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

Anonymous authors
Paper under double-blind review

Abstract

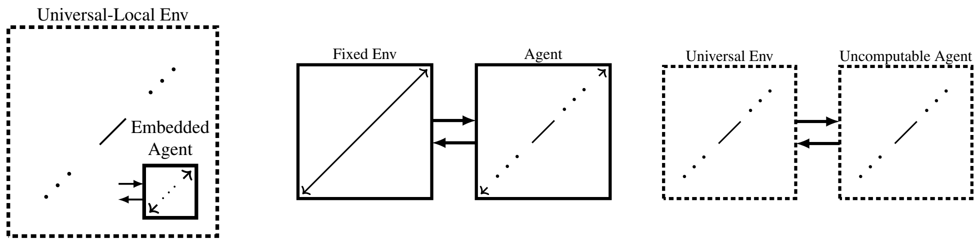
Continual learning is often motivated by the idea, known as the big world hypothesis, that the "world is bigger" than the agent. Recent problem formulations capture this idea by explicitly constraining an agent relative to the environment. These constraints lead to solutions in which the agent continually 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 reward-free and partially-observable Markov decision process over a countably infinite state-space. We propose an objective for this setting, which we call *interactivity*, that measures an agent's ability to continually adapt its behaviour to learn new predictions. To support experimentation on continual adaptation, we develop a synthetic benchmark in which an interactivity-seeking agent constructs its own non-stationary stream of experience from which it must continually learn to predict.

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 & 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 considered, but these are difficult to incorporate. Such constraints include limits on the agent's compute (see discussion on measuring compute in Section 4.1, Verwimp et al., 2024) and on the energy used by the agent's hardware (Javed & Sutton, 2024). Information theory provides a framework to formalize explicit agent constraints (Kumar et al., 2023; 2024). However, outside

⋆Correspondence to: Alex Lewandowski <lewandowski@ualberta.ca>.



(a) This work (b) Traditional RL (c) AIXI

Figure 1: **Comparing the agent's relationship to the environment in our work, traditional RL, and AIXI.** This work considers a universal-local environment (defined in Section 3), in which agents of varying sizes are embedded and implicitly constrained (defined in Section 4). Traditional RL involves a fixed environment and agents of varying size, where the agent is often unconstrained by being "bigger" than the considered environment. AIXI involves a computationally universal environment and an uncomputable agent, both of which are unconstrained.

of simple and well-specified pairs of agent and environment, these constraints can be difficult to characterize without knowledge of the true information-theoretic quantities involved between the state maintained by the agent and its future sensory stream from the environment. This framework also does not prescribe new algorithms that improve an agent's capability for using its limited capacity. In addition, explicit constraints hinder the effectiveness of scaling up the agent's capacity, which has been a 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). The embedded aspect of all intelligent systems, by existing in the physical world, is not often considered to be part of the problem formulation (Demski & Garrabrant, 2019). However, the physical world is a clear example of a world bigger than any agent, suggesting that embedded agency may be useful in formulating the big world hypothesis.

To provide a general environment in which an agent can be embedded, we define a *universal-local environment*. This environment is a Markov process that is computationally universal—capable of simulating any computation—where the transition dynamics can be localized to a neighbourhood of the state-space. Our approach is similar to universal artificial intelligence (Hutter, 2005), which considers a computationally universal environment to explore the limits of the theoretically optimal, but uncomputable, AIXI agent (Hutter, 2000). The AIXI agent was also extended to an embedded agent, simulated within the computationally universal environment, providing an uncomputable definition of a theoretically optimal capacity-bounded agent (Orseau & Ring, 2012). Our approach similarly considers embedding an agent in a computationally universal environment, but with the added restriction that the environment's transition dynamics are local. In particular, our departure is aimed towards capturing the big world hypothesis and avoiding the limitations of explicit agent constraints, while also being amenable to computable approximation.

To define an embedded agent, we consider an embedded automaton simulated within the state-space of our universal-local environment. This automaton interacts with a partially observable Markov decision process, defined on the boundary between the automaton and the rest of the universal-local environment. We then propose *interactivity* that measures an embedded automaton's ability to adapt its future behaviour, conditioned on its past behaviour, using Kolmogorov complexity. An agent's interactivity is a computational measure of its adaptivity, and as such is always upper bounded by the capacity of the agent. Interactivity is similar to previously considered intrinsic motivation objectives (Chentanez et al., 2004; Schmidhuber, 2010), and specifically predictive information (Bialek et al., 2001; Still & Precup, 2012). However, interactivity differs because of its formulation in terms of behaviours using Kolmogorov complexity. This makes interactivity better suited to sequential decision making in the constrained and partially observable setting that we consider.

We also develop a reinforcement learning algorithm to maximize interactivity by recasting Kolmogorov complexity in terms of the prediction error incurred by the agent. Interactivity can be viewed from this perspective is derived from a value function in the undiscounted setting that predicts the agent's average future behaviour. Maximizing interactivity involves learning a policy to direct the agent's future behaviour to new experiences from which it continually learns. We show that maximizing interactivity leads to the common desideratum of the continual learning problem, in which any agent that stops learning is suboptimal. Finally, we develop a synthetic benchmark to support experimentation on continual adaptation.

2 Background

The environment that we consider uses computational universality—the capability of performing arbitrary computations—to embed an agent. In particular, we make use of the Church-Turing thesis, which implies that all computationally universal systems are equivalent in their capabilities (Church, 1936; Turing, 1937). This allows us to define a general environment without reference to any specific computationally universal system (*e.g.*, a universal Turing machine). The Church-Turing thesis also implies that any computationally universal system can simulate any other computational system. We will use this to define the agent as an automaton performing a computation simulated within the environment.

To understand the capabilities of such an agent, we will consider the properties of its input/output behaviour. In particular, we will use the Kolmogorov complexity of a string, which is the length of the shortest program that computes it and halts (Kolmogorov, 1965; Solomonoff, 1964; Chaitin, 1966). An automaton is a bounded computation, and thus it can only produce output strings with bounded Kolmogorov complexity given its finite capacity.

Definition 1. *The Kolmogorov complexity of a string x, conditioned on another string y, is the length of the shortest program, $|c|$, that outputs x given y as input, $\mathbb{K}(x|y) = \min\{|c| : \mathcal{U}(c, y) = x\}$, where $\mathcal{U}$ is a reference universal Turing machine. The unconditional Kolmogorov complexity sets y to be the empty string, denoted ϵ.*

While Kolmogorov complexity depends on the choice of a universal Turing machine, any specific choice affects the Kolmogorov complexity by, at most, an additive constant independent of the specific string (Li & Vitányi, 2019). This is because, by the Church-Turing thesis, any universal Turing machine can simulate another (*e.g.*, via a compiler).

3 A Universal-Local Environment

We begin by defining a general notion of environment in which an agent can be embedded. Specifically, we consider an environment that is capable of simulating arbitrary computations on its state-space (Section 3.1), such that any bounded computation can be localized to a portion of the environment's state-space (Section 3.2). These two properties will be used in Section 4 to define an automaton on the state-space of this environment. Such an automaton will be used to represent an agent, ensuring that it is always implicitly constrained in its computational capacity relative to its environment.

3.1 Markov Representation of a Computationally Universal Environment

We use *environment* to refer to a general history-based process that is defined over a finite set of symbols, and without an explicit notion of agent.

Definition 2. *An environment, $\mathcal{E} = (\Sigma, \mathbb{C})$, is a discrete process defined over a finite symbol-set, Σ, that maps a string of symbols, $\sigma_{0:t-1} = \sigma_0\sigma_1 \cdots \sigma_{t-1}$, to the next-symbol that extends the string, $\sigma_t \in \Sigma$, using the construction function, $\sigma_t = \mathbb{C}(\sigma_{0:t-1})$.*

An environment is computationally universal if it is equivalent to a universal Turing machine, meaning that it is capable of simulating any computation given a suitable initial string of symbols. Such an environment can also be represented as a Markov process on a countably infinite state-space.

Proposition 1 (Universal Markov Environment)**.** *There exists a Markov representation of a computationally universal environment, $\mathcal{M}(\mathcal{E}) = (\Omega, \mathbb{U})$, defined over the countably infinite state-space, Ω, in which the state, $\omega_t \in \Omega$, is updated using the transition function, $\omega_{t+1} = \mathbb{U}(\omega_t)$.*

All proofs of propositions and theorems can be found in Section A of the Appendix.

We emphasize that, despite using a Markov representation, the universal Markov environment is more general than the Markov environments typically considered in reinforcement learning. In particular, a universal Markov environment is capable of simulating any other computation, which will be crucial to define an embedded agent in Section 4.

3.2 Defining Locality with Boundaried Markov Processes

Intuitively, locality means that we can consider the environment's transition dynamics on a restricted portion of the state-space. Specifically, we use the term substate-space to refer to the portion of the state-space restricted to a finite index-set.

Definition 3. *A substate-space, Ω_Λ, is defined as a restriction of the state-space, Ω, to a finite index-set, $\texttt{Idx}(\Omega_\Lambda) := \Lambda$ where $|\Lambda| < \infty$, such that $\Omega_\Lambda = \{\omega_\Lambda : \omega \in \Omega\}$ where $\omega_\Lambda = \{\omega_i\}_{i \in \Lambda}$. We use square set notation to denote operations on the index set, such as $\Omega_\Lambda \sqsubseteq \Omega$ to denote the inclusion of the index-set, $\Lambda \subseteq \texttt{Idx}(\Omega)$, and the union of index-sets, $\Omega_{\Lambda_1} \sqcup \Omega_{\Lambda_2} = \Omega_{\Lambda_1 \cup \Lambda_2}$.*

We now consider the environment's transition dynamics restricted to a generic substate-space, $X \sqsubseteq \Omega$, without reference to the specific index-set, $\texttt{Idx}(X)$. In particular, we define a boundaried Markov process in which the one-step transition dynamics, $\mathbb{U}_X$, depend on another substate-space, $B_X \sqsubseteq \Omega$, referred to as the boundary-space for a given substate-space, X.

Definition 4. *A boundaried Markov process, $\mathcal{M}_X = (X, B_X, \mathbb{U}_X)$, is a discrete process in which the substate-space, X, and its boundary-space, B_X, together define the one-step transition dynamics of the substate-space, $x_{t+1} = \mathbb{U}_X(x_t, b_t)$, for $x_{t+1}, x_t \in X$ and $b_t \in B_X$.*

The boundary-space is defined for one-step dynamics; A larger boundary-space is generally needed for multi-step transition dynamics. This is because the current substate, $x_t \in X$, and the current boundary, $b_t \in B_X$, only define the next-substate, $x_{t+1} \in X$, and not the next-boundary, $b_{t+1} \in B_X$. We use this fact to define a local environment that consists of nested boundaried Markov processes.

Definition 5 (Locality)**.** *A universal Markov environment is local if, for any two proper substate-spaces, $W \sqsubsetneq X \sqsubseteq \Omega$, there exists boundaried Markov processes on these substate-spaces with corresponding index-sets that are properly contained, $W \sqcup B_W \sqsubsetneq X \sqcup B_X$.*

Thus, a *universal-local environment* is a universal Markov environment that is also local. This environment is capable of simulating arbitrary computations, and any bounded computation is localized to a portion of the environment's state-space. It can be understood as a computationally universal Markov process in which longer-term dynamics are a function of a larger portion of the state-space.

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, within Conway's Game of Life, a universal Turing machine can be simulated (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. The one-step transition dynamics on any substate-space depend on the adjacent neighbourhood of that substate-space, which defines the boundary-space (see Figure 2). Conway's game of life is local because if one substate-space contains another, then the boundary-spaces (the adjacent neighbourhood of the substate-spaces) are also also contained.

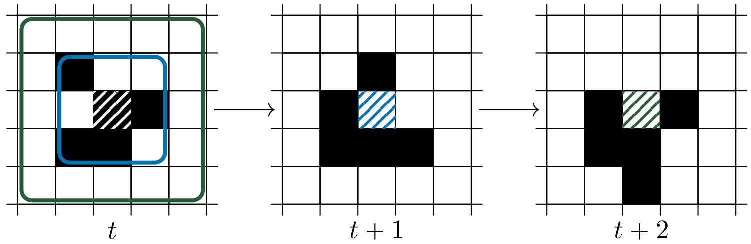


Figure 2: **Conway's Game of Life is a cellular automaton and an example of a universal-local environment.** The state-space is an infinite 2D grid, in which cells live (black) with 2 or 3 neighbours, but die (white) otherwise, and dead cells with 3 neighbours become alive. The blue and green borders (*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.

While Conway's Game of Life has the potential to simulate any computation 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 will consider and formalize the implicit constraints faced by an agent if it were embedded in such an environment.

4 A Computationally-Embedded Agent

We define an embedded agent as an automaton that is simulated within the universal-local environment. This embedded automaton is equivalent to a boundaried Markov process, with the boundary-space acting as an interface that separates the automaton from the rest of the universal-local environment (Jiang, 2019; Harutyunyan, 2020). We prove that an embedded automaton is equivalent to an agent interacting with a partially observable Markov decision process, under some conditions on this boundary-space. Using Kolmogorov complexity, we propose *interactivity* as a measure of the embedded agent's ability to adapt its future behaviour, using experience from its past behaviour. We prove that interactivity is constrained by any finite capacity and discuss the way in which interacitivty measures a general capability for continual adaptation.

4.1 Embedding an Agent as an Automaton in a Universal-Local Environment

A universal-local environment can simulate arbitrary computations, which we use to define an embedded automaton, $\mathcal{A}$, on the environment's state-space, Ω. Moreover, due to locality, the embedded automaton can be localized to a substate-space, $A \sqsubseteq \Omega$ (see Figure 3, left).

Definition 6. *An embedded automaton is defined by $\mathcal{A} = (A, I_A, O_A, \mathbb{U}_A, \pi_A)$, where $A \sqsubseteq \Omega$ is the internal substate-space of the automaton, $I_A, O_A \sqsubseteq B_A$ are input and output spaces defined on the boundary-space, B_A, and $\mathbb{U}_A, \pi_A$ are the automaton's transition and output function respectively.*

Relating this to an agent in reinforcement learning, we may think of the input-space as the observation-space,[1] the internal substate as the parameters of a function approximator, the output-space as an action-space, the transition function as a learning rule, and the output function as a policy.

Proposition 2 (Embedded Agent)**.** *An embedded automaton is equivalent to an agent interacting with a (potentially reward-free) partially observable Markov decision process, if its boundary-space consists of only the input and output spaces, $I_A \sqcup O_A = B_A$.*

Now that we have defined both the embedded agent and its partially observable Markov decision process within the same universal-local environment, we can describe their relationship.

Proposition 3 (Implicitly Constrained)**.** *Every embedded agent is implicitly constrained, relative to its partially observable Markov decision process, limiting its memory and computational capacity.*

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

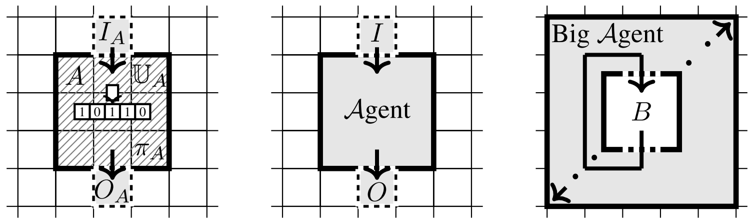


Figure 3: **An illustrative depiction of a computationally-embedded agent interacting with its environment.** An embedded automaton is simulated on the substate-space of the universal-local environment, $A \sqsubseteq \Omega$, reading from its input-space, I_A, updating its internal state with $\mathbb{U}_A$, and writing to its output-space, O_A, with its output function, π_A (*left*). A computationally-embedded agent is characterized by its input-output behaviour, with the goal of maximizing interactivity, rather than the internal specification of its computations (*middle*). We consider an idealized setting, referred to as a Big Agent, in which the agent has full control over its experience, and only observes its previous output on the boundary-space, B (*right*).

While every embedded agent is implicitly constrained, some may generate simple output sequences that do not require more than agent's capacity. For example, a periodic output sequence would not require more capacity than the period of the sequence. We will show, however, that agents are constrained by their finite capacity when adapting to their 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 an embedded agent's intrinsic ability to adapt its future behaviour, towards higher complexity, conditioned on its past behaviour. Specifically, we use Kolmogorov complexity to formalize this otherwise intuitive notion of adaptation and complexity.

Following Proposition 2, we represent an embedded agent as an embedded automaton $\mathcal{A}$ where its input and output spaces determine its boundary-space, $I_A \cup O_A = B_A$. Thus, the behaviour of the agent is determined by the values taken on the boundary-space, $b_t = (i_t, \pi_A(i_t)) \in B_A$ where $i_t \in I_A$ and $\pi_A(i_t) \in O_A$. At any time t, the behaviour can be separated into past, $b_{0:t} = b_0 b_1 \cdots b_t$ and the T-horizon future, $b_{t+1:T} = b_{t+1} b_{t+2} \cdots b_{t+T}$.

Definition 7. *An agent's interactivity at time t is the average difference in the unconditional Kolmogorov complexity of its future behaviour and the conditional Kolmogorov complexity of its future behaviour, conditioned on its past behaviour,* $\mathbb{I}_t^*(\mathcal{A}) = \lim_{T\to\infty} \frac{1}{T}\big(\mathbb{K}(b_{t+1:T}) - \mathbb{K}(b_{t+1:T}|b_{0:t})\big)$.

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

4.3 An Interactivity-Maximizing Agent Faces a Big World

The interactivity of any embedded agent is always constrained by its capacity. That is, with a given capacity, an embedded agent can only sustain a given level of interactivity. However, if the embedded agent is given more capacity, then it could use the additional capacity to increase its interactivity.

Theorem 1 (Big World)**.** *The interactivity of an embedded agent is upper bounded by its capacity.*

An interactivity-maximizing agent has an ability to continually adapt its future behaviour by using its past experience. This suggests the following interactivity thesis:

Interactivity measures a general capability for continual adaptation.

We refer to this as the interactivity thesis, rather than a hypothesis, to reflect its speculative and philosophical nature. An agent's capability for continual adaptation with low interactivity is limited because its future behaviour is either: i) simple, or ii) complex, but not predictable from its past experience. In either case, the thesis stresses the relative notion of capabilities. A simple agent could be capable of some adaptation, but its capabilities would be greater if its past experience was used to produce more complex behaviour. Moreover, an agent that produces complex behaviour could only be recognized as an adaptation if this complexity can be attributed, via prediction, to its past experience. Embracing the interactivity thesis naturally leads to a relative spectrum of possible adaptive agents.

5 Maximizing Interactivity with Reinforcement Learning

Interactivity is defined using Kolmogorov complexity, which is not computable in general. However, for an automaton, Kolmogorov complexity is computable by enumerating all programs up to the size of the automaton (Li & Vitányi, 2019). This brute-force approach would require more than the capacity of the automaton, necessitating approximation (see Theorem 1).

To approximate interactivity, we use the distortion-rate perspective on Kolmogorov complexity that considers the achievable error under a constraint (Vereshchagin & Vitányi, 2010). Specifically, we (i) impose a constraint by replacing the reference universal Turing machine with the embedded agent, and (ii) measure the error achieved by the embedded agent under a choice of loss function.

Definition 8. *The agent-relativized complexity, for a given embedded agent, $\mathcal{A}$, of a string x, given a string y, is the error of the best prediction by the agent, $\mathbb{A}_{\mathcal{A}}(x|y) = \min_a\{\sum_{i=1}^{|x|} \ell(x_i, \hat{x}_i) : \hat{x} = \mathcal{A}(a, y)\}$, where ℓ is a loss and $a \in A$ is a substate.*

Agent-relativized complexity is determined by the agent's predictions, making it to amenable to learning. If the agent-relativized complexity of a string, x, conditioned on the empty string is large, $\mathbb{A}_{\mathcal{A},\ell}(x|\epsilon) > 0$, then the agent is unable to predict x accurately and the complexity of that string is relatively high. If additional information, y, can be provided to the agent to reduce the prediction error, $\mathbb{A}_{\mathcal{A},\ell}(x|y) = 0$, then the conditional complexity of x is relatively low, and the additional information is useful to the agent's predictions.

We can now consider the interactivity relativized to an agent, $\mathcal{A}$, where we replace Kolmogorov complexity with agent-relativized complexity. Going forward, in the context of an agent, we will refer to agent-relativized complexity simply as complexity.

$$\mathbb{I}_t(\mathcal{A}) = \lim_{T\to\infty} \frac{1}{T}\big(\mathbb{A}_{\mathcal{A}}(b_{t+1:T}) - \mathbb{A}_{\mathcal{A}}(b_{t+1:T}|b_{0:t-1})\big).$$

The unconditional complexity, $\mathbb{A}_{\mathcal{A}}(b_{t+1:T})$, measures the error incurred by the agent when predicting its future behaviour, without having learned from prior experience. That is, without the current substate, a_t. Whereas conditional complexity, $\mathbb{A}_{\mathcal{A}}(b_{t+1:T}|b_{0:t-1})$, measures the error of the agents predictions given the current substate, a_t, encoding its past experience.

Using the agent-relativized perspective, we now develop a reinforcement learning algorithm for maximizing interactivity. Our reinforcement learning approach involves (i) learning a prediction of the conditional complexity via a value function, (ii) approximating the unconditional complexity by using an agent that has access to a subset of the past experience, and (iii) learning a policy to maximize the difference between these two predictions.

5.1 Learning Predictions of Conditional Agent-Relativized Complexity

Conditional complexity involves learning a prediction of the agent's behaviour, and we show how such a prediction can be learned via a value function. In particular, we consider the undiscounted setting of reinforcement learning, where the discount factor is deprecated, $\gamma = 1$, in favour of the long-term average of signals (Sutton & Barto, 2018). However, we are interested in learning the long-term average behaviour, rather than an externally provided reward signal.

Specifically, we consider an agent that produces a sequence of behaviour tuples of input and output, $b_t = (i_t, \pi(a_t, i_t))$, where at each timestep the agent also updates its internal substate $a_{t+1} = \mathbb{U}(a_t, b_t)$. We consider predictions made by this agent as part of its internal substate, rather than as an output, because the predictions do not directly interface with the environment.

Given such an agent, we are interested in learning the conditional complexity of its behaviour, $\mathbb{A}_{\mathcal{A}}(b_{t+1:T}|b_{0:t-1})$. Conditional complexity can be understood as a prediction about the long-term average behaviour of the agent. In this undiscounted setting, the long-term average behaviour is represented as the limit of the finite averages,[2] $b(\mathcal{A}) = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} b_t$.

The long-term average behaviour can be estimated with an online average. However, this online estimate does not provide a prediction about future behaviour, which is needed in the definition of conditional complexity. Instead, a differential value function can be defined recursively as, $v^{\mathcal{A}}(a_t, b_t) = b_{t+1} - b(\mathcal{A}) + v^{\mathcal{A}}(a_{t+1}, b_{t+1})$. An approximation to the differential value function, $\hat{v}(a_t, b_t)$, can be learned with temporal difference learning, where $b(\mathcal{A})$ is replaced with the online average, $\bar{b}_{t+1} = \bar{b}_t + \beta(b_t - \bar{b}_t)$, to form the temporal difference error, $\delta_t(a_t, b_t, a_{t+1}, b_{t+1}) = b_{t+1} - \bar{b}_{t+1} + \hat{v}(a_{t+1}, b_{t+1}) - \hat{v}(a_t, b_t)$. This differential value function can be interpreted as a undiscounted version of successor features (Barreto et al., 2017), that also include the future actions of the agent.

The conditional complexity is implicitly learned by learning this differential value function of future behaviour with temporal difference learning. That is, if the temporal difference error is low, then the estimate of the future behaviour is accurate. In particular, this implies that a suitable approximation to the conditional complexity can be defined with a temporal difference error loss function,

$$\mathbb{A}_{\mathcal{A}}(b_{t+1:T}|b_{0:t-1}) \approx \hat{\mathbb{A}}_{\mathcal{A}}(b_{t+1:T}|b_{0:t-1}) = \sum_{k=1}^{T} \ell\left(b_{t+k}, \hat{b}_{t+k}(b_{0:t-1})\right) = \sum_{k=1}^{T} \delta_{t+k}^2.$$

The temporal difference error is conditioned on past experience through the learned approximation to the differential value function. Moreover, this prediction approach amortizes the minimization in the definition of conditional complexity by iteratively learning the differential value function online. While this approach does not directly learn a prediction of the future temporal difference errors, the finite-horizon temporal difference errors provides an approximation.

5.2 Semi-Conditional Predictions of Unconditional Agent-Relativized Complexity

Maximizing interactivity also requires an approximation of the unconditional complexity. Without it, maximizing interactivity would reduce to minimizing the conditional complexity, which would simply minimize the temporal difference errors and learn the differential value function. However, it is not clear how the unconditional complexity could be learned because, by definition, it is not conditioned on any previous experience.

Rather than learning a completely unconditional complexity, we instead consider the behaviour of agent if it had not learned on a particular finite horizon, denoted by H. That is, we approximate the unconditional complexity with a semi-conditional complexity,

$$\hat{\mathbb{A}}_H(b_{t+1:T}) := \mathbb{A}(b_{t+1:T}|b_{0:t-H}) \leqslant \mathbb{A}(b_{t+1:T}).$$

Where the inequality follows, up to subadditive factors, because conditioning decreases complexity (Grunwald & Vitányi, 2004). In addition, the lower bound means that this approximation is also effective for approximately maximizing interactivity.

[2]The computational agents that we consider are determinsitic, meaning that we can drop the expected values in the following definitions.

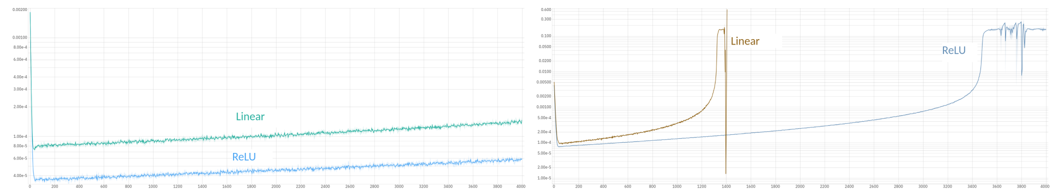


Figure 4: **Maximizing Interactivity in Behavioural Self-Prediction** With a step-size of 0.05, both experience an initial drop in interactivity and slowly improve over time (*left*). With a step-size of 0.1, both networks surpass their initial interactivity, with the linear network diverging (*right*).

5.3 Maximizing Interactivity as a Continual Learning Problem

Maximizing interactivity involves producing future behaviour that maximizes the difference between unconditional and conditional complexity. This can be accomplished by optimizing the approximations provided by semi-conditional and conditional complexity, under a horizon, H,

$$\mathbb{I}_t^H(\mathcal{A}) = \frac{1}{H}\left(\hat{\mathbb{A}}(b_{t:t+H}|b_{0:t-H}) - \hat{\mathbb{A}}(b_{t:t+H}|b_{0:t})\right) \qquad (1)$$

$$\frac{1}{H}\left(\sum_{k=1}^{H} \delta^2_{t+k}(a_{t-H}, b_{t+k}, a_{t-H}, b_{t+k+1}) - \sum_{k=1}^{H} \delta^2_{t+k}(a_{t+k}, b_{t+k}, a_{t+k+1}, b_{t+k+1})\right) \qquad (2)$$

Where the semi-conditional complexity maintains a fixed agent substate, a_{t-H}. In practice, this requires bi-level optimization to account for the agent's substate changing in the agent's conditional complexity. This can be handled by auto-differentiation such as by MAML (Finn et al., 2017), but with an online update rather than to initialization that is more similar to cross-prop (Veeriah et al., 2017). Maximizing this lower bound on interactivity is thus possible with gradient-based learning, and the approximation approaches the agent-relativized interactivity in the horizon limit, $H \to \infty$, where we treat a_z and $b_{0:z}$ as empty for $z < 0$.

As the agent's capacity increases, so too does its maximum possible interactivity. While all the agents we consider are bounded by finite capacity, if we consider the agent's infinite capacity limit as computationally universal, then maximizing interactivity becomes uncomputable. In Section 4.3, we proved that the interactivity of an embedded agent is upper bounded by its capacity. We now show a similar result for an interactivity-maximizing reinforcement learning agent.

Theorem 2. *Any bounded agent that seeks to maximize its interactivity through learning is i) limited by its finite capacity constraint and, ii) suboptimal if it stops learning.*

The desiderata of Theorem 2 were previously described as conditions for a big world simulator (Kumar et al., 2024). This demonstrates that maximizing interactivity is well-characterized by the big world hypothesis. Thus, maximizing interactivity appears to be a general problem setting in which the best use of an agent's limited resources is to continually adapt.

6 Evaluating Continual Adaptation With Behavioural Self-Prediction

Behavioural self-prediction provides a synthetic benchmark in which the agent creates its own non-stationary stream of experience, from which it must continually learn. A learning algorithm predicting its own future learning behaviour faces an implicit constraint, because it cannot observe its entire parameter set, or accurately predict what it will learn and output in the future. An illustrative depiction of the problem setting is given in Figure 3 (right), in which an agent has full control over its experience stream. The advantage of this approach is that it does not require an external environment, or any collected data. Instead, it directly evaluates the learning algorithms capabilities for learning from, and adapting to, the experience that it produces online. In particular, any learning algorithms that stops learning achieves the lowest possible performance.

We trained a two-layer network using either `Linear` or `ReLU` activations on the H-horizon approximation to interactivity, $H = 10$. We used conventional stochastic gradient descent which is generally more stable than adaptive methods (Finn et al., 2017), and used the same step-size for inner-learning of average future behaviour and for outer-learning of interactivity-maximizing behaviour. While this network is relatively shallow, the meta-gradient calculation for maximizing interactivity makes the effective network depth $2H = 20$ layers. Our findings indicate that linear methods are initially capable of fast adaptation, but that this always lead to performance collapse (Figure 4). This is surprising because linear methods are known to be stable baselines in conventional continual learning scenarios (Lewandowski et al., 2025; Dohare et al., 2024). This suggests that continual adaptation requires balancing fast adaptation with stability. Please see Appendix B for additional details.

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 contributions include: (i) characterizing the implicit constrains faced by an embedded agent, (ii) proposing interactivity as a computational measure of adaptivity, and (iii) developing a reinforcement learning algorithm for maximizing interactivity. Our work shows that maximizing interactivity leads to the common desideratum of the continual learning problem in which any agent that stops learning is suboptimal.

The key to Theorems 1 and 2 is the fact that interactivity does not depend on external feedback, but rather is defined in terms of the past and future behaviour of the agent. This is a departure from dogmas common to reinforcement learning (Abel et al., 2024). While interactivity could potentially provide a rich source of intrinsic feedback, it also introduces challenges the stability of our algorithms combining nonlinear representations, temporal difference learning, and online learning.

Maximizing interactivity provides a problem setting for studying continual learning in isolation. A promising direction is the development of an efficient algorithms for maximizing interactivity, one which bypasses costly meta-gradients and directly approximates agent-relativized complexity. Experimental evaluation in this setting also requires special consideration. Holding the agent fixed for evaluation, as is commonly done in machine learning, is not be 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., 2024). Overcoming these obstacles would require re-evaluation of several components of empirical practice in machine learning, and we thus leave an empirical investigation for future work.

We close with the following conjecture regarding interactivity and its utility as a general objective in an arbitrary environment: if an agent is capable of sustaining a particular level of interactivity, then it is also capable of behaviours that achieve other goals in that environment—such as maximizing external reward—that require equal or less interactivity.

References

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 2023.

David Abel, Mark K Ho, and Anna Harutyunyan. Three dogmas of reinforcement learning. *Reinforcement Learning Journal*, 1, 2024.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in Neural Information Processing Systems*, 2017.

Andrew G. Barto. Intrinsic motivation and reinforcement learning. *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47, 2013.

Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays, Vol. 2*. Academic Press, 1982.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.

Nuttapong Chentanez, Andrew G. Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in Neural Information Processing Systems*, 2004.

Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

Alonzo Church. An unsolvable problem of elementary number theory. *American journal of mathematics*, 58(2):345–363, 1936.

John H. Conway. The game of life. *Scientific American*, 223(4):4, 1970.

Abram Demski and Scott Garrabrant. Embedded agency. *CoRR*, abs/1902.09469v3, 2019.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632:768–774, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022.

Karl J Friston, Jean Daunizeau, James Kilner, and Stefan J Kiebel. Action and behavior: a free-energy formulation. *Biological cybernetics*, 102:227–260, 2010.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Stephen Grossberg and Stephen Grossberg. How does a brain build a cognitive code? *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, pp. 1–52, 1982.

Peter Grunwald and Paul Vitányi. Shannon information and Kolmogorov complexity. *CoRR*, abs/cs/0410002, 2004.

Anna Harutyunyan. What is an agent?, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Jordan Hoffmann et al. Training compute-optimal large language models. *CoRR*, abs/2203.15556v1, 2022.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Marcus Hutter. A theory of universal artificial intelligence based on algorithmic complexity. *arXiv preprint cs/0004001*, 2000.

Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005. ISBN 3-540-22139-5.

Khurram Javed and Richard S. Sutton. The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame Workshop at Reinforcement Learning Conference*, 2024.

Nan Jiang. On value functions and the agent-environment boundary. *CoRR*, abs/1905.13341v3, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pp. 128–135. IEEE, 2005.

A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–11, 1965.

Saurabh Kumar, Henrik Marklund, Ashish Rao, Yifan Zhu, Hong Jun Jeon, Yueyang Liu, and Benjamin Van Roy. Continual learning as computationally constrained reinforcement learning. *CoRR*, abs/2307.04345, 2023.

Saurabh Kumar, Hong Jun Jeon, Alex Lewandowski, and Benjamin Van Roy. The need for a big world simulator: A scientific challenge for continual learning. In *Finding the Frame Workshop at Reinforcement Learning Conference*, 2024.

Alex Lewandowski, Dale Schuurmans, and Marlos C. Machado. Plastic learning with deep fourier features. In *International Conference on Learning Representations*, 2025.

Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, 4th Edition*. Texts in Computer Science. Springer, 2019.

Golnaz Mesbahi, Olya Mastikhina, Parham Mohammad Panahi, Martha White, and Adam White. Tuning for the unknown: Revisiting evaluation strategies for lifelong rl. *CoRR*, abs/2404.02113v2, 2024.

Edan J. Meyer, Adam White, and Marlos C. Machado. Harnessing discrete representations for continual reinforcement learning. *Reinforcement Learning Journal*, 2:606–628, 2024.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999v3, 2018.

Laurent Orseau and Mark Ring. Space-time embedded intelligence. In *International Conference on Artificial General Intelligence*, 2012.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Ameya Prabhu, Philip H.S. Torr, and Puneet K. Dokania. GDumb: A simple approach that questions our progress in continual learning. In *European Conference on Computer Vision*, 2020.

Paul Rendell. A universal turing machine in Conway's game of life. In *International Conference on High Performance Computing & Simulation*, 2011.

Noor Sajid, Philip J Ball, Thomas Parr, and Karl J Friston. Active inference: demystified and compared. *Neural computation*, 33(3):674–712, 2021.

J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010. ISSN 1943-0604. DOI: 10.1109/TAMD.2010.2056368.

R. J. Solomonoff. A formal theory of inductive inference. Part I. *Information and Control*, 7:1–22, 1964.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131:139–148, 2012.

J. Storck, S. Hochreiter, and J. Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pp. 159–164. EC2 & Cie, 1995.

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

Richard S. Sutton, Anna Koop, and David Silver. On the role of tracking in stationary environments. In *International Conference on Machine Learning*, 2007.

Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1937.

Vivek Veeriah, Shangtong Zhang, and Richard S Sutton. Crossprop: Learning representations by stochastic meta-gradient descent in neural networks. In *Machine Learning and Knowledge Discovery in Databases*, 2017.

Joel Veness, Kee Siong Ng, Marcus Hutter, William T. B. Uther, and David Silver. A monte-carlo AIXI approximation. *J. Artif. Intell. Res.*, 40:95–142, 2011. DOI: 10.1613/JAIR.3125. URL https://doi.org/10.1613/jair.3125.

Nikolai K. Vereshchagin and Paul M.B. Vitányi. Rate distortion and denoising of individual data using kolmogorov complexity. *IEEE Transactions on Information Theory*, 56:3438–3454, 2010.

Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L. Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, Christoph H. Lampert, Martin Mundt, Razvan Pascanu, Adrian Popescu, Andreas S. Tolias, Joost van de Weijer, Bing Liu, Vincenzo Lomonaco, Tinne Tuytelaars, and Gido M van de Ven. Continual learning: Applications and the road forward. *Transactions on Machine Learning Research*, 2024.

A Proofs

Proof of Proposition 1. Our proof is constructive, using the computationally universal environment's symbol-set, Σ, and construction function, $\mathbb{C}$, we define a Markov environment on a countably infinite state-space, Ω with a transition function, $\mathbb{U}$.

Constructing the state-space (Ω)

The state-space of the Markov environment is represented as a subset of a sequence-space over the union of the symbol-set and a blank symbol, $\Sigma \cup \{\square\}$. The finite string of symbols corresponds to a finite set of indices, whereas the other indices are all represented by the blank symbol.

For simplicity, we consider the bi-infinite sequence-space, $(\Sigma \cup \{\square\})^{\mathbb{Z}} = \{\{\omega_i\}_{i\in\mathbb{Z}} : \omega_i \in \Sigma \cup \{\square\}\}$, indexed by the set of integers, $\mathbb{Z}$. Other higher dimensional sequence-space are also possible, such as pairs of bi-infinite sequences, corresponding to a grid.

Each finite string, $\sigma_{-t:t} \in \Sigma^\star$, is encoded as a sequence padded with blank symbols, $E(\sigma_{-t:t}) = \omega = \{\dots, \square, \sigma_{-t}, \sigma_{-t+1} \dots, \sigma_{t-1}, \sigma_t, \square, \dots\} \in (\Sigma \cup \{\square\})^{\mathbb{Z}}$. The state-space is a countably infinite proper subset of the sequence-space considered, $\Omega \subsetneq (\Sigma \cup \{\square\})^{\mathbb{Z}}$, because each string is finite.

Constructing the transition function ($\mathbb{U}$)

The transition function is defined by a composition of a function that decodes the string from the sequence, and the construction function underlying the computationally universal environment.

The decoding function maps the current state, ω_t, and retrieves the string, $D(\omega_t) = \sigma_{0:t}$. In the bi-infinite sequence space that we consider, this involves scanning the input to find the blank symbols that delimit the indices that enclose the non-blank symbols. The decoding function returns the first and last symbol that is adjacent to the blank symbol, $\square$.

With the decoding function, the transition function then uses the construction function to obtain the next-symbol, $\sigma_{t+1} = \mathbb{C}(\sigma_{0:t})$. Finally, it concatenates the retrieved string with the next-symbol to produce the next string, $\sigma_{0:t+1}$. This string is then encoded as the next state, $\omega_{t+1} = E(\sigma_{0:t+1})$. In the bi-infinite sequence space that we consider, this involves replacing the last blank symbol with the new symbol σ_t. ☐

Proof of Proposition 2. We outline the transition dynamics of an embedded automaton within the environment. We will then show that, under assumptions of this boundary-space, such an automaton is equivalent to an agent interacting with a partially-observable Markov decision process.

The next internal sub-state is determined by the automaton's state-update function, $\mathbb{U}_A$. By definition, the boundary-space of the automaton, B_A, determines the one-step transition dynamics of the embedded automaton's internal sub-state. This means that the automaton's internal sub-state is updated according to $a_{t+1} = \mathbb{U}_A(a_t, b_t)$. Thus, the embedded automaton is a boundaried Markov process.

In the case where the input and output determine the boundary-space, $I_A \sqcup O_A = B_A$, the automaton can be completely separated from the universal-local environment. That is, conditioned on the input and output spaces, the automaton's next substate is determined. Moreover, because the automaton observes the input and output space, and because this determines the next-substate, the automaton has agency in the determination of its future substate by the outputs that it takes. The automaton can thus be viewed equivalent to an agent interacting with an environment, in the conventional reinforcement learning paradigm.

While this agent can completely determine its output, conditioned on its substate and input, it cannot in general predict or control its input. The input to such an embedded automaton is equivalent to an observation provided to it by the environment. This environment, by definition, is a Markov process in the countably infinite-state space Ω. Thus, from the view of the automaton, it is facing a partially-observable Markov decision process with a countably infinite state-space.

Note that the partially observable Markov decision process is, in general, reward-free. The environment could provide a reward to the agent, through the input-space, and the agent could be programmed in such a way to maximize the sum of its future reward signal. However, this requires additional assumptions about the environment and the agent and so we do not prescribe a reward function.

Thus, when the input and output spaces determine the boundary-space of an embedded automaton, it can be thought of as an agent interacting with a (reward-free) partially observable Markov decision process with a countably infinite state-space. ☐

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).

Thus, an embedded automaton simulated in a universal-local environment is implicitly constrained relative to its partially observable Markov decision process.

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.
2. **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 prove a more precise statement, that the maximum possible H-horizon interactivity for a finite-state automaton is upper bounded by its capacity. For an automaton, $\mathcal{A}$, the capacity is proportional to the size of its internal sub-state space, $|A|$. Encoding an automaton on a Turing machine is dominated by the cost of encoding its transition function which is on the order of $O(|A| \log |A|)$.

Because we make no assumptions on the environment (and hence the inputs), we consider only the complexity produced by the automaton's outputs, conditioned on its inputs, which we write simply by replacing b_t with π_t. Thus, the H-horizon interactivity that we consider is,

$$\mathbb{I}^\star_H(\mathcal{A}) = \frac{1}{H}\left(\mathbb{K}(\pi_{t:H}) - \mathbb{K}(\pi_{t:H} \mid \pi_{0:t-1})\right),$$

and we show that we can upper bound it in terms of capacity, $|A|$. All inequalities below pertaining to Kolmogorov complexity are subadditive, meaning they hide constant terms, $O(1)$.

By the symmetry of information (Li & Vitányi, 2019), we have

$$\mathbb{K}(\pi_{0:t-1}) \geqslant \left(\mathbb{K}(\pi_{t:H}) - \mathbb{K}(\pi_{t:H} \mid \pi_{0:t-1})\right) \tag{3}$$

Which we can use to upper bound H-horizon interactivity,

$$\frac{1}{H}\mathbb{K}(\pi_{0:t-1}) \geqslant \frac{1}{H}\left(\mathbb{K}(\pi_{t:H}) - \mathbb{K}(\pi_{t:H} \mid \pi_{0:t-1})\right) = \mathbb{I}^*_H(\mathcal{A}) \tag{4}$$

It remains to bound the complexity of the behaviour, $\mathbb{K}(\pi_{0:t-1})$. Because the behaviour is produced by an automaton, we have that the encoding length of the automaton upper bounds the minimum program that produces the sequence

$$|A| \log |A| \geqslant \min\{|c| : \mathcal{U}(c) = \pi_{0:t-1}\} := \mathbb{K}(\pi_{0:t-1}).$$

Putting this together, we have the desired upper bound in terms of capacity,

$$\frac{|A| \log |A|}{H} \geqslant \frac{1}{H}\mathbb{K}(\pi_{0:t-1}) \geqslant \mathbb{I}^\star_H(\mathcal{A}).$$

For a fixed automaton size, asymptotically such an upper bound goes to zero. That is, any bounded computation has bounded unnormalized interactivity, and zero asymptotic interactivity. However, for large and finite H, this upper bound is tight.

□

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 bounded learning agent, $\mathcal{A}$, with capacity $C(\mathcal{A})$

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

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, $\mathbb{A}(b_{t:H}|b_{0:t}) = 0$, has conditional Kolmogorov complexity is strictly less than the capacity of the agent, $\mathbb{K}(b_{t:H}|b_{0:t}) < C(\mathcal{A})$. In fact, we have, for $\alpha < 1$, that $\mathbb{K}(b_{t:H}|b_{0:t}) \leqslant \alpha C(\mathcal{A})$. 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, interactivity is effectively bounded by capacity,

$$(1-\alpha)C(\mathcal{A}) - O(1) \leqslant \mathbb{I}^{\star}_{H}(\mathcal{A}) \leqslant C(\mathcal{A}) + O(1).$$

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.

Suppose that the agent were to stop learning at time t. For the corresponding finite-state automaton, $\mathcal{A}$, the state transition function, $\mathbb{U}_A$ encodes the learning rule. An agent that has stopped learning is thus equivalent to an automaton that stops updating its internal state. In this case, the automaton's internal state remains constant $a_{t'} = a$ for all $t' > t$.

A finite-state automaton has a capacity on the order of $C(\mathcal{A}) = O(|I_A||A| \log |A|)$. 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 $C(\mathcal{A}^{-}) = O(|I_A|)$, because the terms needed to encode the transition function, $O(|A| \log |A|)$, 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.

□

B Experimental Details for Behavioural Self-Prediction

The problem that we consider involves predicting the learning algorithms own future predictions. We consider a function approximator in which its input space is equal to its output space, $\pi_\theta : X \to X$, and parameterized by θ. That is, the function approximator's output can be used as subsequent input. The learning algorithm updates the parameters of the function approximator, $U : \theta \to \theta$.

The function approximator is tasked with predicting the future average of its behaviour iteratively and online. At the first timestep, we randomly initialize the function approximator's parameters $\theta_0 \sim p(\theta)$ and randomly sample the initial input, $b_0 \sim p(b)$, according to standard initialization distributions. The function approximator, π_θ is then trained to maximize its interactivity. Specifically, the following steps are repeated at each timestep:

- A learning trajectory of H outputs is produced by iteratively updating the function approximator along the trajectory, using the learning algorithm to learn successor features with $TD(0)$. The sum of losses at each time step in the trajectory is a function of a sequence of parameter updates, and provides the estimate $\hat{\mathbb{A}}(b_{t:T}|b_{0:t})$.
- The trajectory of outputs produced by the iteratively updated function approximator is then used to update a copy of the function approximator which was not iteratively updated. Here, the sum of losses at each time step in the trajectory is an estimate for $\hat{\mathbb{A}}(b_{t:T}|b_{0:t-H})$.
- Interactivity is estimated by the difference in the two estimated of the agent-relativized complexity, $\hat{\mathbb{A}}(b_{t:T}|b_{0:t-H}) - \hat{\mathbb{A}}(b_{t:T}|b_{0:t})$.
- The same learning algorithm used to generate the learning trajectory is used to maximize the estimate of interactivity, which produces a single update to the parameters, $\theta' = U(\theta)$. This updated parameter is used to produce the output which will be used as the next input.

B.1 Experimental Results

We trained a two-layer network using either `Linear` or `ReLU` activations on the H-horizon approximation to interactivity, $H = 10$. We used conventional stochastic gradient descent which is generally more stable than adaptive methods (Finn et al., 2017), and used the same step-size for inner-learning of average future behaviour and for outer-learning of interactivity-maximizing behaviour. While this network is relatively shallow, the meta-gradient calculation for maximizing interactivity makes the effective network depth $2H = 20$ layers. Our findings indicate that linear methods are initially capable of fast adaptation, but that this always lead to performance collapse (Figure 4). This is surprising because linear methods are known to be stable baselines in conventional continual learning scenarios (Lewandowski et al., 2025; Dohare et al., 2024).

B.2 Interpreting Experimental Results As A Continual Learning Benchmark

Our experimental that maximizing interactivity requires balancing adaptation and stability. That is, maximizing interactivity involves the canonical plasticity-stability trade-off of continual learning (Grossberg & Grossberg, 1982; Parisi et al., 2019). 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. This environment represents the implicit computational constraints faced by an agent learning to predict its own future learning behaviour. Behavioural self-prediction specifically evaluates a learning algorithm's capabilities for continual adaptation. Thus, any algorithm that stops learning is suboptimal in this setting, regardless of its capacity, must continually learn to be optimal.

B.3 Limitations of Experiments

Our experiments used seemingly shallow networks, with a depth of $D = 2$. However, with the meta-gradient calculation over a finite horizon of $H = 10$, the effective depth of the networks during auto-differentiation is $H \cdot D = 20$. Meta-gradient methods for deep networks at depth can exhibit more pathological learning dynamics because they account for curvature when differentiating through gradients. 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-gradient is a second-order method, 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).

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 & 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 & 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 & 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.