual learning challenges inherent to a big world.

Beyond serving as an evaluation task for continual learning, interactivity maximization can serve as an auxiliary objective for agents in conventional reinforcement learning environments. Specifically, as an intrinsic reward, interactivity maximization would incentivize directed exploration similar to curiosity-driven methods. To maximize interactivity with an external environment, the agent would need to learn an environment model to apply our meta-gradient approach, or learn a value function that directly approximates interactivity.

This practical potential motivates a refinement of the interactivity thesis: if an agent can sustain a particular level of interactivity, then it possesses the adaptive capacity to learn any behaviour with equal or lower interactivity—including those that maximize external reward. Since every behaviour has measurable interactivity, sustaining high interactivity indicates the adaptive capacity needed to learn other goal-directed behaviours.

Acknowledgments

The research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chair Program, and the Digital Research Alliance of Canada.

References

- Abel, D., Barreto, A., Van Roy, B., Precup, D., van Hasselt, H. P., and Singh, S. (2023). A definition of continual reinforcement learning. In *Neural Information Processing Systems*.
- Abel, D., Ho, M. K., and Harutyunyan, A. (2024). Three dogmas of reinforcement learning. *Reinforcement Learning Journal*, 1.
- Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., and Silver, D. (2017). Successor features for transfer in reinforcement learning. In *Neural Information Processing Systems*.
- Barto, A. G. (2013). Intrinsic motivation and reinforcement learning. *Intrinsically motivated learning in natural and artificial systems*, pages 17–47.
- Berlekamp, E. R., Conway, J. H., and Guy, R. K. (1982). Winning Ways for Your Mathematical Plays, Vol. 2. Academic Press.
- Bialek, W., Nemenman, I., and Tishby, N. (2001). Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463.
- Chaitin, G. J. (1966). On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569.
- Chentanez, N., Barto, A. G., and Singh, S. (2004). Intrinsically motivated reinforcement learning. In *Neural Information Processing Systems*.
- Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2(2):137–167.
- Church, A. (1936). An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363.
- Conway, J. H. (1970). The game of life. Scientific American, 223(4):4.
- Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5:613–624.
- Demski, A. and Garrabrant, S. (2019). Embedded agency. CoRR, abs/1902.09469v3.
- Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., and Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, 632:768–774.
- Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*.
- Flennerhag, S., Schroecker, Y., Zahavy, T., van Hasselt, H., Silver, D., and Singh, S. (2022). Bootstrapped meta-learning. In *International Conference on Learning Representations*.
- Friston, K. J., Daunizeau, J., Kilner, J., and Kiebel, S. J. (2010). Action and behavior: a free-energy formulation. *Biological cybernetics*, 102:227–260.
- Gregor, K., Rezende, D. J., and Wierstra, D. (2016). Variational intrinsic control. *arXiv preprint* arXiv:1611.07507.
- Grossberg, S. and Grossberg, S. (1982). How does a brain build a cognitive code? *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, pages 1–52.
- Harutyunyan, A. (2020). What is an agent?

- Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.
- Hinton, G., Srivastava, N., and Swersky, K. (2012). Neural networks for machine learning. lecture 6a. overview of mini-batch gradient descent.
- Hoffmann, J. et al. (2022). Training compute-optimal large language models. *CoRR*, abs/2203.15556v1.
- Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). Vime: Variational information maximizing exploration.
- Hutter, M. (2000). A theory of universal artificial intelligence based on algorithmic complexity. arXiv preprint cs/0004001.
- Hutter, M. (2005). Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability. Springer, Berlin.
- Javed, K. and Sutton, R. S. (2024). The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame Workshop at Reinforcement Learning Conference*.
- Jiang, N. (2019). On value functions and the agent-environment boundary. CoRR, abs/1905.13341v3.
- Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.
- Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.
- Klyubin, A. S., Polani, D., and Nehaniv, C. L. (2005). Empowerment: A universal agent-centric measure of control. In 2005 ieee congress on evolutionary computation, volume 1, pages 128–135. IEEE.
- Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11.
- Kumar, S., Jeon, H. J., Lewandowski, A., and Van Roy, B. (2024). The need for a big world simulator: A scientific challenge for continual learning. In *Finding the Frame Workshop at Reinforcement Learning Conference*.
- Kumar, S., Marklund, H., Rao, A., Zhu, Y., Jeon, H. J., Liu, Y., and Van Roy, B. (2023). Continual learning as computationally constrained reinforcement learning. *CoRR*, abs/2307.04345.
- Lewandowski, A., Bortkiewicz, M., Kumar, S., György, A., Schuurmans, D., Ostaszewski, M., and Machado, M. C. (2025a). Learning continually by spectral regularization. In *International Conference on Learning Representations*.
- Lewandowski, A., Schuurmans, D., and Machado, M. C. (2025b). Plastic learning with deep fourier features. In *International Conference on Learning Representations*.
- Li, M. and Vitányi, P. M. B. (2019). *An Introduction to Kolmogorov Complexity and Its Applications*, 4th Edition. Texts in Computer Science. Springer.
- Mesbahi, G., Panahi, P. M., Mastikhina, O., Tang, S., White, M., and White, A. (2025). Position: Lifetime tuning is incompatible with continual reinforcement learning. In *International Conference on Machine Learning, Position Paper Track*.
- Meyer, E. J., White, A., and Machado, M. C. (2024). Harnessing discrete representations for continual reinforcement learning. *Reinforcement Learning Journal*, 2:606–628.
- Mohamed, S. and Jimenez Rezende, D. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. volume 28.

- Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms. CoRR, abs/1803.02999v3.
- Orseau, L. and Ring, M. (2012). Space-time embedded intelligence. In *International Conference on Artificial General Intelligence*.
- Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71.
- Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). GDumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*.
- Rendell, P. (2011). A universal turing machine in Conway's game of life. In *International Conference on High Performance Computing & Simulation*.
- Sajid, N., Ball, P. J., Parr, T., and Friston, K. J. (2021). Active inference: demystified and compared. *Neural computation*, 33(3):674–712.
- Schmidhuber, J. (2010). Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247.
- Solomonoff, R. J. (1964). A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22.
- Still, S. and Precup, D. (2012). An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131:139–148.
- Storck, J., Hochreiter, S., and Schmidhuber, J. (1995). Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. EC2 & Cie.
- Sutton, R. S. (1984). Temporal credit assignment in reinforcement learning. PhD thesis, University of Massachusetts Amherst.
- Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44.
- Sutton, R. S., Koop, A., and Silver, D. (2007). On the role of tracking in stationary environments. In *International Conference on Machine Learning*.
- Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265.
- Veeriah, V., Zhang, S., and Sutton, R. S. (2017). Crossprop: Learning representations by stochastic meta-gradient descent in neural networks. In *Machine Learning and Knowledge Discovery in Databases*.
- Veness, J., Ng, K. S., Hutter, M., Uther, W. T. B., and Silver, D. (2011). A monte-carlo AIXI approximation. *J. Artif. Intell. Res.*, 40:95–142.
- Vereshchagin, N. K. and Vitányi, P. M. (2010). Rate distortion and denoising of individual data using kolmogorov complexity. *IEEE Transactions on Information Theory*, 56:3438–3454.
- Verwimp, E., Aljundi, R., Ben-David, S., Bethge, M., Cossu, A., Gepperth, A., Hayes, T. L., Hüllermeier, E., Kanan, C., Kudithipudi, D., Lampert, C. H., Mundt, M., Pascanu, R., Popescu, A., Tolias, A. S., Weijer, J. v. d., Liu, B., Lomonaco, V., Tuytelaars, T., and Ven, G. M. v. d. (2024). Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*.

A Proofs

Proof of Proposition 1. Let $M = (Q, \Sigma', \Gamma, \delta, q_0, \square', F)$ be a Turing machine where:

- Q is the finite set of states
- Σ' is the input alphabet
- Γ is the tape alphabet with $\Sigma' \subset \Gamma$
- $\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$ is the initial state
- $\square' \in \Gamma$ is the blank symbol
- $F \subseteq Q$ is the set of final states

Our proof constructs an algorithmic Markov process, $\mathcal{E} = (\Omega, \Lambda, \mathbb{T})$. Specifically, it involves showing how the Turing machine can be represented on a state-space Ω , with its transition function represented in the Markov transition function, \mathbb{T} .

Constructing the state-space (Ω)

Let $\Lambda = \mathbb{Z}$ denote the integers for tape positions, $\Sigma = ((Q \cup \{\square''\}) \times \Gamma) \cup \{\square\}$ where \square is the Markov blank symbol. Then each $\omega \in \Omega$ represents a complete Turing machine configuration by encoding the tape contents at each position, the current state and head position.

Specifically, for a TM configuration with state q, head at position h, and tape contents ... $a_{-1}a_0a_1...$, we define:

$$\omega(\lambda) = \begin{cases} (q, a_h) & \text{if } \lambda = h \text{ (head position)} \\ (\square'', a_\lambda) & \text{if } \lambda \neq h \text{ and } a_\lambda \neq \square' \\ \square & \text{otherwise} \end{cases}$$

Constructing the transition function (\mathbb{T})

The Markov transition $\mathbb{T}(\omega)$ simulates one step of the Turing machine. It first finds the head position h where $\omega(h)=(q,a)$ for some $q\in Q, a\in \Gamma$. It then applies the Turing machine transition: $\delta(q,a)=(q',a',d)$ where $d\in\{L,R\}$. Lastly, we construct ω' by:

- Setting $\omega'(h) = (\square'', a')$ (write new symbol)
- Setting $\omega'(h + \text{offset}(d)) = (q', b)$ where

$$b = \begin{cases} c & \text{if } \omega(h + \text{offset}(d)) = (\square'', c) \text{ for some } c \in \Gamma \\ \square' & \text{if } \omega(h + \text{offset}(d)) = \square \end{cases}$$

and offset
$$(L) = -1$$
, offset $(R) = 1$

• Leaving all other positions unchanged: $\omega'(\lambda) = \omega(\lambda)$ for all $\lambda \notin \{h, h + \text{offset}(d)\}$

At each step, $|\omega'| < \infty$ because each step changes at most 2 positions, preserving finiteness. Note also that $\mathbb T$ is computable in $O(\operatorname{poly}(|\omega|))$; finding the head and updating positions requires linear scan and constant updates.

This construction preserves the computational behaviour of the Turing machine while representing it as an algorithmic Markov process.

Proof of Proposition 2. We prove each part in sequence.

Part 1: The automaton is equivalent to a k-horizon boundaried Markov process.

Given that $b^k(Z) = X \cup Y$, the k-step transition dynamics on the internal substate-space $\Omega|_Z$ depend only on the current internal substate and the boundary-space $\Omega|_{X \cup Y}$.

We define twok-horizon boundaried Markov process. First, for the substate, we have: $\mathcal{E}|_Z = (\Omega|_Z, \Omega|_{X \cup Y}, \mathbb{T}|_Z^k)$ where $\mathbb{T}|_Z^k(z, (x, y)) := u(z, x, y)$. Next, for the policy, we have: $\mathcal{E}|_Y = (\Omega|_Y, \Omega|_{Z \cup X}, \mathbb{T}|_Y^k)$ where $\mathbb{T}|_Y^k(z, (x, y)) := \pi(z, x, y)$.

That is, both the policy, π , and the internal substate update function, u, is simulated by the k-step composition of the environment's transition function. By the definition of uniform locality and the condition $b^k(Z) = X \cup Y$, the k-step transition on $\Omega|_Z$ is fully determined by the current substate $z \in \Omega|_Z$ and the boundary $(x,y) \in \Omega|_{X \cup Y}$. This establishes the equivalence.

Part 2: The automaton's environment is a partially observable Markov decision process.

From the automaton's perspective, the environment consists of:

- State space: The complement of the automaton's substate-space, $\Omega \setminus \Omega|_Z$
- Action space: The output space $\Omega|_Y$
- Observation space: The input space $\Omega|_X$
- Transition function: Given the current environment state and the automaton's action $y \in \Omega|_Y$, the next environment state follows from $\mathbb T$
- Observation function: The automaton observes $x \in \Omega|_X$ from the current environment state

Since the automaton only observes $\Omega|_X$ and not the full environment state $\Omega\setminus\Omega|_Z$, this constitutes partial observability. The Markov property holds for the underlying environment state transitions via \mathbb{T}

Part 3: The automaton's interaction is equivalent to a stateful policy acting on the environment.

The automaton maintains an internal state $z \in \Omega|_Z$ and produces actions via the output function $\pi: \Omega|_Z \times \Omega|_X \times \Omega|_Y \to \Omega|_Y$. This defines a stateful policy:

$$\pi_{\text{stateful}}(x, z) = \pi(z, x, y)$$

where the internal state z is updated according to:

$$z_{t+1} = u(z_t, x_t, y_t)$$

This is precisely the definition of a stateful policy in partially observable environments, where the policy maintains internal memory (the substate z) and conditions its actions on both observations and this internal state.

Proof of Proposition 3. We first show that an embedded automaton is only capable of a limited form of computation relative to the partially observable Markov decision process. We then outline additional constraints that result from the automaton being embedded.

An embedded automaton is equivalent to a finite-state machine. This means that the automaton is only capable of recognizing a regular language. The partially observable Markov decision process that it faces, however, is a function of an unbounded substate-space of the computationally universal environment. This means that it can, in general, generate a recursively-enumerable language. The embedded automaton is thus implicitly computationally constrained, because of the separation between automaton and the environment in the Chomsky hierarchy (Chomsky, 1959).

Moreover, for a sufficiently long behaviour sequence, the automaton must eventually return to a previous state (i.e., T >> |Z|). Thus, any behavioural sequence with a period longer than |Z| cannot be represented by the automaton.

Thus, an embedded automaton simulated in a universal-local environment is implicitly constrained: there exist input-output behaviours that it cannot realize.

There are two additional ways in which an embedded automaton is implicitly constrained:

- 1. **Minimum size**: The size of an embedded automaton, including the size of its input and output spaces, cannot be arbitrarily small, and thus there exists a minimum size. This implies that the automaton cannot read and write to arbitrarily small parts of the environment, constraining its observation and action spaces.
- Simulation time: Simulating an embedded automaton in a universal-local environment may
 also incur a simulation overhead. This constrains the automaton by the fact that several
 transitions in the environment may be necessary to simulate a single transition for the
 automaton.

While the embedded automaton is computationally constrained relative to its environment, these two additional constraints limit the information made available to the automaton about the environment. Specifically, an automaton generally cannot observe, process and output information at the same granularity, or at the same timescale, as the environment because of constraints on its size and its simulation time.

Proof of Theorem 1. We denote the maximum achievable T-horizon interactivity for a given automaton as,

$$\max_{\mathcal{A}} \mathbb{I}_T(\mathcal{A}|x_t, b_{0:t-1}).$$

All inequalities below pertaining to Kolmogorov complexity are subadditive, meaning they hide constant terms, O(1).

Intuition: interactivity depends on a substate-space that grows with the horizon, due to locality. Specifically, the future behaviour of an embedded automaton is determined by (i) the universal-local environment's transition function, (ii) the embedded automaton's initial internal state, and (iii) a substate-space of the environment that grows with the horizon of behaviour considered. If an embedded automaton's past behaviour were predictive of its future behaviour of a given horizon, then it would also imply that its past behaviour is predictive of a substate-space growing with that horizon. For a large enough horizon, the size of this substate-space will eventually be larger than the capacity of the automaton. An embedded automaton with a given capacity cannot maximize interactivity beyond a given horizon, meaning that it actively faces an implicit capacity constraint.

Formally: an automaton with sufficiently high interactivity will produce a behaviour with high unconditional Kolmogorov complexity, $\mathbb{K}(b_{t:t+T-1}) > C(\mathcal{A})$. Because the behaviour is generated by the automaton, we know that the Kolmogorov complexity is also upper bounded by the capacity, $C(\mathcal{A}) + c > \mathbb{K}(b_{t:t+T-1})$, where c is a constant independent of \mathcal{A} .

Thus, we can upper bounded interactivity in terms of the unconditional complexity,

$$\mathbb{I}_{T}(\mathcal{A}|x_{t}, b_{0:t-1}) = (\mathbb{K}(b_{t:t+T-1}) - \mathbb{K}(b_{t:t+T-1} | b_{0:t-1})) \\
\leq \mathbb{K}(b_{t:t+T-1})$$

Which uses the fact that the conditional algorithmic complexity is positive, $\mathbb{K}(b_{t:t+T-1} \mid b_{0:t-1}) > 0$.

Next we use the fact that the Kolmogorov complexity of a sequence produced by an automaton is upper bounded by its capacity, which implies that,

$$\max_{\mathcal{A}} \mathbb{I}_T(\mathcal{A}|x_t, b_{0:t-1}) \le \mathbb{K}(b_{t:t+T-1}) \le C(\mathcal{A})$$

Proof of Theorem 2. We provide a proof for each of the two desiderata

(i) The first property follows from an argument that is similar to Theorem 1, but adapted to a learning agent, A, with some bounded capacity C(A). We are interested in what interactivity the best such agent can achieve, $\max_{A} \mathbb{I}_T(A|x_t, b_{0:t-1})$.

A bounded agent that maximizes its interactivity will have a non-zero unconditional agent-relativized complexity, $\mathbb{K}_{\mathcal{A}}(b_{t:t+T-1}|\epsilon) > 0$ (otherwise, its interactivity would be zero). This implies that the unconditional Kolmogorov complexity of its behaviour is on the order of the capacity of the agent, $\mathbb{K}(b_{t:H}) \geq C(\mathcal{A}) - O(1)$, where O(1) is a constant independent of the agent. Because the behaviour

is generated by the automaton, we know that the Kolmogorov complexity is also upper bounded in terms of the capacity, $C(A) + O(1) \ge \mathbb{K}(b_{t:H})$.

Such an agent will also have low conditional agent-relativized complexity (otherwise, its interactivity would be low). An optimal learning agent that minimizes the agent-relativized complexity, $\mathbb{K}_{\mathcal{A}}(b_{t:t+T-1}|b_{0:t-1})=0$, has conditional Kolmogorov complexity is strictly less than the capacity of the agent, $\mathbb{K}(b_{t:t+T-1}|b_{0:t-1})< C(\mathcal{A})$. In fact, we have, for $\alpha<1$, that $\mathbb{K}(b_{t:t+T-1}|b_{0:t-1})\leq \alpha C(\mathcal{A})$. This is because the agent can only use a fraction of its capacity on predicting its future behaviour (in addition to making predictions, an agent selects actions, and updates its substate).

Taken together, we have that the performance of an interactivity-seeking agent interactivity is upper and lower bounded by capacity,

$$(1 - \alpha)C(\mathcal{A}) - O(1) \le \max_{\mathcal{A}} \mathbb{I}_T(\mathcal{A}|x_t, b_{0:t-1}) \le C(\mathcal{A}) + O(1).$$

An agent with a given capacity cannot maximize its interactivity without increasing its capacity. Thus, a bounded agent that seeks to maximize its interactivity through learning is limited by its finite capacity constraint.

(ii) For the second property, we demonstrate the necessity of continual adaptation for maximizing interactivity, by considering the role of the embedded agent's transition function.

Continual Adaptation in Automata

First we consider the finite-state automaton, \mathcal{A} , and how its substate transition function, u, encodes its learning. An automaton agent that has stopped learning is thus equivalent to one that stops updating its internal state. In this case, the automaton's internal state remains constant $z_{t'}=z$ for all t'>t. A finite-state automaton has a capacity on the order of $C(\mathcal{A})\in O(poly(|A|))$. But, a finite-state automaton that does not update its internal state, denoted by \mathcal{A}^- , has a reduction in its capacity. In particular, the capacity is reduced to $C(\mathcal{A}^-)=O(|\Omega|_X|)$, because the terms needed to encode the transition function are no longer needed for an automaton that does not use the transition function. Using the upper bounds on interactivity from the Theorem 1, we conclude that an agent that stops learning reduces its future output complexity from $O(|I_A||A|\log|A|)$ to $O(|I_A|)$. Thus, it is suboptimal to stop learning.

Continual Adaptation in Reinforcement Learning Agents

Value parameters: If the parameters of the value function stop being updated, the interactivity objective immediately collapses to 0.

Policy parameters: Suppose the policy's parameters stop being updated, meaning that the policy becomes fixed. Then the sequence of actions taken by the fixed policy becomes a Markov process, which is predictable. Under the Markovian dynamics induced by a fixed policy, the value function could converge to an optimal static prediction of the future behaviour. This would lower both the unconditional Kolmogorov complexity and interactivity. Thus, an agent maximizing interactivity is suboptimal if it stops updating its policy parameters.

B Experimental Details for Behavioural Self-Prediction

The code for the experiments can be found at: https://github.com/AlexLewandowski/bigger-world-interactivity

The problem that we consider involves a value function predicting a policy's actions, where the policy is trained to maximize interactivity. Specifically, the policy is adversarial to the static value function but cooperative with the dynamic value function. We consider a policy that maps its previous action to a new action. In this case, there is no external environment providing observations.

The value function, which we restrict to be linear, is tasked with predicting the future behaviour of the policy iteratively and online. At the first timestep, we randomly initialize the parameters of the policy $\theta_0 \sim p(\theta)$ and of the value function, $\mathbf{W}_0 \sim p(\mathbf{W})$, using standard distributions for neural network initialization. We then randomly sample the initial action, $b_0 \sim N(0, 1/d)$, where d=1000 denotes the dimensionality of the action. The function approximator, π_θ is then trained to maximize

its interactivity whereas the value function $v_{\mathbf{W}}$ is trained using TD(0). Specifically, the following steps are repeated at each timestep t,

- A trajectory of T actions is produced by the policy, π_{θ_t} .
- A copy of the current value function is frozen, and we record the temporal difference errors incurred along this trajectory: $\{\delta_{t,\gamma}(z_t,\theta_t),\ldots,\delta_{t+T,\gamma}(z_t,\theta_t)\}$. The sum of squared temporal difference errors provides a measure of the unconditional algorithmic complexity, $\mathbb{K}_{\mathcal{A}}(b_{t:T}|\epsilon) = \sum_{k=1}^{T} \delta_{t+k,\gamma}^2(z_t,\theta_t)$.
- A copy of the current value function is updated dynamically along the trajectory, and we record the temporal difference errors incurred along this trajectory: $\{\delta_{t,\gamma}(z_t,\theta_t),\ldots,\delta_{t+T,\gamma}(z_{t+T},\theta)\}$. Where each z_{t+k} corresponds to the updated parameters of the value function. The sum of squared temporal difference errors provides a measure of the conditional algorithmic complexity, $\mathbb{K}_{\mathcal{A}}(b_{t:t+T-1}|b_{0:t-1}) = \sum_{k=1}^T \delta_{t+k,\gamma}^2(z_{t+k},\theta_t)$.
- We update the policy using a single step from a gradient-based optimizer on the loss,

$$J(\theta) = \sum_{k=1}^{T} \delta_{t+k,\gamma}^{2}(z_{t}, \theta) - \delta_{t+k,\gamma}^{2}(z_{t+k}, \theta).$$

- We take an action using the updated policy, $b_{t+1} = \pi(b_t; \theta_{t+1})$.
- We now update the value function with a single step of TD(0) using the temporal difference error, $\delta_t = b_{t+1} + \gamma v(b_{t+1}, \mathbf{W}_t) v(b_t, \mathbf{W}_t)$.

B.1 Interpreting Interactivity Maximization As A Continual Learning Benchmark

Our theoretical and experimental results show that maximizing interactivity requires both fast adaptation to increase complexity with higher prediction errors and stability to sustain adaptation over time. That is, maximizing interactivity involves the canonical plasticity-stability trade-off of continual learning (Grossberg and Grossberg, 1982; Parisi et al., 2019). Our empirical results demonstrate that deep nonlinear baselines fail at striking this balance, whereas deep linear networks appear to naturally achieve this balance. This suggests that this synthetic benchmark isolates the key challenge in continual learning, while also not requiring outside data or environments. This is significant because few environments are designed specifically to evaluate continual adaptation. Thus, it is suboptimal to stop learning in this setting, regardless of the capacity of the algorithm or function approximator.

B.2 Limitations of Experiments

Our experiments used relatively shallow networks, with a maximum depth of D=4. However, with the meta-gradient calculation over a finite horizon of T=10, the effective depth of the networks during auto-differentiation is $T\cdot D=40$. Meta-gradient methods for deep networks can exhibit more pathological learning dynamics due to increased curvature, leading to instability that could partially explain the discrepancy between linear and nonlinear networks. Understanding how to control curvature using only first-order methods is key for effective meta-gradient descent in this setting.

The meta-gradient method poses several limitations in scaling. Ideally, we would prefer to scale the horizon and the capacity of the function approximator. However, because meta-gradients involve a second-order terms, involving the hessian, and because the horizon is multiplicative with the depth of the network, we have a computational complexity on the order $O(HD^2)$, where D is the depth of the network. Scaling both the horizon and the capacity results in a effective cubic scaling. A more promising direction involves bootstrapping meta-gradients (Flennerhag et al., 2022), and other first-order approximation (Nichol et al., 2018).

Experimental evaluation in this setting also requires special consideration. Holding the agent fixed for evaluation, as is commonly done in machine learning, is not appropriate given that interactivity is defined as an online objective. In addition, standard approaches to hyperparameter tuning may not be feasible for evaluating the long-term performance of a continual learning agent (Mesbahi et al., 2025). Overcoming these obstacles to more fairly assess dependence on hyperparameters would require re-evaluation of several components of empirical practice in machine learning.

C Additional Background and Related Work

C.1 Algorithmic complexity

The Kolmogorov complexity (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966) of an object (encoded as a binary string) is the length of the shortest program that computes it and halts. Unlike traditional information theory, it measures the complexity of an individual object without depending on a stochastic source or ensemble.

The Kolmogorov complexity of a string depends on the choice of a universal Turing machine. However, since any universal Turing machine can simulate another (e.g., via a compiler), the choice of the machine affects the Kolmogorov complexity by, at most, an additive constant independent of the specific string (Li and Vitányi, 2019).

Kolmogorov complexity is closely tied to compression, where the shortest description represents the most efficient compression for the given universal Turing machine. Although Kolmogorov complexity is uncomputable, it is possible to compute improving upper bounds by searching over all possible programs in parallel and tracking the shortest candidate that generates the target string (Li and Vitányi, 2019).

C.2 AIXI

AIXI defines a general Bayes-optimal reinforcement learning agent in an unknown computable environment (Hutter, 2005). In this framework, the environment is represented by a Turing machine with unidirectional input and output tapes, and bidirectional working/internal tapes. The agent's actions are received by the environment on its input tape, based on which it can write a computable history-based reward and observation on its output tape.

The AIXI agent acts in a Bayes-optimal manner by planning based on a posterior estimate over all computable environments, using Solomonoff's universal prior as a starting point (Solomonoff, 1964). This prior assigns higher probability to 'simpler' environments—those with lower Kolmogorov complexity. However, both Solomonoff's prior and AIXI are incomputable, making the development of practical approximations within this framework a key area of interest (Veness et al., 2011).

C.3 Connections to intrinsic motivation and the free energy principle

Previous work has explored several intrinsic drives that can guide agent behaviour without the need for explicit external rewards (Schmidhuber, 2010; Barto, 2013). Many approaches to intrinsic motivation are developed within the framework of traditional RL, where the agents are not constrained relative to the environment. As a result, these approaches may not be well-suited to a big world. Nevertheless, interactivity shares connections to ideas such as mutual information maximization in intrinsic motivation.

The information gain of a dynamics model can serve as an intrinsic or auxiliary reward, promoting curious exploration (Storck et al., 1995; Houthooft et al., 2016) Unlike curiosity driven by information gain, the goal of interactivity is not to learn an accurate model of the world.

Another related concept is Empowerment (Klyubin et al., 2005), where an agent seeks to maximize its control over its environment. Empowerment-seeking agents aim to maximize the mutual information between their actions and future states. Such agents avoid states where their actions have low influence and prefer states that allow for a wide range of controllable outcomes. This objective can also be used to learn a set of behaviours (or options) that lead to different final states (Mohamed and Jimenez Rezende, 2015; Gregor et al., 2016). As discussed earlier, interactivity-maximizing agents produce complex yet predictable behaviour, which is not directly tied to the concept of control. Furthermore, unlike objectives grounded in traditional (Shannon) information theory, interactivity relies on asymmetric algorithmic mutual information between previous inputs and future outputs.

Active inference describes agentic behavior in partially observable environments as the minimization of free energy (Friston et al., 2010; Sajid et al., 2021). Free-energy minimization prefers selecting actions that lead to highly predictable states—inputs that are unsurprising to the agent's model. In contrast to free-energy minimization, maximizing interactivity actively discourages low-complexity predictable states.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]
Justification:
Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the
 contributions made in the paper and important assumptions and limitations. A No or
 NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: In the discussion, the limitations of an efficient algorithm for maximizing ineractivity is discussed.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes, in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes, in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
- (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
- (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
- (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: Not currently in the supplementary material, but we plan to open source it. Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be
 possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not
 including code, unless this is central to the contribution (e.g., for a new open-source
 benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: In the appendix and supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Shaded error bars are provided, but no statistical p-test.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error
 of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [No]

Justification: No, only a local cluster was used which did not log computational resources, but they can be replicated on a modern GPU in less than 24 hours.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]
Justification:
Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a
 deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: It does not, as it is purely a theoretical investigation.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: None of these were used.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do
 not require this, but we encourage authors to take this into account and make a best
 faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: Not used.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

 If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]
Justification:
Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]
Justification:
Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]
Justification:
Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent)
 may be required for any human subjects research. If you obtained IRB approval, you
 should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [No]
Justification:
Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.