GRADIENT-FREE REINFORCEMENT LEARNING WITH ACTIVE NEURAL GENERATIVE CODING

Alexander G. Ororbia [*]
Rochester Institute of Technology
ago@cs.rit.edu

Ankur Mali [*]
Pennsylvania State University
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 its 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 can operate even with sparse reward signals, drawing inspiration from the cognitive theory of planning as inference. We demonstrate on several reinforcement learning control problems, in the online learning setting, that our proposed modeling framework performs competitively with deep Q-learning models. The robust performance of our agents offers promising evidence that a backprop-free approach for neural inference and learning can drive goal-directed behavior.

Keywords Predictive processing · reinforcement learning · planning as inference · neural generative coding

1 Introduction

Manipulating one's environment in the effort to understand it is an essential ingredient of learning in humans [72, 6]. 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, abundant 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 [46, 71]. Planning-as-inference (PAI) [7] 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 [18, 79], 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) [38, 68], most models require exorbitant quantities of data to train well, struggling to learn tasks as efficiently as humans and animals [2]. As a result, a key challenge is how to design RL methods that successfully resolve environmental uncertainty and complexity given limited resources and data.

It has been argued that developing approaches that progressively build a useful model of the agent's world, i.e., a world [24] or dynamics model [75], would greatly reduce the sample inefficiency that plagues current RL, particularly since most modern state-of-the-art approaches are model-free - they directly attempt to learn a policy from immediate interactions with the environment, i.e., associative/habitual learning. Crucially, a generative model would facilitate more intelligent and efficient exploration of large and complex search spaces, which is critical in

[*]Both authors contributed equally.

scenarios where the reward signal is sparse, difficult to learn from, and is often problem-specific (which hinders the design of more general-purpose agents). Furthermore, a world model would be useful for designing mechanisms that facilitate planning over long temporal horizons such as in the Dyna-Q setup [75]. Thus, the active inference framework provides a promising path towards powerful, model-based RL [78]. Nonetheless, learning a world model is not easy – if the dynamics of the agent's world are complicated then a complex generative model will be needed, requiring more data and thus worsening sample efficiency (the very problem we want to solve). Therefore, the generative model employed must learn quickly and online if any benefit is to be realized.

Although PAI and active inference offer an excellent story for biological system behavior and a promising model-based RL framework, most computational implementations are formulated with explainability in mind (favoring meaningfully labeled albeit low-dimensional, discrete state/action spaces) and in the form of complex probabilistic graphical models that do not scale easily [19, 16, 21, 22]. In response, effort has been made to scale active inference by using deep ANNs [80, 77] trained by the popular backpropagation of errors (backprop) [62]. 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 [31, 87]. Importantly, from a practical point-of-view, they also suffer from practical issues related to their backprop-centric design [52]. This raises the question: can a biologically-motivated alternative to backprop 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 will develop one possible alternative that positively answers this question. As a result, 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 [46].

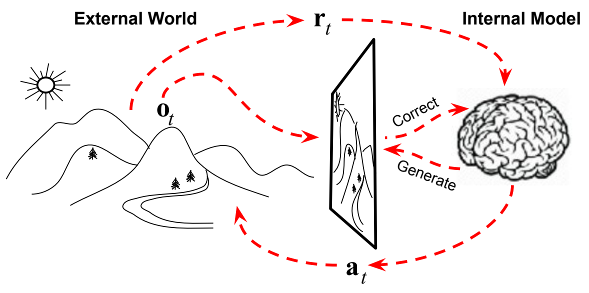


Figure 1: Schematic representation of the intuition behind an ANGC model. The agent continually predicts the state of the world, acts to manipulate it, and then corrects its internal model given observations and reward signals.

While many backprop-alternative (gradient-free) algorithms have recently been proposed [44, 47, 32, 35, 64, 23, 83, 52], few have been investigated outside the context of supervised learning, with some notable exceptions in sequence [85, 49, 36] and generative modeling [48]. In the realm of RL, aside from neuro-evolutionary approaches [74, 27], the dearth of work is more prescient and it is our intent 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 proposed agent system, we demonstrate how a scalable, biologically-plausible inference and learning process, grounded in the neuro-biologically theory of predictive processing [14, 12], 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 even in settings with sparse rewards which other gradient-free methods such as neuroevolution [74, 27] struggle with.[2] To evaluate the efficacy of ANGC, we implement an agent structure that is tasked with solving several control problems often experimented with in RL and compare performance against deep Q-learning, a popular backprop-based approach.

2 Active Neural Generative Coding

To specify our proposed ANGC agent, the high-level intuition of which is illustrated in Figure 1, we start by first defining the fundamental building block used to construct it – the neural generative coding circuit.

2.1 The Neural Generative Coding Circuit

Neural generative coding (NGC) is a recently developed framework [48] that generalizes classical ideas in predictive processing [59, 12] to the construction of scalable neural models that model and predict both static and temporal patterns [49, 48]. An NGC model is composed of L layers of stateful neurons, where the activity values of each layer $\ell = \{0, 1, \cdots, L\}$ is represented by the vector $\mathbf{z}^\ell \in \mathcal{R}^{J_\ell \times 1}$ (any layer contains J_ℓ neurons emitting activity signals), that are engaged in a process of never-ending guess-then-correct. Generally, an NGC model's bottom-most

[2]This is due to the fact that it is difficult to determine a strong encoding solution/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.

Algorithm 1 The NGC model projections, inference, and weight update routines.

```
Input: Sample $(\mathbf{x}^i, \mathbf{x}^o)$ (from sensory stream), $\beta$, $\beta_e$, $\gamma_v$, $\gamma_e$, $\eta$, $K$, $\Theta$, choice of $\phi^\ell$ & $g_\ell$, and $c_\epsilon = 10^{-6}$
function PROJECT($\mathbf{x}^i, \Theta$)
    // Ancestrally project pattern through model
    $\bar{\mathbf{z}}^L = \mathbf{x}^i$
    for $\ell = L-1$ to 0 do
        Get $\mathbf{W}^{\ell+1}$ from $\Theta$,  $\bar{\mathbf{z}}^\ell = g_\ell(\mathbf{W}^{\ell+1} * \phi^{\ell+1}(\bar{\mathbf{z}}^{\ell+1}))$
    Return $\bar{\mathbf{z}}^0$
function INFERSTATES($\mathbf{x}^i, \mathbf{x}^o, \Theta$)
    // Simulate stimulus presentation time over $K$ steps
    $\mathbf{z}^0 = \mathbf{x}^o, \mathbf{z}^1 = \mathbf{0}, \cdots, \mathbf{z}^\ell = \mathbf{0}, \cdots, \mathbf{z}^L = \mathbf{x}^i$
    $\mathbf{e}^0 = \mathbf{z}^0 - 0, \mathbf{e}^1 = \mathbf{0}, \cdots, \mathbf{e}^\ell = \mathbf{0}, \cdots, \mathbf{e}^L = \mathbf{0}$
    for $k = 1$ to $K$ do
        // Correct latent states given current value of error units
        for $\ell = 1$ to $L$ do
            Get $\mathbf{E}^\ell$ from $\Theta$,  $\mathbf{z}^\ell \leftarrow \mathbf{z}^\ell + \beta(-\gamma_v \mathbf{z}^\ell - \mathbf{e}^\ell + (\mathbf{E}^\ell * \mathbf{e}^{\ell-1}) + \vartheta(\mathbf{z}^\ell)))$    ▷ Maps to Eqn. 2
        // Compute layer-wise predictions and error neuron values
        for $\ell = L-1$ to 0 do
            Get $\mathbf{W}^{\ell+1}$ from $\Theta$,  $\bar{\mathbf{z}}^\ell = g_\ell(\mathbf{W}^{\ell+1} * \phi^{\ell+1}(\mathbf{z}^{\ell+1}))$,  $\mathbf{e}^\ell = \frac{1}{2\beta_e}(\phi^\ell(\mathbf{z}^\ell) - \bar{\mathbf{z}}^\ell)$    ▷ Eqn. 1
    $\Lambda = \{\mathbf{z}^0, \mathbf{z}^1, \cdots, \mathbf{z}^\ell, \cdots, \mathbf{z}^L\}$, $\mathcal{E} = \{\mathbf{e}^0, \mathbf{e}^1, \cdots, \mathbf{e}^\ell, \cdots, \mathbf{e}^{L-1}\}$
    Return $\Lambda, \mathcal{E}$
function UPDATEWEIGHTS($\Lambda, \mathcal{E}, \Theta$)
    // Adjust synaptic weight adjustments given states and error neurons
    for $\ell = 1$ to $L$ do
        Retrieve $(\mathbf{W}^\ell, \mathbf{E}^\ell)$ from $\Theta$
        $\Delta\mathbf{W}^\ell = \mathbf{e}^\ell * (\phi^{\ell+1}(\mathbf{z}^\ell))^T$,  $\Delta\mathbf{W}^\ell \leftarrow \frac{\Delta\mathbf{W}^\ell}{||\Delta\mathbf{W}^\ell||_2 + c_\epsilon}$
        $\Delta\mathbf{E}^\ell = \gamma_e(\Delta\mathbf{W}^\ell)^T$,  $\Delta\mathbf{E}^\ell \leftarrow \frac{\Delta\mathbf{E}^\ell}{||\Delta\mathbf{E}^\ell||_2 + c_\epsilon}$
        $\mathbf{W}^\ell \leftarrow \mathbf{W}^\ell + \eta\Delta\mathbf{W}^\ell$,  $\mathbf{E}^\ell \leftarrow \mathbf{E}^\ell + \eta\Delta\mathbf{E}^\ell$    ▷ Can alternatively use Adam or RMSprop
        $\mathbf{W}^\ell \leftarrow \frac{2\mathbf{W}^\ell}{||\mathbf{W}^\ell||_2 + c_\epsilon}$,  $\mathbf{E}^\ell \leftarrow \frac{2\mathbf{E}^\ell}{||\mathbf{E}^\ell||_2 + c_\epsilon}$    ▷ Override current values of $(\mathbf{W}^\ell, \mathbf{E}^\ell)$ in $\Theta$
    $\widehat{\Theta} = \{\mathbf{W}^0, \mathbf{E}^0, \mathbf{W}^1, \mathbf{E}^1, \cdots, \mathbf{W}^L, \mathbf{E}^L\}$
    Return $\widehat{\Theta}$
```

layer $\mathbf{z}^0$ is clamped to a sensory pattern extracted from the environment. However, in this work, we design an NGC model that clamps both its top-most and bottom-most layers 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 vector pairs $(\mathbf{x}^i, \mathbf{x}^o)$ where $\mathbf{x}^i \in \mathcal{R}^{J_L \times 1}$ and $\mathbf{x}^o \in \mathcal{R}^{J_0 \times 1}$.

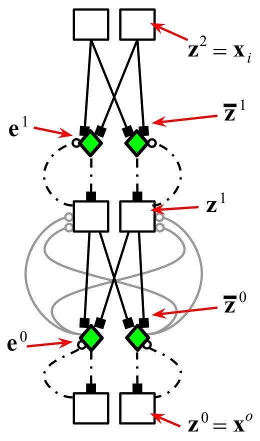
Specifically, in an NGC model, layer $\mathbf{z}^{\ell+1}$ attempts to guess the current post-activity value of layer $\phi^\ell(\mathbf{z}^\ell)$ by generating a prediction vector $\bar{\mathbf{z}}^\ell$ using a matrix of forward synaptic weights $\mathbf{W}^{\ell+1} \in \mathcal{R}^{J_\ell \times J_{\ell+1}}$. 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: $\mathbf{e}^\ell = \frac{1}{\beta_e}(\phi^\ell(\mathbf{z}^\ell) - \bar{\mathbf{z}}^\ell)$.[3] 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:

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

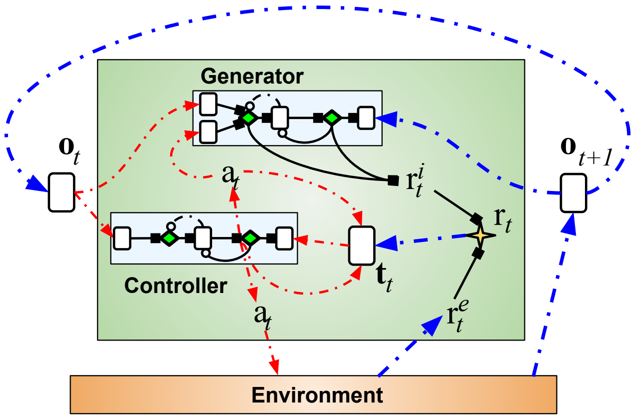
$$\mathbf{z}^{\ell+1} \leftarrow \mathbf{z}^{\ell+1} + \beta\Big(\overbrace{-\gamma_v \mathbf{z}^{\ell+1}}^{\text{leak}} \overbrace{-\mathbf{e}^{\ell+1} + (\mathbf{E}^{\ell+1} \cdot \mathbf{e}^\ell)}^{\text{top-down + bottom-up pressure}} + \overbrace{\vartheta(\mathbf{z}^{\ell+1})}^{\text{lateral term}})\Big)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

where $*$ 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 (though for the bottom layer g_0 could be set to a function such as the logistic sigmoid depending on the type of data being predicted [48]). In Equation 2, the

[3] One may replace the term $\frac{1}{2\beta_e}$ with Σ^{-1}, i.e., a learnable, lateral modulation matrix that applies precision-weighting to the error units, to recover the setup of [48]. We defer the use of precision weights for future work.



(a) An example NGC circuit.



(b) The ANGC agent architecture.

Figure 2: An NGC circuit is depicted on the left and the high-level ANGC architecture (with both an NGC controller and generator) is shown on the right. Green diamonds represent error neurons, empty rectangles represent state neurons, solid arrows represent individual synapses, dash-dotted arrows represent direct copying of information, open circles indicate excitatory signals, and filled squares/diamonds indicate inhibitory signals.

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 excitation/inhibition term, which is a function that creates different competition patterns among the neurons inside of layer ℓ [48] – 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 appropriate internal activity values $\{\mathbf{z}^1, \cdots, \mathbf{z}^{L-1}\}$ (see INFERSTATES in Algorithm 1 for how this cycle is implemented).

After the appropriate internal activities have been found, the synaptic weight matrices may be adjusted using a simple local error Hebbian rule adapted from local representation alignment (LRA) [52, 50]:

$$\Delta \mathbf{W}^\ell = \mathbf{e}^\ell * (\phi^\ell(\mathbf{z}^{\ell+1}))^T, \text{ and, } \Delta \mathbf{E}^\ell = \gamma_e (\Delta \mathbf{W}^\ell)^T$$

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). Once the updates to the NGC weight matrices have been computed, an update rule such as stochastic gradient ascent, Adam [30], or RMSprop [76] can be used (see UPDATEWEIGHTS in Algorithm 1 for an example of how gradient ascent is concretely implemented).

The online objective that an NGC model attempts to minimize is known as total discrepancy [51], from which the error neuron, state update expressions, and local synaptic adjustments may be derived [50, 48]. The total discrepancy objective, which could also be interpreted as a form of free energy [15] specialized for the case of stateful neural models that utilize arbitrary forward and error synaptic wiring pathways [50], can be expressed in many forms including the linear combination of local density functions [48] or the summation of local distance functions [49]. For this study, the form of total discrepancy we used to derive the expressions above is the linear combination of distance functions: $\mathcal{L}(\Theta) = \sum_{\ell=0}^{L-1} \mathcal{L}(\mathbf{z}^\ell, \bar{\mathbf{z}}^\ell) = \sum_{\ell=0}^{L-1} \frac{1}{2\beta_e} ||(\mathbf{z}^\ell - \bar{\mathbf{z}}^\ell)||_2^2 = \sum_{\ell=0}^{L-1} \frac{1}{2\beta_e} ||\mathbf{e}^\ell||_2^2$.

Algorithm 1 puts all of the equations and relevant details presented above together to describe the inference and learning procedures of a full NGC model that processes $(\mathbf{x}^i, \mathbf{x}^o)$ from a data stream. Notice that the algorithm breaks down the model processing into three routines – INFERSTATES($\circ$), UPDATEWEIGHTS($\circ$), and PROJECT($\circ$). INFERSTATES($\circ$) is simply the K-step described earlier to find reasonable values of the latent state activities given clamped data and UPDATEWEIGHTS($\circ$) is the complementary procedure used for adjusting the synaptic weight parameters once state activities have been found after using INFERSTATES($\circ$). 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 [48] but accepts any clamped input pattern instead of samples drawn from a prior distribution. Figure 2a graphically depicts a three layer NGC model with 2 neurons in each layer.

2.2 Generalizing to Active Neural Coding

Given the definition of the NGC building block in the above section, we now turn our attention to the generalization that incorporates actions. 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, this means that our ANGC agent will maximize the following:

$$r_t = \alpha_e r^e_t + \alpha_i r^i_t = r^{in}_t + r^{ep}_t \tag{3}$$

which is a reward signal that can be decomposed into an instrumental (or goal-oriented) signal r^{in}_t and an epistemic (or exploration/information maximizing) signal r^{ep}_t. Each component signal is controlled by 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^e_t to drive r^{ep}_t) or an external goal-directing signal (r^i_t to drive r^{in}_t).

Note that while we have chosen to interpret and represent the active inference view of the exploration-exploitation trade-off as (dopamine) scalars, our objective, notably the instrumental signal, is not limited to this scheme and could potentially incorporate an encoding of more complex functions such as (prior) distribution functions over (goal) states.[4] This generality is afforded by the complete class theorem [8, 82, 13], which says that, for any pair of reward functions (or preferences) and choice behavior, there exist some prior beliefs that render the choices Bayes optimal. Understanding the theoretical backing of the complete class theorem means that our ANGC framework, despite the fact that it commits to a particular form of neural processing (to provide a concrete implementation for simulation), could be written down in terms of Bayesian decision processes even though the form we present does not explicitly do so. In Equation 3, we further highlight our active inference formulation connects nicely with the recently popular use of extrinsic versus intrinsic reward values to facilitate "curiosity-driven learning" [53, 9].

As indicated by the architecture diagram in Figure 2b, 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^{ep}_t, and the controller, which is responsible for choosing the actions such that the full reward r_t, which includes the instrumental term r^{in}_t, is maximized.

The NGC Generator Once the generator's top-most latent state is clamped to the 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 INFERSTATES defined in Algorithm 1, 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 $[\cdot, \cdot]$ is to indicate the vector concatenation of $\mathbf{o}_t$ and a_t. Once latent activities have been found, the generator then updates its synapses via routine UPDATEWEIGHTS in Algorithm 1.

The generator plays in an important role for driving the exploration conducted by an ANGC agent. Specifically, as the generator progressively learns to how to synthesize future observations, the current activities of the 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 $r^e_t = \sum_\ell ||\mathbf{e}^\ell||^2_2$ which we observe is the result of summing across layers as well as across each error neuron vector's respective dimensions.[5] The epistemic term $r^{ep}_t = \alpha_e r^e_t$ is then combined with an instrumental term $r^{in}_t = \alpha_i r^i_t$, i.e., the scalar signal produced externally (by the environment or by another neural system), to guide the agent towards a goal state(s), according to Equation 3. The final value r_t is then subsequently used to adapt the controller described in the following sub-section.

The NGC Controller 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 any step in time, given a target scalar value (produced by the environment and the generator), the controller will also infer a suitable set of latent activities using the INFERSTATES routine defined in Algorithm 1.

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 done by the TOONEHOT function call in Algorithm 2), computing the boot-strap estimate of the

[4] If rewards are viewed as log priors, i.e., $p(\mathbf{o}) \propto \exp(r(\mathbf{o}))$ [20], then other choices of $p(\mathbf{o})$ are possible.

[5] Observe that this term is also proportional to the generator's total discrepancy, i.e., $r^e_t = \sum_\ell ||\mathbf{e}^\ell||^2_2 \propto \mathcal{L}(\Theta_g)$.

Algorithm 2 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 $\mathcal{M}$, $E$, $T$, $\alpha_e$, $\alpha_i$, $\epsilon_{decay}$, $\epsilon$, and $\gamma$
**function** SIMULATEPROCESS($\mathbb{S}, E, T, \Theta_c, \Theta_g, \mathcal{M}, \alpha_e, \alpha_i, \epsilon, \epsilon_{decay}$)
    $r^i_{max} = 1$
    **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
            $\mathbf{d}_t = \text{PROJECT}(\mathbf{o}_t, \Theta_c)$, $p \sim \mathbb{U}(0, 1)$
            $\Big(p < \epsilon \rightarrow a_t \sim \mathbb{U}_d(1, A)\Big) \wedge \Big(p \geq \epsilon \rightarrow a_t = \arg\max_a \mathbf{d}_t\Big)$,  $\mathbf{a}_t = \text{TOONEHOT}(a_t)$
            // Get next state/observation from environment & compute component reward signals
            $(r^e_t, \mathbf{o}_{t+1}) \leftarrow \mathbb{S}(a_t)$,  $(\Lambda, \mathcal{E}) = \text{INFERSTATES}([\mathbf{a}_t, \mathbf{o}_t], \mathbf{o}_{t+1}, \Theta_g)$
            $r^i_t = \sum_\ell ||\mathcal{E}[\ell]||^2_2$,  $r^i_{max} = \max(r^i_t, r^i_{max})$,  $r^i_t \leftarrow \frac{r^i_t}{r^i_{max}}$,  $r_t = \alpha_e r^e_t + \alpha_i r^i_t$
            // 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
            $t_j = \begin{cases} r_j & \text{if } \mathbf{o}_j \text{ is terminal} \\ r_j + \gamma \max_a \text{PROJECT}(\mathbf{o}_{j+1}, \Theta_c) & \text{otherwise} \end{cases}$
            $\mathbf{a}_j = \text{TOONEHOT}(a_j)$,  $\mathbf{t}_j = t_j \mathbf{a}_j + (1 - \mathbf{a}_j) \otimes \text{PROJECT}(\mathbf{o}_j, \Theta_c)$
            // Update controller $\Theta_c$
            $(\Lambda_c, \mathcal{E}_c) = \text{INFERSTATES}(\mathbf{o}_j, \mathbf{t}_j, \Theta_c)$,  $\Theta_c \leftarrow \text{UPDATEWEIGHTS}(\Lambda_c, \mathcal{E}_c, \Theta_c)$
            // Update generator $\Theta_g$
            $(\Lambda_g, \mathcal{E}_g) = \text{INFERSTATES}([\mathbf{a}_j, \mathbf{o}_j], \mathbf{o}_{j+1}, \Theta_g)$,  $\Theta_g \leftarrow \text{UPDATEWEIGHTS}(\Lambda_g, \mathcal{E}_g, \Theta_g)$
        $\epsilon \leftarrow \max(0.05, \epsilon * \epsilon_{decay})$
```

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 equation:

$$\mathbf{t}_t = t_t \mathbf{a}_t + (1 - \mathbf{a}_t) \otimes \text{PROJECT}(\mathbf{o}_t, \Theta_c) \quad (4)$$

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

$$(\mathbf{o}_t \text{ is terminal} \rightarrow t_t = r_t) \wedge (\mathbf{o}_t \text{ is not terminal} \rightarrow t_t = r_t + \gamma \max_a \text{PROJECT}(\mathbf{o}_{t+1}, \Theta_c)). \quad (5)$$

Once $\mathbf{t}_t$ has been prepared, we may run the controller to find its latent activities for $\mathbf{o}_t$ and $\mathbf{t}_t$ using INFERSTATES and calculate the local weight updates via the UPDATEWEIGHTS routine (from Algorithm 1). Furthermore, observe that the second sub-expression in Equation 5 involves re-using the controller to estimate the (reward) value of the future observation/state, i.e., the $\gamma \max_a \text{PROJECT}(\mathbf{o}_{t+1}, \Theta_c)$ term. This specific term could then 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 [39], where $\widehat{\Theta}_c$ are the parameters of a "target controller" that are initialized to be the values of Θ_c at the start of the simulation and only ever updated every C transitions 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 proposed ANGC framework prescribes the joint interaction of the controller and generator modules describe above. At each time step, the agent, given observation/state $\mathbf{o}_t \in \mathcal{R}^{D \times 1}$ (which could contain continuous or discrete variables), is to perform a discrete action a_t[6] 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 of the sensory sample drawn from the environment, allowing it iteratively craft a compressed internal impression of the agent's world. The inability of the generator to accurately predict the incoming sensory sample $\mathbf{o}_{t+1}$ will be used to guide the agent to explore its environment in a fashion that ultimately reduces its (long-term) surprisal and thus improve the controller's ability to extract an effective policy/plan to achieve a goal.

The complete ANGC agent is specified in Algorithm 2[7] and graphically depicted in Figure 2b. Note that Algorithm 2 implements the the full simulation of the agent's inference and synaptic adjustment over an E-episode long

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

[7]Note that $\mathcal{E}[\ell]$ means retrieve the ℓth item in $\mathcal{E}$.

Table 1: For each control task, below are the meta-parameter configurations used for the ANGC agents.

	Cartpole		**Mountain Car**		**Lunar Lander**	
	Controller	**Generator**	**Controller**	**Generator**	**Controller**	**Generator**
$\phi(\cdot)$	ReLU	ReLU	ReLU6	ReLU6	ReLU6	ReLU6
Lat Dim	$[256, 128]$	$[256, 128]$	$[128, 128]$	$[128, 128]$	$[512, 256]$	$[128, 128]$
Rule	RMSprop	Adam	Adam	Adam	Adam	Adam
η	0.0005	0.001	0.001	0.001	0.001	0.001
ϵ_{decay}	0.97	–	0.95	–	0.995	–
C	100	–	200	–	200	–
γ	0.99	–	0.99	–	0.99	–
N_{mem}	10^6	10^6	$500,000$	$500,000$	$500,000$	$500,000$
N_{batch}	256	256	128	128	256	256

stream (where each episode is at most T steps long – note that T can vary with time as in the case of episode streams). In addition to the target controller modification described in the last sub-section, we also integrate a simple form of experience replay memory $\mathcal{M}$ [56, 39] (implemented as a ring buffer where mini-batches used for training are created by sampling stored transitions uniformly at random). This stabilizes the learning process by removing correlations in the observation sequence encountered by the ANGC agent.

3 Experiments

The performance of our proposed ANGC agent is evaluated on three classical control problems commonly used in reinforcement learning (RL). Specifically, we compare our ANGC to a random agent (where actions are taken at each step in time uniformly at random) and a deep Q-network (DQN) [39] on: 1) the inverted pendulum (cartpole) problem, 2) the mountain car problem, and 3) the lunar lander problem.

3.1 Control Tasks

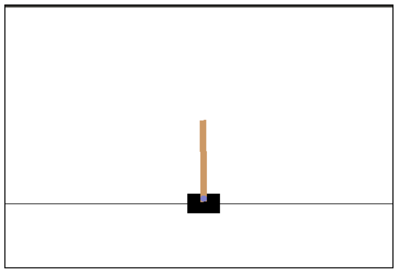
The Inverted Pendulum Problem: In the inverted pendulum problem, also known as the cart-pole problem, the goal is to keep a pole balanced upright atop a movable cart. The state space of this problem is summarized in a 4-D vector $\mathbf{o}_t$ comprised of four values – cart position, cart velocity, the angle θ of the pole, and the angular velocity. The agent can choose between two discrete actions – move the cart to the left (with a fixed force value) or move the cart to the right. The reward function for this problem yields a value of 1.0 for every time step that the episode does not end (maximum $T = 500$). An episode ends when the angle of the pole is more than $15°$ from vertical or when the base of the cart moves than 2.4 units from the center of the problem space. This task is considered solved when an agent obtains an average cumulative reward of 475.0 over the past 100 episodes.

The Mountain Car Problem: In this problem [43], a car is located at the bottom of a (1-D) valley and the agent's goal is to drive the car and park at the top of the hill (to the right). The target location is marked by a flag-post at the x-coordinate value 0.5. The state space is a 2-D vector comprised of two values – the position of the car (along the x-axis) and the car's velocity. The agent chooses one of three discrete actions – accelerate the car to the left, accelerate the car to the right, or do nothing. A reward of -1.0 is given to the agent for every step that the episode does not end (maximum $T = 200$) – this encourages agents to find a way to the hilltop quickly. This task is considered solved when an agent obtains an average of -110.0 cumulative reward over the past 100 episodes.

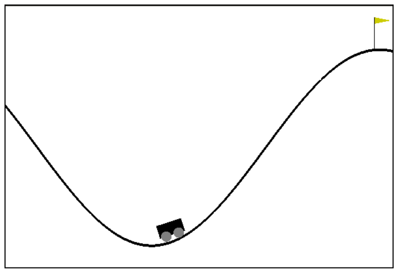
The Lunar Lander Problem: In the lunar lander problem, an agent is tasked with landing a rocket onto a landing pad located at the (2-D) coordinate $(0, 0)$. The state space for this problem is an 8-D vector. The agent can choose between four possible discrete actions – fire the left engine, fire the right engine, fire the rocket upwards, or do nothing. The agent receives a reward of 100 for landing on the pad, 10 points for each of the rocket legs that are standing, and -0.3 for every time step the rocket's engines are firing. This task is considered solved when an agent obtains an average of 200.0 cumulative reward over the past 100 episodes.

3.2 Training Setup

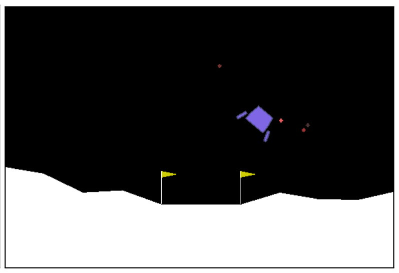
ANGC Agent Setup: For all of the ANGC agents across all trials, we used fixed configurations of meta-parameters for each control task. In Table 1, we present several of the key values chosen (based on preliminary experimentation) for the meta-parameters of each of the two sub-models of the agent, i.e., the controller (Controller) and the generative model (Generator). Initial synaptic weight values for both were initialized by sampling a



(a) Inverted pendulum (cart-pole).



(b) Mountain car.



(c) Lunar lander.

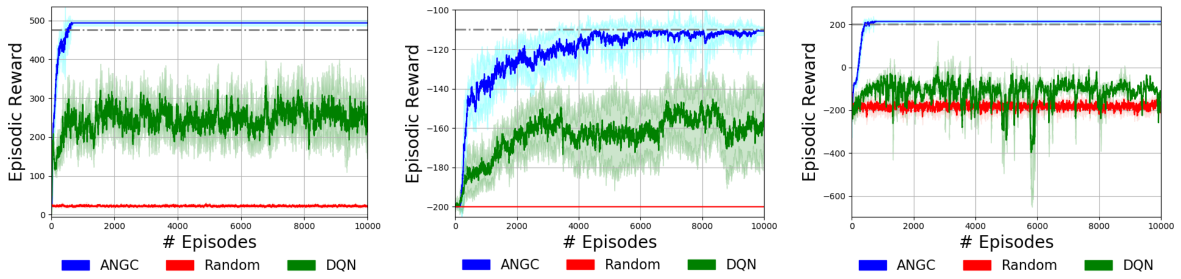


Figure 3: Reward curves for ANGC, the DQN, and random baselines. Mean and standard deviation over 10 trials have been plotted. Dash-dotted, horizontal (gray) lines depict the threshold for problem solution.

centered Gaussian with standard deviation of 0.025. In the table, "Lat Dim" refers to the number of neurons in each latent layer of the model, e.g., $[128, 128]$ means two internal layers of 128 neurons were used, and "Rule" refers to the choice of the weight update rule used, such as RMSprop [76] or Adam [30].

For all ANGC agents, $\alpha_e = \alpha_i = 1.0$ was use to set the importance factors of both the epistemic and instrumental signals. Both the controller and generative model of each agent was trained using a single, shared experience replay buffer with a maximum capacity of N_{mem} transitions from which mini-batches of N_{batch} transition samples were sampled in order to compute parameter updates at any single time-step of each simulation. Each agent made use of 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 the very start of each trial).

DQN Agent Setup: For the DQN 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 ($\epsilon = 0.1$). A discount factor of $\gamma = 0.99$ was used for all DQNs. The linear rectifier was used as the activation function and Adam was used to update the weight values, which 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 global learning rate, the hidden dimensions, and number of layers were tuned – hidden layer sizes were selected from within the range of $[32, 256]$ and the number of layers was chosen from within the range of $[1, 3]$.

3.3 Results

In Figure 3, we visualize the accumulated reward as a function of the episode count, 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 trial mean (darker color central curve) and standard deviation (lighter color envelope). In each plot, a dash-dotted horizontal indicates the threshold for fully solving each task.

It is immediately apparent from our reward plots, across all three control benchmarks, that the ANGC agent is not only competitive with backprop-based, deep Q-learning but it is able to learn a good policy from the simulated episodes far earlier. This highlights the value of our ANGC framework for designing agents – the sample efficiency improves for each of the benchmarks we investigate given that far fewer episodes are required by the ANGC agents to generalize well and even ultimately formally solve the control problem (as indicated by their ability to reach each problem's solution threshold).

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 problem, which is arguably the hardest of the problems examined in this study. The ANGC's effectiveness on this problem is, we argue, largely due to its own internally generated epistemic modulation factor r^{ep}_t. In other words, the ANGC agent is driven to explore states that surprise it most, meaning it is most "curious" about states that yield the highest magnitude total discrepancy (or, indirectly, the greatest free energy). This feature presents a clean, neural generative coding implementation of the epistemic/surprisal term key to the active inference framework [21] which theoretically is meant to encourage a more principled and efficient form of environmental exploration. Furthermore, this term, much akin to the intrinsic reward produced in curiosity-driven models [58], allows the agent to operate in settings where there even exists no obvious, external reward (or instrumental term).

While our ANGC agent results are promising, integrating additional mechanisms typically used in deep RL would be a fruitful next step. Since our agent framework has already proven to be compatible with commonly-used RL heuristics such as experience replay and target network stability, integrating other heuristics would help to further improve performance.

3.4 Discussion

First and foremost, the fundamental building blocks of our ANGC agent align with the plethora of predictive processing computational models that have been proposed to explain brain function [59, 14, 4]. This provides a desirable grounding of our model in computational cognitive neuroscience by connecting it to a prominent Bayesian brain theory as well as with established general principles of neurophysiology (for example, it combines diffuse inhibitory feedback connections with driving feedforward excitatory connections). In addition, the synaptic weight adjustments computed in our framework are local, corresponding to, if we include the factor $1/2\beta_e$ (and ideally replace it with the learnable precision matrix of [48], a three-factor (error) Hebbian update rule. While there are many elements of our NGC building block that preclude it from serving as a complete and proper computational model of actual neural circuitry, e.g., synapses are currently allowed to be negative and positive, neurons are communicating with real-values instead of spikes [81], etc., it represents a step forward towards a computational framework that facilitates a scalable simulation of neuro-biologically plausible computation that generalizes well on complicated statistical learning problems. We also note that our implementation further makes several design choices for the sake of computational speed and therefore only represents one possible implementation of predictive processing generalized for action-driven adaptation to dynamic environments. However, despite these limitations, our ANGC framework offers a promising path towards viable neural agents that learn without backprop, no longer subject to the algorithm's particular constraints [5]. Practically, one could leverage the parallelism afforded by high performance computing to potentially simulate very large, non-differentiable neural circuits given that our NGC modules offer natural layer-wise parallel calculations and do not suffer from the forward and backward-locking problems that plague backprop [29]. Furthermore, our ANGC contributes to the effort to create more biologically-motivated models and update rules either based on or for reinforcement learning [37, 1, 86] and motor control.

Second, key to our overall ANGC agent design is the notion of novelty or surprise [3], which is what provides our neural system with a means to explore an environment beyond a uniform random sampling scheme. This connects our ANGC framework to the family of RL models that have been designed to generate intrinsic reward signals [65, 73, 70, 54], which are inspired by the psychological concept of curiosity inherent to human agents [63, 69]. Crucially, curiosity provides an agent with the means to acquire new skills that might prove useful for maximizing rewards in downstream tasks – as a result, one could view the total discrepancy signal produced by the ANGC's generator (to create r^{ep}_t) as one potential source of "curiosity" that drives the agent, even in the presence of nearly no external reward signals (as was the case in our mountain car experiment). Note that there are many other forms of intrinsic reward signals such as those based on policy-entropy [60, 25], information gain [28, 67], prediction error [58, 10], state entropy [33], state uncertainty [55], and empowerment [34, 40]. Since our agent employs and adapts a dynamic generative model to produce the necessary signals to drive its curiosity, our agent also contributes to the work in improving model-based RL through the use of world models [24, 45, 11, 26]. However, world models in modern-day, model-based RL are learned with backprop whereas our ANGC agent uses the same parallel neural processing and gradient-free weight updating as the controller, further obviating the need for learning the entire system in pieces or using evolution to learn an action model [24]. Notably, generative world models could be useful when integrated into powerful planning algorithms [61, 84]. Extending and using our ANGC's generator for long-term planning will be the subject of future work.

Finally, our ANGC agent framework offers a simple predictive processing interpretation of active inference and the more general, theoretical planning-as-inference. Planning-as-inference (PAI) [7] broadly encapsulates the view that a decision-making agent utilizes an internal cognitive model that represents the agent's future as the joint probability distribution over actions, (outcome) states, and rewards. Active inference [21], which is a particular instantiation of PAI, suggests that agents select actions such that they maximise the evidence for an internal model that is also biased towards the agent's preferences. In Equation 3, we took advantage of the complete class theorem [8] to develop a simple weighted sum of the two signals central to the general active inference optimization objective – an instrumental term (which drives the agent towards a goal or preferred state of the world) and an epistemic term (which drives the agent to search through its environment by attending to states that surprise it the most). While we focus on scalar signals, which would likely be components of the error function that is embodied in the firing rates of dopamine neurons [42, 41, 66], one could instead use encodings of goal states, priors, or other functions that could facilitate more complex behavior and longer-term planning in an ANGC agent. Neural process theory has also been developed for active inference [17, 57] which could be used to further modify our framework towards greater neurobiological plausibility.

4 Conclusion

In this paper, we proposed active neural generative coding (ANGC), a framework for learning goal-directed agents without backpropagation of errors. We demonstrated on three control problems in reinforcement learning the effectiveness of our agent framework for learning models that are competitive with popular backprop-based ones, such as the deep Q-network. 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 intrinsic signals (based on total discrepancy) to augment problem-specific extrinsic rewards. Furthermore, given its better sample efficiency as demonstrated by our experiments, the ANGC framework could prove useful in more complex environments, offering an important means of implementing planning-as-inference.

References

[1] ALEXANDER, W. H., AND BROWN, J. W. Frontal cortex function as derived from hierarchical predictive coding. *Scientific reports 8*, 1 (2018), 1–11.

[2] ARULKUMARAN, K., DEISENROTH, M. P., BRUNDAGE, M., AND BHARATH, A. A. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

[3] BARTO, A., MIROLLI, M., AND BALDASSARRE, G. Novelty or surprise? *Frontiers in psychology 4* (2013), 907.

[4] BASTOS, A. M., USREY, W. M., ADAMS, R. A., MANGUN, G. R., FRIES, P., AND FRISTON, K. J. Canonical microcircuits for predictive coding. *Neuron 76*, 4 (2012), 695–711.

[5] BENGIO, Y., LEE, D.-H., BORNSCHEIN, J., MESNARD, T., AND LIN, Z. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156* (2015).

[6] BERLYNE, D. E. Curiosity and exploration. *Science 153*, 3731 (1966), 25–33.

[7] BOTVINICK, M., AND TOUSSAINT, M. Planning as inference. *Trends in cognitive sciences 16*, 10 (2012), 485–488.

[8] BROWN, L. D. A complete class theorem for statistical problems with finite sample spaces. *The Annals of Statistics* (1981), 1289–1300.

[9] BURDA, Y., EDWARDS, H., PATHAK, D., STORKEY, A., DARRELL, T., AND EFROS, A. A. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355* (2018).

[10] BURDA, Y., EDWARDS, H., STORKEY, A., AND KLIMOV, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894* (2018).

[11] CHUA, K., CALANDRA, R., MCALLISTER, R., AND LEVINE, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114* (2018).

[12] CLARK, A. *Surfing uncertainty: Prediction, action, and the embodied mind.* Oxford University Press, 2015.

[13] DAUNIZEAU, J., DEN OUDEN, H. E., PESSIGLIONE, M., KIEBEL, S. J., STEPHAN, K. E., AND FRISTON, K. J. Observing the observer (i): meta-bayesian models of learning and decision-making. *PloS one 5*, 12 (2010), e15554.

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

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

[16] FRISTON, K., FITZGERALD, T., RIGOLI, F., SCHWARTENBECK, P., AND PEZZULO, G. Active inference: a process theory. *Neural computation 29*, 1 (2017), 1–49.

[17] FRISTON, K., AND KIEBEL, S. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences 364*, 1521 (2009), 1211–1221.

[18] FRISTON, K., MATTOUT, J., AND KILNER, J. Action understanding and active inference. *Biological cybernetics 104*, 1 (2011), 137–160.

[19] FRISTON, K., RIGOLI, F., OGNIBENE, D., MATHYS, C., FITZGERALD, T., AND PEZZULO, G. Active inference and epistemic value. *Cognitive neuroscience 6*, 4 (2015), 187–214.

[20] FRISTON, K., SAMOTHRAKIS, S., AND MONTAGUE, R. Active inference and agency: optimal control without cost functions. *Biological cybernetics 106*, 8 (2012), 523–541.

[21] FRISTON, K. J., LIN, M., FRITH, C. D., PEZZULO, G., HOBSON, J. A., AND ONDOBAKA, S. Active inference, curiosity and insight. *Neural computation 29*, 10 (2017), 2633–2683.

[22] FRISTON, K. J., ROSCH, R., PARR, T., PRICE, C., AND BOWMAN, H. Deep temporal models and active inference. *Neuroscience & Biobehavioral Reviews 90* (2018), 486–501.

[23] GUERGUIEV, J., LILLICRAP, T. P., AND RICHARDS, B. A. Towards deep learning with segregated dendrites. *ELife 6* (2017), e22901.

[24] HA, D., AND SCHMIDHUBER, J. Recurrent world models facilitate policy evolution. *arXiv preprint arXiv:1809.01999* (2018).

[25] HAARNOJA, T., ZHOU, A., ABBEEL, P., AND LEVINE, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning* (2018), PMLR, pp. 1861–1870.

[26] HAFNER, D., LILLICRAP, T., FISCHER, I., VILLEGAS, R., HA, D., LEE, H., AND DAVIDSON, J. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning* (2019), PMLR, pp. 2555–2565.

[27] HEIDRICH-MEISNER, V., AND IGEL, C. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms 64*, 4 (2009), 152–168. Special Issue: Reinforcement Learning.

[28] HOUTHOOFT, R., CHEN, X., DUAN, Y., SCHULMAN, J., DE TURCK, F., AND ABBEEL, P. Vime: Variational information maximizing exploration. *arXiv preprint arXiv:1605.09674* (2016).

[29] JADERBERG, M., CZARNECKI, W. M., OSINDERO, S., VINYALS, O., GRAVES, A., SILVER, D., AND KAVUKCUOGLU, K. Decoupled neural interfaces using synthetic gradients. In *International Conference on Machine Learning* (2017), PMLR, pp. 1627–1635.

[30] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[31] LAKE, B. M., ULLMAN, T. D., TENENBAUM, J. B., AND GERSHMAN, S. J. Building machines that learn and think like people. *Behavioral and brain sciences 40* (2017).

[32] LEE, D.-H., ZHANG, S., FISCHER, A., AND BENGIO, Y. Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2015), Springer, pp. 498–515.

[33] LEE, L., EYSENBACH, B., PARISOTTO, E., XING, E., LEVINE, S., AND SALAKHUTDINOV, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274* (2019).

[34] LEIBFRIED, F., PASCUAL-DIAZ, S., AND GRAU-MOYA, J. A unified bellman optimality principle combining reward maximization and empowerment. *arXiv preprint arXiv:1907.12392* (2019).

[35] LILLICRAP, T. P., COWNDEN, D., TWEED, D. B., AND AKERMAN, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications 7* (2016), 13276.

[36] MANCHEV, N., AND SPRATLING, M. W. Target propagation in recurrent neural networks. *Journal of Machine Learning Research 21*, 7 (2020), 1–33.

[37] MAZZONI, P., ANDERSEN, R. A., AND JORDAN, M. I. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences 88*, 10 (1991), 4433–4437.

[38] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[39] 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. Human-level control through deep reinforcement learning. *nature 518*, 7540 (2015), 529–533.

[40] Mohamed, S., and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. *arXiv preprint arXiv:1509.08731* (2015).

[41] Montague, P. R., Dayan, P., and Sejnowski, T. J. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of neuroscience 16*, 5 (1996), 1936–1947.

[42] Montague, P. R., and Sejnowski, T. J. The predictive brain: temporal coincidence and temporal order in synaptic learning mechanisms. *Learning & Memory 1*, 1 (1994), 1–33.

[43] Moore, A. W. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Machine Learning Proceedings 1991*. Elsevier, 1991, pp. 333–337.

[44] Movellan, J. R. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist Models*. Elsevier, 1991, pp. 10–17.

[45] Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), IEEE, pp. 7559–7566.

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

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

[48] Ororbia, A., and Kifer, D. The neural coding framework for learning generative models. *arXiv preprint arXiv:2012.03405* (2020).

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

[50] Ororbia, A., Mali, A., Kifer, D., and Giles, C. L. Large-scale gradient-free deep learning with recursive local representation alignment. *arXiv e-prints* (2020), arXiv–2002.

[51] Ororbia, A. G., Haffner, P., Reitter, D., and Giles, C. L. Learning to adapt by minimizing discrepancy. *arXiv preprint arXiv:1711.11542* (2017).

[52] Ororbia, A. G., and Mali, A. Biologically motivated algorithms for propagating local target representations. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 4651–4658.

[53] Oudeyer, P.-Y. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546* (2018).

[54] Oudeyer, P.-Y., and Kaplan, F. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics 1* (2009), 6.

[55] O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty bellman equation and exploration. In *International Conference on Machine Learning* (2018), pp. 3836–3845.

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

[57] Parr, T., and Friston, K. J. The active construction of the visual world. *Neuropsychologia 104* (2017), 92–101.

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

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

[60] Rawlik, K. C. *On probabilistic inference approaches to stochastic optimal control*. PhD thesis, The University of Edinburgh, 2013. Edinburgh, Scotland.

[61] Rubinstein, R. Y. Optimization of computer simulation models with rare events. *European Journal of Operational Research 99*, 1 (1997), 89–112.

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

[63] RYAN, R. M., AND DECI, E. L. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology 25*, 1 (2000), 54–67.

[64] SCELLIER, B., AND BENGIO, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience 11* (2017), 24.

[65] SCHMIDHUBER, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats* (1991), pp. 222–227.

[66] SCHULTZ, W., DAYAN, P., AND MONTAGUE, P. R. A neural substrate of prediction and reward. *Science 275*, 5306 (1997), 1593–1599.

[67] SHYAM, P., JAŚKOWSKI, W., AND GOMEZ, F. Model-based active exploration. In *International Conference on Machine Learning* (2019), PMLR, pp. 5779–5788.

[68] SILVER, D., HUBERT, T., SCHRITTWIESER, J., ANTONOGLOU, I., LAI, M., GUEZ, A., LANCTOT, M., SIFRE, L., KUMARAN, D., GRAEPEL, T., ET AL. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science 362*, 6419 (2018), 1140–1144.

[69] SILVIA, P. J. Curiosity and motivation. *The Oxford handbook of human motivation* (2012), 157–166.

[70] SINGH, S. P., BARTO, A. G., AND CHENTANEZ, N. Intrinsically motivated reinforcement learning. In *NIPS* (2004).

[71] SOLWAY, A., AND BOTVINICK, M. M. Goal-directed decision making as probabilistic inference: a computational framework and potential neural correlates. *Psychological review 119*, 1 (2012), 120.

[72] SPIELBERGER, C. D., AND STARR, L. M. Curiosity and exploratory behavior. *Motivation: Theory and research* (1994), 221–243.

[73] STORCK, J., HOCHREITER, S., AND SCHMIDHUBER, J. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris* (1995), vol. 2, Citeseer, pp. 159–164.

[74] SUCH, F. P., MADHAVAN, V., CONTI, E., LEHMAN, J., STANLEY, K. O., AND CLUNE, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *CoRR abs/1712.06567* (2017).

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

[76] TIELEMAN, T., AND HINTON, G. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[77] TSCHANTZ, A., BALTIERI, M., SETH, A. K., AND BUCKLEY, C. L. Scaling active inference. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), IEEE, pp. 1–8.

[78] TSCHANTZ, A., MILLIDGE, B., SETH, A. K., AND BUCKLEY, C. L. Reinforcement learning through active inference. *arXiv preprint arXiv:2002.12636* (2020).

[79] TSCHANTZ, A., SETH, A. K., AND BUCKLEY, C. L. Learning action-oriented models through active inference. *PLoS computational biology 16*, 4 (2020), e1007805.

[80] UELTZHÖFFER, K. Deep active inference. *Biological cybernetics 112*, 6 (2018), 547–573.

[81] WACONGNE, C., CHANGEUX, J.-P., AND DEHAENE, S. A neuronal model of predictive coding accounting for the mismatch negativity. *Journal of Neuroscience 32*, 11 (2012), 3665–3678.

[82] WALD, A. An essentially complete class of admissible decision functions. *The Annals of Mathematical Statistics* (1947), 549–555.

[83] WHITTINGTON, J. C., AND BOGACZ, R. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation 29*, 5 (2017), 1229–1262.

[84] WILLIAMS, G., DREWS, P., GOLDFAIN, B., REHG, J. M., AND THEODOROU, E. A. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), IEEE, pp. 1433–1440.

[85] WISEMAN, S., CHOPRA, S., RANZATO, M., SZLAM, A., SUN, R., CHINTALA, S., AND VASILACHE, N. Training language models using target-propagation. *arXiv preprint arXiv:1702.04770* (2017).

[86] YAMAKAWA, H. Attentional reinforcement learning in the brain. *New Generation Computing 38*, 1 (2020), 49–64.

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