Backprop-Free Reinforcement Learning with Active Neural Generative Coding

Alexander G. Ororbia[1], **Ankur Mali**[2]

[1] Rochester Institute of Technology
[2] The Pennsylvania State University
ago@cs.rit.edu, aam35@psu.edu

Abstract

In humans, perceptual awareness facilitates the fast recognition and extraction of information from sensory input. This awareness largely depends on how the human agent interacts with the environment. In this work, we propose *active neural generative coding*, a computational framework for learning action-driven generative models without backpropagation of errors (backprop) in dynamic environments. Specifically, we develop an intelligent agent that operates even with sparse rewards, drawing inspiration from the cognitive theory of planning as inference. We demonstrate on several simple control problems that our framework performs competitively with deep Q-learning. The robust performance of our agent offers promising evidence that a backprop-free approach for neural inference and learning can drive goal-directed behavior.

Introduction

Manipulating one's environment in the effort to understand it is an essential ingredient of learning in humans (Spielberger and Starr 1994; Berlyne 1966). In cognitive neuroscience, behavioral and neurobiological evidence indicates a distinction between goal-directed and habitual action selection in reward-based decision-making. With respect to habitual action selection, or actions taken based on situation-response associations, ample evidence exists to support the temporal-difference (TD) account from reinforcement learning. In this account, the neurotransmitter dopamine creates an error signal based on reward prediction to drive (state) updates in the corpus striatum, a particular neuronal region in the basal ganglia that affects an agent's choice of action. In contrast, goal-directed action requires prospective planning where actions are taken based on predictions of their future potential outcomes (Niv 2009; Solway and Botvinick 2012). Planning-as-inference (PAI) (Botvinick and Toussaint 2012) attempts to account for goal-directed behavior by casting it as a problem of probabilistic inference where an agent manipulates an internal model that estimates the probability of potential action-outcome-reward sequences.

One important, emerging theoretical framework for PAI is that of active inference (Friston, Mattout, and Kilner 2011; Tschantz, Seth, and Buckley 2020), which posits that biological agents learn a probabilistic generative model by interacting with their world, adjusting the internal states of this model to account for the evidence that they acquire from their environment. This scheme unifies perception, action, and learning in adaptive systems by framing them as processes that result from approximate Bayesian inference, elegantly tackling the exploration-exploitation trade-off inherent to organism survival. The emergence of this framework is timely – in reinforcement learning (RL) research, despite the recent successes afforded by artificial neural networks (ANNs) (Mnih et al. 2013; Silver et al. 2018), most models require exorbitant quantities of data to train well, struggling to learn tasks as efficiently as humans and animals (Arulkumaran et al. 2017). As a result, a key challenge is how to design RL methods that successfully resolve environmental uncertainty and complexity given limited resources and data. Model-based RL, explored in statistical learning research through world (Ha and Schmidhuber 2018) or dynamics models (Sutton 1990), offers a promising means of tackling this challenge and active inference provides a promising path towards instantiating it in a powerful yet neurocognitively meaningful way (Tschantz et al. 2020b).

Although PAI and active inference offer an excellent story for biological system behavior and a promising model-based RL setup, most computational implementations are formulated with explainability in mind (favoring meaningfully labeled albeit low-dimensional, discrete state/action spaces) yet in the form of complex probabilistic graphical models that do not scale easily (Friston et al. 2015, 2017a,b, 2018). In response, effort has been made to scale active inference by using deep ANNs (Ueltzhöffer 2018; Tschantz et al. 2020a) trained by the popular backpropagation of errors (backprop) (Rumelhart, Hinton, and Williams 1986). While ANNs represent a powerful step in the right direction, one common criticism of using them within the normative framework of RL is that they have little biological relevance despite their conceptual value (Lake et al. 2017; Zador 2019). Importantly, from a practical point-of-view, they also suffer from practical issues related to their backprop-centric design (Ororbia and Mali 2019). This raises the question: can a biologically-motivated alternative to backprop-based ANNs also facilitate reinforcement learning through active inference in a scalable way? In this paper, motivated by the fact that animals and humans solve the RL problem, we develop one alternative that positively answers this question. As a re-

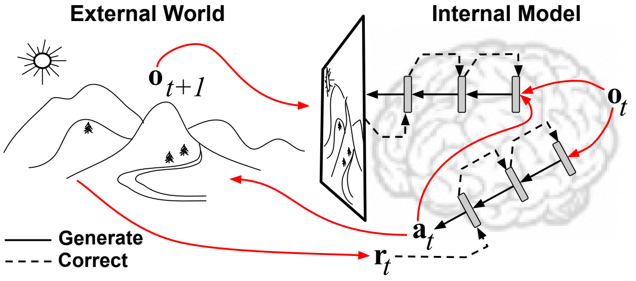


Figure 1: An ANGC agent predicts the world, acts to manipulate it, and then corrects itself given observations/rewards.

sult, the neural agent we propose represents a promising step forward towards better modeling the approximations that biological neural circuitry implements when facing real-world resource constraints and limitations, creating the potential for developing new theoretical insights. Such insights will allow us to design agents better capable of dealing with continuous, noisy sensory patterns (Niv 2009).

While many backprop-alternative (backprop-free) algorithms have recently been proposed (Movellan 1991; O'Reilly 1996; Lee et al. 2015; Lillicrap et al. 2016; Scellier and Bengio 2017; Guerguiev, Lillicrap, and Richards 2017; Whittington and Bogacz 2017; Ororbia and Mali 2019), few have been investigated outside the context of supervised learning, with some notable exceptions in sequence (Wiseman et al. 2017; Ororbia et al. 2020a; Manchev and Spratling 2020) and generative modeling (Ororbia and Kifer 2020). In the realm of RL, aside from neuro-evolutionary approaches (Such et al. 2017; Heidrich-Meisner and Igel 2009) or methods that build on top of them (Najarro and Risi 2020), the dearth of work is more prescient and our intent is to close this gap by providing a backprop-free approach to inference and learning, which we call *active neural generative coding* (ANGC), to drive goal-oriented agents. In our system, we demonstrate how a scalable, biologically-plausible inference and learning process, grounded in the theory of predictive processing (Friston 2005; Clark 2015), can lead to adaptive, self-motivated behavior in the effort to balance the exploration-exploitation trade-off in RL. One key element to consider is that ANGC offers robustness in settings with sparse rewards which other backprop-free methods such as neuroevolution (Such et al. 2017; Heidrich-Meisner and Igel 2009) struggle with.[1] To evaluate ANGC's efficacy, we implement an agent structure tasked with solving control problems often experimented with in RL and compare performance against several backprop-based approaches.

Active Neural Generative Coding

To specify our proposed ANGC agent, the high-level intuition of which is illustrated in Figure 1, we start by defining of the fundamental building block used to construct it, i.e., the neural generative coding circuit. Specifically, we examine its neural dynamics (for figuring out hidden layer values given inputs and outputs) and its synaptic weight updates.

The Neural Generative Coding Circuit

Neural generative coding (NGC) is a recently developed framework (Ororbia and Kifer 2020) that generalizes classical ideas in predictive processing (Rao and Ballard 1999; Clark 2015) to the construction of scalable neural models that model and predict both static and temporal patterns (Ororbia et al. 2020a; Ororbia and Kifer 2020). An NGC model is composed of $L+1$ layers of stateful neurons $\mathfrak{N}^0, \mathfrak{N}^1, \cdots, \mathfrak{N}^L$ that are engaged in a process of never-ending guess-then-correct, where $\mathfrak{N}^\ell$ contains J_ℓ neurons (each neuron has a latent state value represented by a scalar). The combined latent state of the neurons in $\mathfrak{N}^\ell$ is represented by the vector $\mathbf{z}^\ell \in \mathcal{R}^{J_\ell \times 1}$ (initially $\mathbf{z}^\ell = \mathbf{0}$ in the presence of a new data pattern). Generally, an NGC model's bottom-most layer $\mathfrak{N}^0$ is clamped to a sensory pattern extracted from the environment. However, in this work, we design a model that clamps both its top-most layer $\mathfrak{N}^L$ and bottom-most layer $\mathfrak{N}^0$ to particular sensory variables, i.e., $\mathbf{z}^L = \mathbf{x}^i$ and $\mathbf{z}^0 = \mathbf{x}^o$, allowing the agent to process streams of data $(\mathbf{x}^i, \mathbf{x}^o)$ where $\mathbf{x}^i \in \mathcal{R}^{J_L \times 1}$ and $\mathbf{x}^o \in \mathcal{R}^{J_0 \times 1}$.

Specifically, in an NGC model, layer $\mathfrak{N}^{\ell+1}$ attempts to guess the current post-activity values of layer $\mathfrak{N}^\ell$, i.e., $\phi^\ell(\mathbf{z}^\ell)$, by generating a prediction vector $\bar{\mathbf{z}}^\ell$ using a matrix of forward synaptic weights $\mathbf{W}^{\ell+1} \in \mathcal{R}^{J_\ell \times J_{\ell+1}}$. The prediction vector is then compared against the target activity by a corresponding set of error neurons $\mathbf{e}^\ell$ which simply perform a direct mismatch calculation as follows: $\mathbf{e}^\ell = \frac{1}{\beta_e}(\phi^\ell(\mathbf{z}^\ell) - \bar{\mathbf{z}}^\ell)$.[2] This error signal is finally transmitted back to the layer that made the prediction $\bar{\mathbf{z}}^\ell$ through a complementary matrix of error synapses $\mathbf{E}^{\ell+1} \in \mathcal{R}^{J_{\ell+1} \times J_\ell}$. Given the description above, the set of equations that characterize the NGC neural circuit and its key computations are:

$$\bar{\mathbf{z}}^\ell = g_\ell\Big(\mathbf{W}^{\ell+1} \cdot \phi^{\ell+1}(\mathbf{z}^{\ell+1})\Big), \; \mathbf{e}^\ell = \frac{1}{2\beta_e}(\phi^\ell(\mathbf{z}^\ell) - \bar{\mathbf{z}}^\ell) \tag{1}$$

$$\mathbf{z}^{\ell+1} \leftarrow \mathbf{z}^{\ell+1} + \beta\Big(\overbrace{-\gamma_v \mathbf{z}^{\ell+1}}^{\text{leak}} + \overbrace{\mathbf{d}^{\ell+1}}^{\text{pressure}} + \overbrace{\vartheta(\mathbf{z}^{\ell+1})}^{\text{lateral term}})\Big) \tag{2}$$
$$\text{where } \mathbf{d}^{\ell+1} = -\mathbf{e}^{\ell+1} + (\mathbf{E}^{\ell+1} \cdot \mathbf{e}^\ell)$$

where $\cdot$ indicates matrix/vector multiplication and $\phi^{\ell+1}$ and g_ℓ are element-wise activation functions, e.g., the hyperbolic tangent $tanh(v) = (\exp(2v) - 1)/(\exp(2v) + 1)$ or the linear rectifier $\phi^\ell(v) = max(0, v)$. In this paper, we set g_ℓ as the identity, i.e., $g_\ell(v) = v$, for all layers. In Equation 2, the coefficient that weights the correction applied to state layer $\ell+1$ is determined by the formula $\beta = \frac{1}{\tau}$ where τ is the integration time constant in the order of milliseconds. The leak variable $-\gamma_v \mathbf{z}^\ell$ decays the state value over time (γ_v is a positive coefficient to control its strength). $\vartheta(\mathbf{z}^\ell)$ is lateral excita-

[1] It is difficult to determine a strong encoding scheme as well as an effective breeding strategy for the underlying genetic algorithm. Such design choices play a large role in the success of the approach.

[2] One may replace $\frac{1}{2\beta_e}$ with Σ^{-1}, i.e., a learnable matrix that applies precision-weighting to the error units, as in (Ororbia and Kifer 2020). We defer using this scheme for future work.

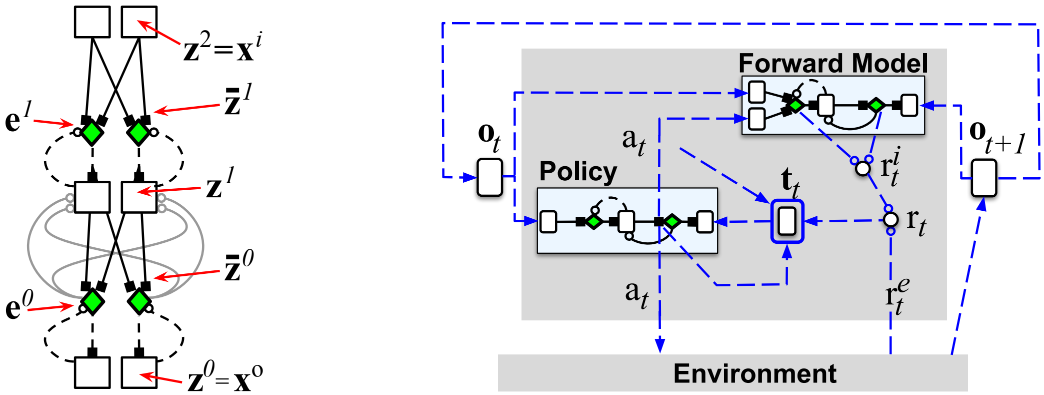


Figure 2: The NGC circuit (left) and the high-level ANGC architecture, with both a controller and generator, (right). Green diamonds represent error neurons, empty rectangles represent state neurons, solid arrows represent individual synapses, dashed arrows represent direct copying of information, open circles indicate excitatory signals, and filled squares indicate inhibitory signals. Note that $\mathbf{t}_t$, which is enclosed in a rounded blue box, entails a more intricate calculation (see Equation 6).

tion/inhibition term, which is a function that creates competition patterns among the neurons inside of layer $\mathfrak{N}^\ell$ (Ororbia and Kifer 2020) – in this paper we set $\vartheta(\mathbf{z}^\ell) = 0$ since its effect is not needed for this study. Upon encountering data $(\mathbf{x}^i, \mathbf{x}^o)$, the model's top and bottom layers are clamped, i.e., $\mathbf{z}^L = \mathbf{x}^i$ and $\mathbf{z}^0 = \mathbf{x}^o$, and Equations 1-2 are run K times in order to search for activity values $\{\mathbf{z}^1, \cdots, \mathbf{z}^{L-1}\}$ (see INFER in the pseudocode provided in the Appendix[3]).

After the internal activities have been found, the synaptic weights may be adjusted using a modulated local error Hebbian rule adapted from local representation alignment (LRA) (Ororbia and Mali 2019; Ororbia et al. 2020b):

$$\Delta \mathbf{W}^\ell = \mathbf{e}^\ell \cdot (\phi^\ell(\mathbf{z}^{\ell+1}))^T \otimes \mathbf{M}^\ell_W \tag{3}$$

$$\Delta \mathbf{E}^\ell = \gamma_e(\mathbf{W}^\ell_t - \mathbf{W}^\ell_{t-1}) \otimes \mathbf{M}^\ell_E \tag{4}$$

where γ_e controls the time-scale at which the error synapses are adjusted (usually values in the range of $[0.9, 1.0]$ are used). The dynamic modulation factors $\mathbf{M}^\ell_W$ and $\mathbf{M}^\ell_E$ help to stabilize learning in the face of non-stationary streams and are based on insights related to nonlinear synaptic dynamics (see Appendix for a full treatment of these factors). Once the weight adjustments have been computed, an update rule such as stochastic gradient ascent, Adam (Kingma and Ba 2014), or RMSprop (Tieleman and Hinton 2012) can be used (see UPDATEWEIGHTS in Algorithm CC, Appendix).

The online objective that an NGC model attempts to minimize is known as total discrepancy (TotD) (Ororbia et al. 2017), from which the error neuron, state update expressions, and local synaptic adjustments may be derived (Ororbia et al. 2020b; Ororbia and Kifer 2020). The (TotD) objective, which could also be interpreted as a form of free energy (Friston 2009) specialized for stateful neural models that utilize arbitrary forward and error synaptic pathways (Ororbia et al. 2020b), can be expressed in many forms including the linear combination of local density functions (Ororbia and Kifer 2020) or the summation of distance functions (Ororbia et al. 2020a). For this study, the form of TotD used is:

$$\mathcal{L}(\Theta) = \sum_{\ell=0}^{L-1} \mathcal{L}(\mathbf{z}^\ell, \bar{\mathbf{z}}^\ell) = \sum_{\ell=0}^{L-1} \frac{1}{\beta_e} ||\phi^\ell(\mathbf{z}^\ell) - \bar{\mathbf{z}}^\ell||^2_2.$$

Algorithm CC (see Appendix), puts all of the equations and relevant details presented so far together to describe how NGC processes $(\mathbf{x}^i, \mathbf{x}^o)$ from a data stream. We note here that the algorithm breaks down the processing into three routines – INFER($\circ$), UPDATEWEIGHTS($\circ$), and PROJECT($\circ$). INFER($\circ$) is simply the K-step process described earlier to find reasonable values of the latent state activities given clamped data and UPDATEWEIGHTS($\circ$) is the complementary procedure used to adjust the synaptic weight parameters once state activities have been found after using INFER($\circ$). PROJECT($\circ$) is a special function that specifically clamps data $\mathbf{x}^i$ to the top-most layer and projects this information directly through the underlying directed graph defined by the NGC architecture – this routine is essentially a variant of the ancestral sampling procedure defined in (Ororbia and Kifer 2020) but accepts a clamped input pattern instead of samples drawn from a prior distribution. Figure 2 (left) graphically depicts a three layer NGC model with 2 neurons per layer.

Generalizing to Active Neural Coding

Given the definition of the NGC block, we turn our attention to its generalization that incorporates actions, i.e., active NGC (ANGC). ANGC is built on the premise that an agent adapts to its environment by balancing a trade-off between (at least) two key quantities – surprisal and preference. This means that our agent is constantly tracking a measurement of how surprising the observations it encounters are at a given time step (which drives exploration) as well as a measurement of its progress towards a goal. In effect, maximizing the sum of these two terms means that the agent will seek observations that are most "suprising" (which yield the most information when attempting to reduce uncertainty) while attempting to reduce its distance to a goal state (which maximizes the discounted long-term future reward). Formally,

[3] Appendix at: https://arxiv.org/abs/2107.07046.

this means that an ANGC agent will maximize:

$$r_t = \alpha_e r_t^e + \alpha_i r_t^i = r_t^{ep} + r_t^{in} \tag{5}$$

which is a reward signal that can be decomposed into an epistemic (or exploration/information maximizing) signal r_t^{ep} and an instrumental (or goal-oriented) signal r_t^{in}. Each component signal is a product of an importance factor, α_e for the epistemic term and α_i for the instrumental term, and a raw internal signal produced either by the generative model (r_t^e to drive r_t^{ep}) or an external goal-directing signal (r_t^i to drive r_t^{in}). Although we chose to interpret and represent the active inference view of the exploration-exploitation trade-off as (dopamine) scalars, our instrumental signal is not limited to this scheme and could incorporate an encoding of more complex functions such as (prior) distribution functions over (goal) states (see Appendix).

As indicated by the architecture diagram in Figure 2 (right), the implementation of our ANGC agent in this paper is a coupling of two NGC circuits – the generator (or dynamic generative model), which is responsible for producing the epistemic term r_t^{ep}, and the controller, which is responsible for choosing the actions such that the full reward r_t, which includes the instrumental term r_t^{in}, is maximized.

The NGC Generator (Forward Model) Once the generator's top-most latent state is clamped to the current D-dimensional observation $\mathbf{o}_t \in \mathcal{R}^{D \times 1}$ and the 1-of-A encoding of the controller's currently chosen action a_t (out of A possible actions), i.e., $\mathbf{a}_t \in \{0,1\}^{A \times 1}$, the generator attempts to predict the value of the next observation of the environment $\mathbf{o}_{t+1}$. Using the routine INFER (Algorithm CC, Appendix), the generator, with parameters Θ_g, searches for a good set of latent state activities to explain output $\mathbf{x}^o = \mathbf{o}_{t+1}$ given input $\mathbf{x}^i = [\mathbf{a}_t, \mathbf{o}_{t+1}]$ where $[\cdot, \cdot]$ indicates the vector concatenation of $\mathbf{o}_t$ and $\mathbf{a}_t$. Once latent activities have been found, the generator then updates its synapses via routine UPDATEWEIGHTS (Algorithm CC, Appendix).

The generator plays a key role in that it drives the exploration conducted by an ANGC agent. Specifically, as the generator progressively learns to how to synthesize future observations, the current activities of its error neurons embedded at each layer, i.e., $\mathcal{E} = \{\mathbf{e}^0, \mathbf{e}^1, \cdots, \mathbf{e}^L\}$, are used to produce an epistemic modulation term. Formally, this means that the exploration signal is calculated as $r_t^e = \sum_\ell ||\mathbf{e}^\ell||_2^2$ which is the result of summing across layers and across each error neuron vector's respective dimensions.[4] The epistemic term $r_t^{ep} = \alpha_e r_t^e$ is then combined with an instrumental term $r_t^{in} = \alpha_i r_t^i$, i.e., the scalar signal produced externally (by the environment or another neural system), to guide the agent towards a goal state(s), via Equation 5. The final r_t is then subsequently used to adapt the controller (described next).

The NGC Controller (Policy Model) With its top-most latent state clamped to the tth observation, i.e., $\mathbf{x}^i = \mathbf{o}_t$, the controller, with parameters Θ_c, will generate a prediction of the full reward signal r_t. Specifically, at time t, given a target scalar (produced by the environment and the generator), the controller will also infer a suitable set of latent activities using the INFER routine defined in Algorithm CC (Appendix).

Since the NGC controller's output layer will estimate a potential reward signal for each possible discrete action that the agent could take (which is typical in many modern Q-learning setups), we must first compose the target activity $\mathbf{t}_t$ for the output nodes once the scalar value r_t is obtained. This is done by first encoding the action as a 1-of-A vector $\mathbf{a}_t$ (this is what done by the TOONEHOT function call in Algorithm 1), computing the boot-strap estimate of the future discounted reward $\mathbf{d}_{t+1} = \text{PROJECT}(\mathbf{o}_{t+1}, \Theta_c)$, and finally checking if the next observation is a terminal. Specifically, the target vector is computed according to the following:

$$\mathbf{t}_t = t_t \mathbf{a}_t + (1 - \mathbf{a}_t) \otimes \text{PROJECT}(\mathbf{o}_t, \Theta_c) \tag{6}$$

where the target scalar t_t is created according to the following logical expression:

$$(\mathbf{o}_t \text{ is terminal} \rightarrow t_t = r_t) \wedge (\mathbf{o}_t \text{ is not terminal} \rightarrow t'_t) \tag{7}$$
$$\text{where } \; t'_t = r_t + \gamma \max_a \text{PROJECT}(\mathbf{o}_{t+1}, \Theta_c)).$$

Once $\mathbf{t}_t$ has been prepared, the controller is run to find its latent activities for $\mathbf{o}_t$ and $\mathbf{t}_t$ using INFER and calculates its local weight updates via the UPDATEWEIGHTS routine (Algorithm CC, Appendix). Furthermore, observe that the second sub-expression in Equation 7 involves re-using the controller to estimate the (reward) value of the future observation, i.e., $\gamma \max_a \text{PROJECT}(\mathbf{o}_{t+1}, \Theta_c)$ term. This term can be replaced with a proxy term $\gamma \max_a \text{PROJECT}(\mathbf{o}_{t+1}, \widehat{\Theta}_c)$ to implement the target network stability mechanism proposed in (Mnih et al. 2015), where $\widehat{\Theta}_c$ are the parameters of a "target controller", initialized to be the values of Θ_c at the start of the simulation and updated every C transitions or by Polyak averaging $\widehat{\Theta}_c = \tau_c \Theta_c + (1 - \tau_c)\widehat{\Theta}_c$.

The ANGC Agent: Putting It All Together At a high level, the ANGC operates, given observation $\mathbf{o}_t$, according to the following steps: 1) the NGC controller takes in $\mathbf{o}_t$ and uses it to produce a discrete action, 2) the ANGC agent next receives observation $\mathbf{o}_{t+1}$ from the environment, i.e., the result of its action, 3) the NGC generator runs the dynamics in Equations 1-2 to find a set of hidden neural activity values that allow a mapping from $[\mathbf{a}, \mathbf{o}_t]$ to $\mathbf{o}_{t+1}$ and then updates its own specific synapses using Equations 3-4, 4) the reward r_t is computed using the extrinsic/problem-specific reward plus the epistemic signal (produced by summing the layer-wise errors inside the generator, i.e., total discrepancy), 5) the NGC controller then runs the dynamics in Equations 1-2 to find a set of hidden neural activity values that allow a mapping from $\mathbf{o}_t$ to $\mathbf{t}_t$ (which contains r_t) and then updates its synapses via Equations 3-4, and, finally, 6) the ANGC agent transitions to $\mathbf{o}_{t+1}$ and moves back to step 1.

The above step-by-step process shows that the NGC generator (forward model) drives information-seeking behavior (facilitating exploration better than that of random epsilon-greedy), allowing the ANGC agent to evaluate if an incoming state will allow for a significant reduction in uncertainty about the environment. The NGC controller (policy) is responsible for estimating future discounted rewards, balancing the seeking of a goal state (since the instrumental term

[4] Observe that r_t^e is the generator's TotD, i.e., $r_t^e = \mathcal{L}(\Theta_g)$.

Algorithm 1: The ANGC total discrepancy process under an environment for E episodes (of maximum length T).

```
**Input:** environment $\mathbb{S}$, controller $\Theta_c$, generator $\Theta_g$, deque memory $\mathcal{M}$, and constants $E$, $T$, $\alpha_e$, $\alpha_i$, $\epsilon_{decay}$, $\epsilon$, $\gamma$
function SimulateProcess($\mathbb{S}, E, T, \Theta_c, \Theta_g, \mathcal{M}, \alpha_e, \alpha_i, \epsilon, \epsilon_{decay}$)
    $r^e_{max} = 1$
    for $e = 1$ to $E$ do
        $\mathbf{o}_t \leftarrow \mathbf{o}_0$ from $\mathbb{S}$        ▷ Get initial state/observation from environment
        for $t = 1$ to $T$ do
            // Sample action $a_t$ according to an $\epsilon$-greedy policy
            $\mathbf{d}_t = \textsc{Project}(\mathbf{o}_t, \Theta_c)$, $p \sim \mathbb{U}(0, 1)$
            $\Big(p < \epsilon \rightarrow a_t \sim \mathbb{U}_d(1, A)\Big) \wedge \Big(p \geq \epsilon \rightarrow a_t = \arg\max_a \mathbf{d}_t\Big)$,  $\mathbf{a}_t = \textsc{toOneHot}(a_t)$
            // Get next state/observation from environment & compute component reward signals
            $(r^i_t, \mathbf{o}_{t+1}) \leftarrow \mathbb{S}(a_t)$,  $(\Lambda, \mathcal{E}) = \textsc{Infer}([\mathbf{a}_t, \mathbf{o}_t], \mathbf{o}_{t+1}, \Theta_g)$
            $r^e_t = \sum_\ell ||\mathcal{E}[\ell]||^2_2$,  $r^e_{max} = \max(r^e_t, r^e_{max})$,  $r^e_t \leftarrow \frac{r^e_t}{r^e_{max}}$,  $r_t = \alpha_e r^e_t + \alpha_i r^i_t$
            // Store transition and update weights from samples in memory
            Store $(\mathbf{o}_t, a_t, r_t, \mathbf{o}_{t+1})$ in $\mathcal{M}$
            $(\mathbf{o}_j, a_j, r_j, \mathbf{o}_{j+1}) \sim \mathcal{M}$        ▷ Sample mini-batch of transitions from memory
            $t_j = \begin{cases} r_j & \text{if } \mathbf{o}_j \text{ is terminal} \\ r_j + \gamma \max_a \textsc{Project}(\mathbf{o}_{j+1}, \Theta_c) & \text{otherwise} \end{cases}$
            $\mathbf{a}_j = \textsc{toOneHot}(a_j)$,  $\mathbf{t}_j = t_j \mathbf{a}_j + (1 - \mathbf{a}_j) \otimes \textsc{Project}(\mathbf{o}_j, \Theta_c)$
            $(\Lambda_c, \mathcal{E}_c) = \textsc{Infer}(\mathbf{o}_j, \mathbf{t}_j, \Theta_c)$,  $\Theta_c \leftarrow \textsc{UpdateWeights}(\Lambda_c, \mathcal{E}_c, \Theta_c)$        ▷ Update controller $\Theta_c$
            $(\Lambda_g, \mathcal{E}_g) = \textsc{Infer}([\mathbf{a}_j, \mathbf{o}_j], \mathbf{o}_{j+1}, \Theta_g)$, $\Theta_g \leftarrow \textsc{UpdateWeights}(\Lambda_g, \mathcal{E}_g, \Theta_g)$        ▷ Update generator $\Theta_g$
        $\epsilon \leftarrow \max(0.05, \epsilon \cdot \epsilon_{decay})$
```

represents the "desire" to solve the problem) with the search for states that will give it the most information about its environment. The controller keeps the agent focused on finding a goal state(s) and reinforcing discovered sequences of actions that lead to these goal states (exploitation) while the generator forces the agent to parsimoniously explore its world and seek elements that it knows least about but will likely help in finding goal states – this reduces the number of episodes and/or sampled states needed to uncover useful policies.

Given that ANGC is inspired by active inference (Friston, Mattout, and Kilner 2011), the intuition behind our approach is that an agent reduces the divergence between its model of the world and the actual world by either: 1) changing its internal model (the generator) so that it better aligns with sampled observations (which is why it seeks states with high epistemic/total discrepancy values), or 2) changing its observations such that they align with its internal model through action, i.e., this is done through the controller tracking its problem-specific performance either through extrinsic reward values or other functions, i.e., prior preferences (Tschantz, Seth, and Buckley 2020). The ANGC agent's balancing act between finding goal states with better exploring its environment strongly relates to the rise of computational curiosity in the RL literature (Pathak et al. 2017) – since we focus on using problem-specific rewards (instead of crafting prior preference distributions as in (Friston, Daunizeau, and Kiebel 2009; Tschantz, Seth, and Buckley 2020)) our agent's instrumental term (in Equation 5) is akin to extrinsic reward and our epistemic term is similar in spirit to intrinsic "curiosity" (Oudeyer 2018; Burda et al. 2018a) (error-based curiosity). Although our curiosity term is the total discrepancy of the NGC generator, this term connects ANGC to the curiosity-based mechanisms and exploration bonuses (Wu and Tian 2016) used to facilitate efficient extraction of robust goal-state seeking policies. Furthermore, the generator component of the ANGC agent connects our work with the recently growing interest in model-based RL where world models are integrated into the agent-environment interaction process, e.g., plan2explore (Sekar et al. 2020), dreamer (Hafner et al. 2019), etc. In a sense, one could view our ANGC as a simple, neurobiologically-plausible instantiation of these kinds of model-based RL approaches, offering a means to train similar systems without backprop. Many other bio-inspired algorithms, such as equilibrium propagation (Scellier and Bengio 2017), do not scale easily to RL problems due to expensive inference phases (whereas NGC's inference is faster – see Appendix for details).

In essence, the proposed ANGC framework prescribes the joint interaction of the controller and generator modules described above. At each time step, the agent, given observation $\mathbf{o}_t \in \mathcal{R}^{D \times 1}$ (which could contain continuous or discrete variables), is to perform a discrete action a_t[5] and receive from its environment the result of its action, i.e., observation $\mathbf{o}_{t+1}$ and possibly an external reward signal r^{ep}_t. The controller is responsible for deciding which action to take next while the generator actively attempts to guess the (next) state of the agent's environment. Upon taking an action a_t, the generator's prediction is corrected using the values within the sensory sample drawn from the environment, allowing it to iteratively craft a compressed internal impression of the agent's world. The inability of the generator to

[5] We focus on discrete actions in this study and leave generalization to continuous actions for future work.

accurately predict the incoming sensory sample $\mathbf{o}_{t+1}$ provides the agent with a strong guide to its exploration of the environment, reducing its (long-term) surprisal and improving the controller's ability to extract an effective policy/plan.

The complete ANGC agent is specified in Algorithm 1[6] and depicted in Figure 2 (right). Note that Algorithm 1 implements the full simulation of ANGC's inference and synaptic adjustment over an E-episode long stream (each episode is at most T steps long – where T could vary with time). In addition to the target controller modification described earlier, we integrate experience replay memory $\mathcal{M}$ (O'Neill et al. 2010; Mnih et al. 2015) (implemented as a ring buffer with mini-batches sampled, uniformly at random, from stored transitions). This stabilizes the learning process by removing correlations in the observation sequence.

Experiments

The performance of the ANGC agent is evaluated on three control problems commonly used in reinforcement learning (RL) and one simulation in robotic control. Specifically, we compare ANGC to a random agent (where actions are taken at each step uniformly at random), a deep Q-network (DQN) (Mnih et al. 2015), the intrinsic curiosity module (ICM) (Pathak et al. 2017), and another powerful intrinsic curiosity baseline known as random network distillation (RnD) (Burda et al. 2018b) on: 1) the inverted pendulum (cartpole) problem, 2) the mountain car problem 3) the lunar lander problem, and 4) the robot reaching problem. Details related to each control problem are provided in the Appendix.

ANGC Agent Setup: For all of the ANGC agents across all trials, we used fixed configurations of meta-parameters for each control task. We provide the key values chosen (based on preliminary experimentation) for the meta-parameters for all ANGC models in the Appendix.

For all ANGC agents, $\alpha_e = \alpha_i = 1.0$ was used as the importance factors for both the epistemic and instrumental signals. Both the controller and generative model were trained using a single, shared experience replay buffer with a maximum capacity of N_{mem} transitions from which mini-batches of N_{batch} transitions were sampled in order to compute parameter updates at any single time-step of each simulation. Each agent also uses an epsilon(ϵ)-greedy policy where ϵ was decayed at the end of each episode according to the rule: $\epsilon \leftarrow \min(0.05, \epsilon * \epsilon_{decay})$ (starting $\epsilon = 1$ at a trial's start).

In addition, we experiment with an ablated form of our ANGC agent, i.e., Instr-ANGC, where the generator/forward model has been removed. This means that the Instr-ANGC only uses the extrinsic reward signal, allowing for a closer examination of what happens if the normal backprop-based neural model was just replaced with our NGC circuit.

Baseline Agent Setups: For the DQN, ICM, and RnD agents, we initially start with 90% exploration and 10% exploitation ($\epsilon = 0.9$) and eventually begin decaying until the condition of 10% exploration is reached, i.e., 90% exploitation ($\epsilon = 0.1$). The discount factor was tuned in the range of $\gamma = [0.91, 0.99]$. The linear rectifier was used as the activation function and Adam was used to update weight values, except for ICM, where AdamW was found to yield more stable updates. For all models, each W^ℓ was initialized according to a centered Gaussian scaled by $\sqrt{2.0/(J_{\ell-1} + J_\ell)}$. The replay buffer size, the learning rate, the hidden dimensions, and number of layers were tuned – hidden layer sizes were selected from the range of $[32, 512]$ and the number of layers was chosen from the set $[1, 2, 3]$. In the Appendix, we provide best configurations used for each model.

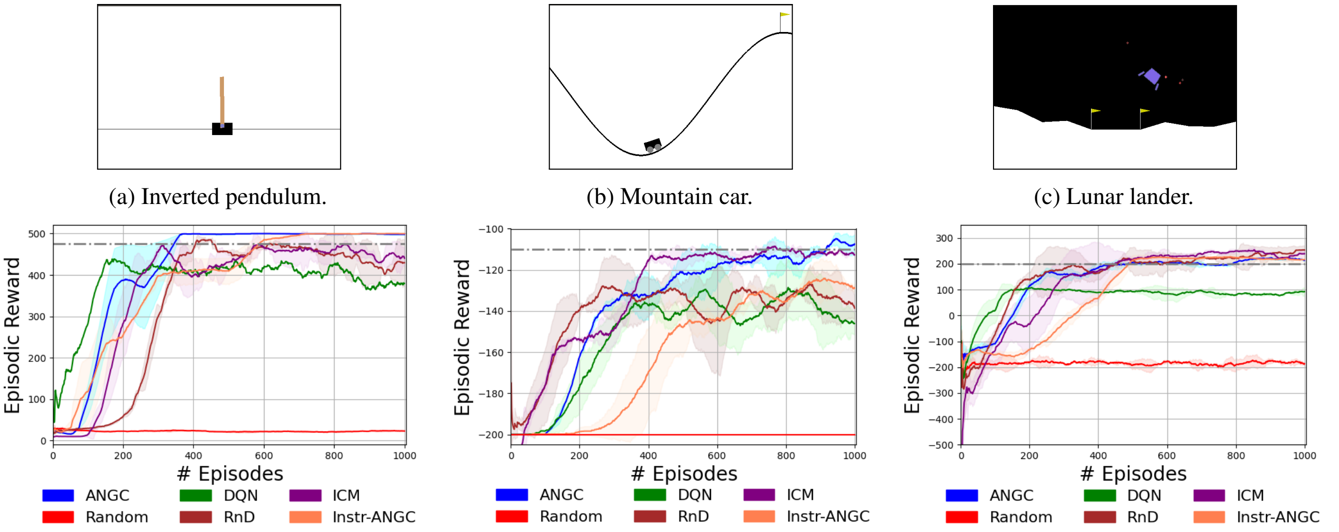
Results: In Figures 3 and 4, we visualize the accumulated reward as a function of the episode count (over the first 1000 episodes, see Appendix for expanded results), smoothing out the curves by plotting the moving average reward as a function of the episode count, i.e., $\mu_t = 0.1 r_t + 0.9 \mu_{t-1}$. Results are averaged over 10 trials and the plots present both the mean (darker color central curve) and standard deviation (lighter color envelope). In each plot, a dash-dotted horizontal line depicts the threshold for fully solving each task.

It is immediately apparent from our reward plots, across all four benchmarks, that the ANGC agent is notably competitive with backprop-based, intrinsic curiosity models (ICM, RnD), extracting a better policy than models that do not incorporate intelligent exploration (the DQN). This highlights the value of our ANGC framework for designing agents – for each of the benchmarks we investigate, fewer episodes are required by ANGC agents to generalize well and even ultimately solve a given control problem (as indicated by their ability to reach each problem's solution threshold). Furthermore, ANGC offers stable performance in general, where models like the RnD sometimes do not (as on the mountain car problem), and is notably quite competitive with ICM in general, even outperforming it on the more complex robotic arm control task. We also note that the DQN reaches its goal quickly in some cases but struggles to maintain itself at the solution threshold, fluctuating around that range (requiring more episodes to fully stabilize).

Crucially, observe that the ANGC agent is capable of effectively tackling control problems involving extremely sparse rewards (or nearly non-existent reward signals) as indicated by its early strong performance on the mountain car and robot reaching problems, which are arguably the hardest of the problems examined in this study. The ANGC's effectiveness on these problems is, we argue, largely due to its own internally generated epistemic modulation factor r^{ep}_t (this is empirically corroborated by the Instr-ANGC's worse performance than the full ANGC). In other words, the ANGC agent explores states that surprise it most, meaning it is most "curious" about states that yield the highest magnitude total discrepancy (or the greatest free energy). This feature presents a clean NGC implementation of the epistemic term key to the active inference framework (Friston et al. 2017b) which, theoretically, is meant to encourage a more principled, efficient form of environmental exploration. Furthermore, this term, much akin to intrinsic curiosity models (Pathak et al. 2017), would allow the agent to operate in settings where even no external reward is available.

Note that the intent of this study was not to engage

[6] $\mathcal{E}[\ell]$ means "retrieve the ℓth item in $\mathcal{E}''$.



(a) Inverted pendulum.

(b) Mountain car.

(c) Lunar lander.

Figure 3: Reward curves for ANGC and baselines (DQN, ICM, RnD, and the ablated model Instr-ANGC). Mean and standard deviation over 10 trials are plotted. Dash-dotted, horizontal (gray) lines depict the problem solution threshold.

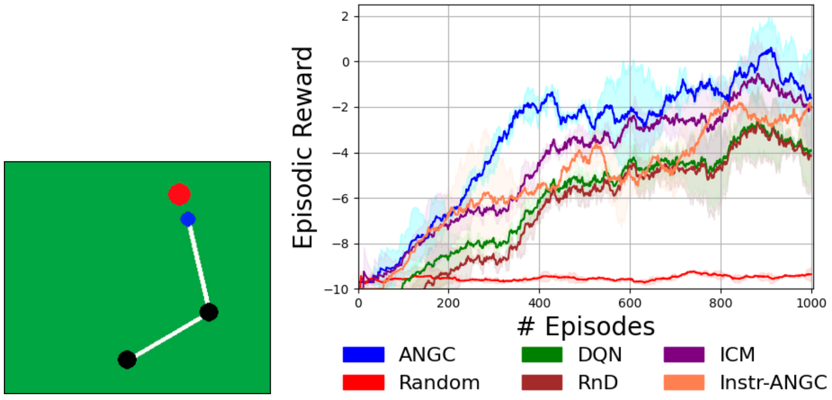


Figure 4: The robot arm reaching problem results. Mean and standard deviation over 10 trials plotted.

in a performance contest given that there is a vast array of problem-tuned, neural-based RL approaches that attain state-of-the-art performance on many tasks. Instead, the intent was to present a promising alternative to backprop-based approaches and demonstrate that ANGC can acquire good policies on control problems that DQN-based models can. Since the results are promising and our framework has proven to be compatible with commonly-used RL heuristics such as experience replay and target network stability, integrating other mechanisms would be a fruitful next step to further improve performance. In the Appendix, we discuss the limitations of ANGC as well as its relationship with free energy optimization, examine ANGC in the context of related work in RL and planning as inference, and conduct further analysis on the control problems presented above.

Conclusion

In this paper, we proposed active neural generative coding (ANGC), a framework for learning goal-directed agents without backpropagation of errors (backprop). We demonstrated, on four control problems, the effectiveness of our approach for learning systems that are competitive with backprop-based ones such as the deep Q-network and intrinsic curiosity variants. Notably, our framework demonstrates the value of leveraging the neuro-biologically grounded learning and inference mechanisms of neural generative coding to dynamically adapt a generative model that provides epistemic signals to augment problem-specific extrinsic rewards. Furthermore, given its robustness, ANGC could prove useful in more complex environments, e.g., the Atari games, challenging robotics problems, offering an important means of implementing longer-term planning-as-inference.

References

Arulkumaran, K.; Deisenroth, M. P.; Brundage, M.; and Bharath, A. A. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Berlyne, D. E. 1966. Curiosity and exploration. *Science*, 153(3731): 25–33.

Botvinick, M.; and Toussaint, M. 2012. Planning as inference. *Trends in cognitive sciences*, 16(10): 485–488.

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

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

Clark, A. 2015. *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press.

Friston, K. 2005. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences*, 360(1456): 815–836.

Friston, K. 2009. The free-energy principle: a rough guide to the brain? *Trends in cognitive sciences*, 13(7): 293–301.

Friston, K.; FitzGerald, T.; Rigoli, F.; Schwartenbeck, P.; and Pezzulo, G. 2017a. Active inference: a process theory. *Neural computation*, 29(1): 1–49.

Friston, K.; Mattout, J.; and Kilner, J. 2011. Action understanding and active inference. *Biological cybernetics*, 104(1): 137–160.

Friston, K.; Rigoli, F.; Ognibene, D.; Mathys, C.; Fitzgerald, T.; and Pezzulo, G. 2015. Active inference and epistemic value. *Cognitive neuroscience*, 6(4): 187–214.

Friston, K. J.; Daunizeau, J.; and Kiebel, S. J. 2009. Reinforcement learning or active inference? *PloS one*, 4(7): e6421.

Friston, K. J.; Lin, M.; Frith, C. D.; Pezzulo, G.; Hobson, J. A.; and Ondobaka, S. 2017b. Active inference, curiosity and insight. *Neural computation*, 29(10): 2633–2683.

Friston, K. J.; Rosch, R.; Parr, T.; Price, C.; and Bowman, H. 2018. Deep temporal models and active inference. *Neuroscience & Biobehavioral Reviews*, 90: 486–501.

Guerguiev, J.; Lillicrap, T. P.; and Richards, B. A. 2017. Towards deep learning with segregated dendrites. *ELife*, 6: e22901.

Ha, D.; and Schmidhuber, J. 2018. Recurrent world models facilitate policy evolution. *arXiv preprint arXiv:1809.01999*.

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Heidrich-Meisner, V.; and Igel, C. 2009. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4): 152–168. Special Issue: Reinforcement Learning.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lake, B. M.; Ullman, T. D.; Tenenbaum, J. B.; and Gershman, S. J. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Lee, D.-H.; Zhang, S.; Fischer, A.; and Bengio, Y. 2015. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 498–515. Springer.

Lillicrap, T. P.; Cownden, D.; Tweed, D. B.; and Akerman, C. J. 2016. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7: 13276.

Manchev, N.; and Spratling, M. W. 2020. Target Propagation in Recurrent Neural Networks. *Journal of Machine Learning Research*, 21(7): 1–33.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Movellan, J. R. 1991. Contrastive Hebbian learning in the continuous Hopfield model. In *Connectionist Models*, 10–17. Elsevier.

Najarro, E.; and Risi, S. 2020. Meta-learning through hebbian plasticity in random networks. *arXiv preprint arXiv:2007.02686*.

Niv, Y. 2009. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3): 139–154.

O'Reilly, R. C. 1996. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5): 895–938.

Ororbia, A.; and Kifer, D. 2020. The Neural Coding Framework for Learning Generative Models. *arXiv preprint arXiv:2012.03405*.

Ororbia, A.; Mali, A.; Giles, C. L.; and Kifer, D. 2020a. Continual learning of recurrent neural networks by locally aligning distributed representations. *IEEE Transactions on Neural Networks and Learning Systems*.

Ororbia, A.; Mali, A.; Kifer, D.; and Giles, C. L. 2020b. Large-Scale Gradient-Free Deep Learning with Recursive Local Representation Alignment. *arXiv e-prints*, arXiv–2002.

Ororbia, A. G.; Haffner, P.; Reitter, D.; and Giles, C. L. 2017. Learning to Adapt by Minimizing Discrepancy. *arXiv preprint arXiv:1711.11542*.

Ororbia, A. G.; and Mali, A. 2019. Biologically motivated algorithms for propagating local target representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4651–4658.

Oudeyer, P.-Y. 2018. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546*.

O'Neill, J.; Pleydell-Bouverie, B.; Dupret, D.; and Csicsvari, J. 2010. Play it again: reactivation of waking experience and memory. *Trends in neurosciences*, 33(5): 220–229.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, 2778–2787. PMLR.

Rao, R. P.; and Ballard, D. H. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1).

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088): 533–536.

Scellier, B.; and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11: 24.

Sekar, R.; Rybkin, O.; Daniilidis, K.; Abbeel, P.; Hafner, D.; and Pathak, D. 2020. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, 8583–8592. PMLR.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Solway, A.; and Botvinick, M. M. 2012. Goal-directed decision making as probabilistic inference: a computational framework and potential neural correlates. *Psychological review*, 119(1): 120.

Spielberger, C. D.; and Starr, L. M. 1994. Curiosity and exploratory behavior. *Motivation: Theory and research*, 221–243.

Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J. 2017. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *CoRR*, abs/1712.06567.

Sutton, R. S. 1990. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, 216–224. Elsevier.

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Tschantz, A.; Baltieri, M.; Seth, A. K.; and Buckley, C. L. 2020a. Scaling active inference. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Tschantz, A.; Millidge, B.; Seth, A. K.; and Buckley, C. L. 2020b. Reinforcement learning through active inference. *arXiv preprint arXiv:2002.12636*.

Tschantz, A.; Seth, A. K.; and Buckley, C. L. 2020. Learning action-oriented models through active inference. *PLoS computational biology*, 16(4): e1007805.

Ueltzhöffer, K. 2018. Deep active inference. *Biological cybernetics*, 112(6): 547–573.

Whittington, J. C.; and Bogacz, R. 2017. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5): 1229–1262.

Wiseman, S.; Chopra, S.; Ranzato, M.; Szlam, A.; Sun, R.; Chintala, S.; and Vasilache, N. 2017. Training language models using target-propagation. *arXiv preprint arXiv:1702.04770*.

Wu, Y.; and Tian, Y. 2016. Training agent for first-person shooter game with actor-critic curriculum learning. *N/A*.

Zador, A. M. 2019. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1): 1–7.