2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Active Predictive Coding: Brain-Inspired Reinforcement Learning for Sparse Reward Robotic Control Problems

Alexander Ororbia[1] and Ankur Mali[2]

2023 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-2365-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRA48891.2023.10160530

Abstract— **In this article, we propose a backpropagation-free approach to robotic control through the neuro-cognitive computational framework of neural generative coding (NGC), designing an agent completely built from predictive processing circuits that facilitate dynamic, online learning from sparse rewards, embodying the principles of planning-as-inference. Concretely, we craft an adaptive agent system, which we call active predictive coding (ActPC), that balances an internally-generated epistemic signal (meant to encourage intelligent exploration) with an internally-generated instrumental signal (meant to encourage goal-seeking behavior) to learn how to control various simulated robotic systems as well as a complex robotic arm using a realistic simulator, i.e., the Surreal Robotics Suite, for the block lifting task and the can pick-and-place problem. Notably, our results demonstrate that the proposed ActPC agent performs well in the face of sparse (extrinsic) reward signals and is competitive with or outperforms several powerful backpropagation-based reinforcement learning approaches.**

I. Introduction

One of the key goals of brain-inspired computing is to develop methods that draw inspiration from computational neuroscience and cognitive science to build agents that are able to interact with their environment intelligently. Notably, brain-inspired computational research seeks to develop intelligent systems that are capable of circumventing the current limitations of modern-day approaches [1], [2], such as deep neural networks trained by the popular backpropagation of errors (or backprop)[3]. This goal is complementary to (and, to an extent, even a precursor to aspects of) the domain of neurorobotics [4], [5], which focuses on designing robotic devices that contain control systems based on principles of animal/human nervous systems and/or brain structures, guided by the key premise that (neural) models are embodied in a body and an environment. In this work, motivated to develop an agent framework that is grounded in neurobiological credit assignment and one that is ultimately suitable for neurorobotics, we develop a biologically-plausible, dynamic neural circuit that is used to craft a complete agent that can tackle **extremely sparse reward** control problems, an issue that many robotic systems must ultimately face, much as humans and animals do in the real world.

Our brain-inspired agent framework is built on two neurocognitive theoretical foundations: predictive processing (or coding) and planning-as-inference. With respect to predictive coding, which views the brain as a type of hierarchical, pattern-creation engine [6] that engages in continual self-correction [7], we implement a stateful circuit where each of its levels/regions is implemented by clusters of neurons that attempt to predict the state of other nearby neural regions. These circuits then adjust synapses based on how different their predictions were from observed signals. This allows us to sidestep many of the key issues central to backprop, e.g., the vanishing/exploding gradient problems [8], the requirement for a long, unstable credit assignment feedback pathway [9], forward and backward locking problems [10], and the need for differentiability [11], [9]. On the other hand, planning-as-inference (PAI) [12], or active inference (AInf) [13], posits that biological organisms (e.g., animals and humans) learn a probabilistic generative model by interacting with their world, adjusting the model's internal states to better account for the evidence that is acquired over time (unifying perception, action, and learning by framing them as processes that result from approximate Bayesian inference). To this end, we develop specialized neural circuits that iteratively craft a dynamic generative model(s) of an agent's environment. These are further used to produce special signals that effectively handle the exploration-exploitation trade-off inherent to reinforcement learning (RL). Our computational agent framework – **active predictive coding (ActPC)** – combines both of the above novel elements to build a multi-circuit agent that tackles various (simulated) robotic control tasks, learning online from extremely sparse feedback.

II. Active Predictive Coding

A. Notation and Problem Setup:

In this work, $\leftarrow$ is assignment, $\odot$ is a Hadamard product, $\cdot$ is a matrix/vector multiplication (or dot product if operators are vectors of the same shape), and $\mathbf{v}^{\mathrm{T}}$ is the transpose. In the appendix, we have provided a notation table to help readers better understand the various components involved in the design of our proposed novel system. With respect to problem framing, we consider the standard RL setup for robotic control tasks: an agent interacts with environment E over discrete time-steps – at time t, given observation $\mathbf{o}_t \in \mathcal{R}^{D \times 1}$, the agent takes action $\mathbf{a}_t$ and then receives reward r_t and observation $\mathbf{o}_{t+1}$. The end goal is to acquire a policy that maximizes the expected future reward.

B. The Neural Generative Coding Base Circuit

Neural generative coding (NGC) is an instantiation of predictive processing brain theory [14], [15], [16], [17], yielding an efficient, robust form of predict-then-correct

[1]Alexander Ororbia is with the Department of Computer Science, Rochester Institute of Technology, 1 Lomb Memorial Dr, Rochester, NY 14623, USA ago@cs.rit.edu

[2]Ankur Mali is with the Department of Computer Science and Engineering, University of South Florida, 4202 E Fowler Ave, Tampa, FL 33620, USA ankurarjunmali@usf.edu

learning and inference. An NGC circuit in ActPC generally receives two sensory vectors, an input $\mathbf{x}^i \in \mathcal{R}^{I\times 1}$ (I is the input dimensionality) and an output $\mathbf{x}^o \in \mathcal{R}^{O\times 1}$ (O is the output or target dimensionality). Compactly, an NGC circuit is composed of L layers of feedforward, stateful neuronal units, i.e., layer ℓ is represented by the state vector $\mathbf{z}^\ell \in \mathcal{R}^{H_\ell \times 1}$ containing H_ℓ units. Given sensory vectors $\mathbf{x}^i$ and $\mathbf{x}^o$, the circuit first clamps the last layer $\mathbf{z}^L$ to the input, i.e, $\mathbf{z}^L = \mathbf{x}^i$, and then clamps the first layer $\mathbf{z}^0$ to the output, i.e., $\mathbf{z}^0 = \mathbf{x}^o$. Once clamped, the circuit will undergo a settling cycle where it processes the input and output vectors for K steps in time, i.e., it processes sensory signals over a stimulus window of K discrete time steps. The activities of the internal neurons (all neurons in between the clamped layers, i.e., $\ell = L-1 \dots 1$) are updated as follows:

$$\mathbf{z}^\ell \leftarrow \mathbf{z}^\ell + \beta\Big(-\gamma \mathbf{z}^\ell + (\mathbf{E}^\ell \cdot \mathbf{e}^{\ell-1}) \otimes \partial\phi^\ell(\mathbf{z}^\ell) - \mathbf{e}^\ell \Big) \quad (1)$$

where $\mathbf{E}^\ell$ is a matrix containing error synapses that pass along mismatch signals from layer $\ell-1$ to ℓ (this can be learnable or set to the scaled transpose of the predictive synaptic matrix, i.e., $\mathbf{E}^\ell = \lambda_e(\mathbf{W}^\ell)^T$). β is the neural state update coefficient and set according to $\beta = \frac{1}{\tau}$, where τ is the integration time constant in the order of milliseconds. Equation 1 (from left to right) indicates that a vector of neural activity changes, at each step within a settling cycle, according to: 1) a leak term (modulated by the factor γ), 2) the bottom-up pressure from mismatch signals in lower level neural regions, and 3) a top-down pressure from the neural region above. $\mathbf{e}^\ell \in \mathcal{R}^{H_\ell \times 1}$ represent an extra population of special neurons that are tasked entirely with calculating mismatch signals at a layer ℓ. That is, $\mathbf{e}^\ell = \mathbf{z}^\ell - \bar{\mathbf{z}}^\ell$, represents the difference between a layer's current activity and an expectation produced from another layer. Specifically, the layer-wise prediction $\bar{\mathbf{z}}^\ell$ is computed as follows:

$$\bar{\mathbf{z}}^\ell = g^\ell\big(\mathbf{W}^{\ell+1} \cdot \phi^{\ell+1}(\mathbf{z}^{\ell+1}) + \alpha_m(\mathbf{M}^{\ell+1} \cdot \mathbf{m}_t)\big) \quad (2)$$

$$\mathbf{m}_t = \Big[(\mathbf{k}_{t-(H-1)}, ..., \mathbf{k}_{t-i}, ..., \mathbf{k}_{t-1}\Big] \text{ and } \mathbf{k}_t = \mathbf{Q} \cdot \mathbf{x}_t \quad (3)$$

where $\alpha_m = 1$ (but can be set to 0 if working memory is not desired), $\mathbf{W}^{\ell+1}$ denotes the matrix of predictive/generative synapses, $\mathbf{M}^{\ell+1}$ is a matrix containing (conditional) memory synapses, and $\mathbf{Q}$ is a random synaptic projection matrix (with each element initialized from a centered Gaussian and standard deviation σ_q). $\mathbf{m}_t$ is a working memory vector that encodes a recent history of observation. $\mathbf{m}_t \in \mathcal{R}^{(IH)\times 1}$ is the concatenation of a small history of randomly projected observations (this paper sets $H = 7$, inspired by classical work in human working memory [18]). We found that generalizing an NGC circuit to operate with a small working memory improved predictive performance and learning stability for inherently time-varying problems such as those encountered in robotics (and in our simulations). $\phi^{\ell+1}$ is the activation function ($\partial\phi^{\ell+1}$ is its derivative) for state variables and g^ℓ (set as the identity) is applied to predictive outputs.

After the circuit processes an input-output pair for K steps (repeatedly applying Equation 1 K times), the synaptic matrices are adjusted with a Hebbian-like update rule:

$$\Delta\mathbf{W} = \mathbf{e}^\ell \cdot (\phi^{\ell+1}(\mathbf{z}^{\ell+1}))^T \odot \mathbf{M}_W \quad (4)$$

$$\Delta\mathbf{E} = \gamma_e(\Delta\mathbf{W})^T \odot \mathbf{M}_E \quad (5)$$

where $\gamma_e < 1$ is a factor that controls the time-scale of the error synaptic evolution (ensuring they change more slowly than the predictive ones). $\mathbf{M}_W$ and $\mathbf{M}_E$ are modulation matrices that perform synaptic scaling to ensure additional stability in the learning process [19]. Note that all circuits in ActPC are implemented according to the mechanistic process described above. We note that the state update dynamics and update rules for an NGC circuit can be derived from the global objective function that it attempts to optimize, i.e., total discrepancy [17], which approximates free energy [20] (see the appendix[1] for details and discussion).

Another important functionality of an NGC circuit is its ability to ancestrally project a vector (akin to a feedforward pass, since no settling process is required) through the underlying directed generative model – we represent this process as $f_{proj}(\mathbf{x}^i; \Theta)$. Formally, ancestrally projecting a vector $\mathbf{x}^i$ through an NGC circuit proceeds as follows:

$$\mathbf{z}^\ell = \bar{\mathbf{z}}^\ell = g^\ell(\mathbf{W}^{\ell+1} \cdot \phi^{\ell+1}(\mathbf{z}^{\ell+1})),\ \forall \ell = (L-1), ..., 0 \quad (6)$$

where $\mathbf{z}^L = \mathbf{x}^i$ (only the input/last layer is clamped to a specific vector, such as current input $\mathbf{x}^i$).

C. The NGC Policy Model

Much as is done in actor-critic/policy approaches, the role of our policy circuit is to describe the expected reward for taking an action $\mathbf{a}_t$ given observation $\mathbf{o}_t$. The policy circuit should critique the actions chosen by the agent's motor-action circuit (described in the next sub-section), allowing the agent to produce actions that maximize long-term reward. To this end, we take inspiration from deterministic policy gradients (DPG) [21] (and its deep backprop-based generalization [22]), which have proven useful for optimizing policies over continuous action spaces. Thus, we design an NGC circuit that plays the role of "actor" (which we label the motor-action model, inspired by the functionality of the human motor cortex) and another circuit that serves as the policy or "critic". We introduce further stability as is done in DPG-based approaches by incorporating a target circuit for both the actor and policy (both updated via Polyak averaging).

Our agent's policy circuit induces the following dynamics (according to the NGC formulation described previously):

$$\bar{\mathbf{z}}^2 = \mathbf{W}^3_a \cdot \mathbf{a}_t + \mathbf{W}^3_z \cdot \mathbf{z}^3_t + \mathbf{b}^2 \quad (7)$$

$$\bar{\mathbf{z}}^1 = \mathbf{W}^2 \cdot \phi(\mathbf{z}^2_t) + \mathbf{b}^1 \quad (8)$$

$$\mathbf{q}_t = \bar{\mathbf{z}}^0 = \mathbf{W}^1 \cdot \phi(\mathbf{z}^1_t) + \mathbf{b}^0 \quad (9)$$

where $\alpha_m = 0$ (we found the policy did not need working memory in early experiments and thus omit it to reduce unneeded computation) and $\mathbf{q}_t \in \mathcal{R}^{A\times 1}$ (A is the number of real-valued action outputs). Note that, for this and later

[1]Appendix can be found at: shorturl.at/bkpQR

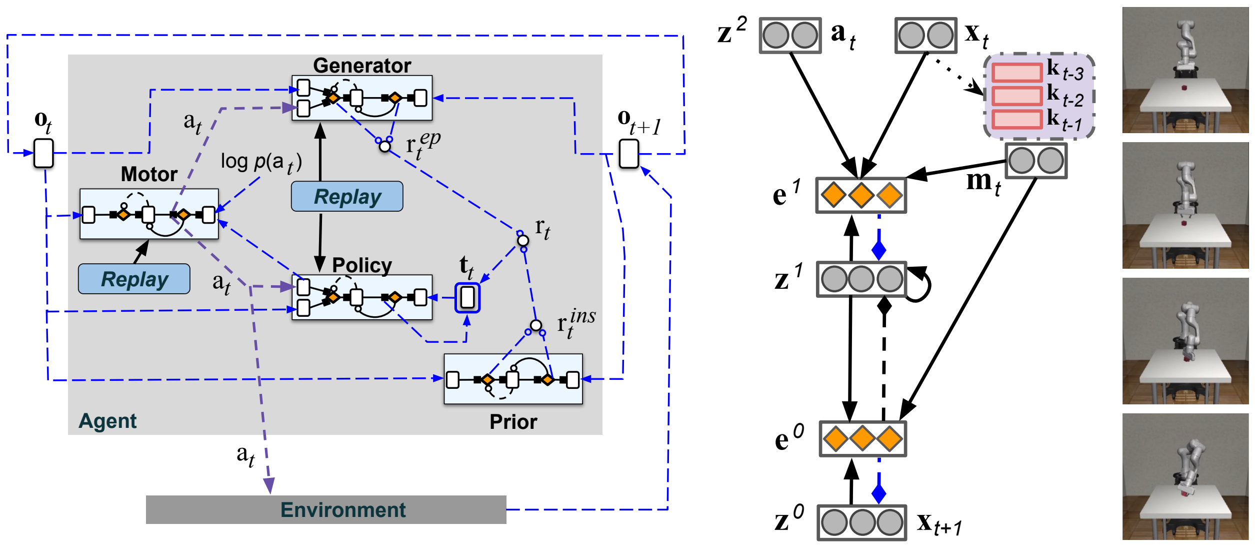

Fig. 1: (Left) Depiction of an ActPC system designed for robotic control. (Middle) A closer look at an ActPC NGC circuit, i.e., a 3-layer version of its generator/dynamics circuit coupled to working memory. Orange diamonds represent error neurons, grey circles represent state neurons, and red rounded square boxes are projected vectors stored in working memory. (Right) A trained ActPC robotic system picking up a block (for the robosuite lift task, steps visually ordered top to bottom).

described circuits, we present four-layer circuit equations for presentation simplicity. Policy parameters are stored in the construct $\Theta_c = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3_a, \mathbf{W}^3_z, \mathbf{M}^2, \mathbf{M}^3, \mathbf{b}^0, \mathbf{b}^1, \mathbf{b}^2\}$.

To update the policy model's synaptic efficacies, we leverage the reward r_t computed by the agent's epistemic/generative and instrumental/prior circuits (described in later sections). Specifically, we compute the target (action-value) vector $\mathbf{t}_t$ by first computing:

$$\mathbf{c}_t = f_{proj}([\mathbf{a}_t, \mathbf{o}_{t+1}]; \Theta_c)$$
$$\mathbf{t}_t = \mathbf{z}^0_t = \begin{cases} r_t \mathbf{1}_a & \text{if } \mathbf{o}_t \text{ is terminal} \\ r_t + \gamma \mathbf{c}_t & \text{otherwise} \end{cases} \quad (10)$$

where $\mathbf{1}_a \in 1^{A\times 1}$ (a vector of ones with the number of rows equal to the number of different continuous actions). Once the target vectors have been created, the K-step settling process (Equation 1) can be executed and all motor-action synapses are updated via Hebbian learning (Equations 4-5).

Another special modification was also made to the policy circuit, which is not apparent in the dynamics equations above. In contrast to the previous sub-section, which introduced one set of error units $\mathbf{e}^\ell$ for each layer of the circuit, our policy circuit contains two sets (or populations) of error neurons that are triggered depending on whether the policy's or actor's synapses are to be updated, i.e., $\mathbf{e}^0 = u_a \mathbf{e}^0_a + (1 - u_a)\mathbf{e}^0_p$, where $u_a = \{0, 1\}$ is a binary switch variable to toggle between adjusting policy synapses or actor synapses. To update its own synapses ($u_a = 0$), the policy model uses the same error neurons depicted in the Sub-Section II-B (with the target action-value vector). On the other hand, when the motor-action circuit's synapses are to be adjusted ($u_a = 1$), a different set of error units are utilized, i.e., $\mathbf{e}^0_a = -\mathbf{1}/A$. This is equivalent to the partial derivative of the average of the policy's actions (for $A > 1$) with respect to its output nodes. These error neurons allow the policy circuit to be temporarily linked to the actor circuit, permitting it to transmit error messages back to its input action nodes (the first term in Equation 7, i.e., $\mathbf{W}^3_a \cdot \mathbf{a}_t$) and back on through to the motor-action circuit. Since the motor-action must first traverse through the policy circuit in order to obtain an output error neuron value, this linkage is necessary (in order to approximately follow the policy gradient). It allows both circuits to run in one single joint settling process. Note that when the motor-action and policy circuits are linked, the actor's output error neurons are clamped to $\mathbf{e}^0 = \mathbf{d}_a$ where $\mathbf{d}_a = \mathbf{E}^3_a \cdot \mathbf{e}^2$ (i.e., the perturbation that the policy circuit would like to apply to error correct its input action $\mathbf{a}_t$).

D. *The NGC Motor-Action Model*

An agent must react to as well as manipulate its environment. In order to do so, the agent needs circuits to drive its actuators. Building upon the notion of planning-as-inference [12], we generalize our NGC circuit to action-driven tasks, particularly in the case of continuous actions which are critical and common in the case of robotics.

Specifically, the motor-action model $f_a\colon \mathbf{z}_t \mapsto \mathbf{a}_t$ is designed to output a continuous control vector at each time step. Thus, the external control signal is $\mathbf{a}_t \in \mathcal{R}^{A\times 1}$ where A is the number of different external actions. Action $\mathbf{a}_t$ impacts the environment that the ActPC agent is currently operating within. Formally, the motor-action model operates, when given input $\mathbf{z}^3 = \mathbf{o}_t$, as follows:

$$\bar{\mathbf{z}}^2 = \mathbf{W}^3 \cdot \mathbf{z}^3 + \mathbf{M}^3 \cdot \mathbf{m}_t + \mathbf{b}_2 \quad (11)$$
$$\bar{\mathbf{z}}^1 = \mathbf{W}^2 \cdot \phi(\mathbf{z}^2) + \mathbf{M}^2 \cdot \mathbf{m}_t + \mathbf{b}_1 \quad (12)$$
$$\mathbf{a}_t = \bar{\mathbf{z}}^0 = \kappa \tanh(\mathbf{W}^1 \cdot \phi(\mathbf{z}^1) + \mathbf{M}^1 \cdot \mathbf{m}_t + \mathbf{b}_0) \quad (13)$$

where $\tanh(\mathbf{v}) = (\exp(2\mathbf{v}) - 1)/(\exp(2\mathbf{v}) + 1)$ and κ is a coefficient that scales the values of the continuous output control signals depending on the lower and upper bounds on the allowable action values for a given problem. The parameters of the motor-action circuit are collected into the construct $\Theta_a = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \mathbf{b}^0, \mathbf{b}^1, \mathbf{b}^2\}$.

To update the synapses for this module, the motor-action circuit must interact with the policy circuit by first running its ancestral projection process (Equation 6) to obtain an immediate action $\mathbf{a}_t$. This produced action is then run through the policy neural structure (also via ancestral projection) to finally produce a set of action-values, one for each continuous action output node. With these action-values produced, the policy's second set of (actor-orienting) output error neurons $\mathbf{e}^0_a$ are used ($\mathbf{e}^0_a = -\mathbf{1}_a/A$). Then both the policy and actor circuits are temporarily coupled together and run with one single joint settling process. This joint settling process is done by clamping the motor-action model's output error neurons to the perturbation $\mathbf{d}_a$ produced by the policy circuit (at each step of the settling process) with respect to its input actions. This can be seen as setting the actor's output error neurons to be equal to $\mathbf{e}^0 = \mathbf{d}_a$ (ignoring its normally computed $\mathbf{e}^0$). This means that the policy and actor are briefly linked together to calculate the actor's synaptic updates (see the appendix for details and pseudocode).

Finally, we note that, for the agent in this study, we coupled the motor-action module with its own small memory module $\mathcal{M}$, based on experience replay [23]. However, inspired by work done in self-imitation learning [24], we only store in this memory trajectories that reach goal states. Specifically, to store a trajectory into $\mathcal{M}$, the ActPC agent measures an episode's cumulative sparse reward. Thus, $r_{epi} = \sum_t r_t$, and we only store it if this cumulative sparse signal is greater than/equal to the highest one of the episodes currently stored in $\mathcal{M}$ (see the appendix for details).

E. The Epistemic Signal: The NGC Generative World Model

Motivated by the finding of expected value estimation in the brain [25], [26], an ActPC agent implements a neural circuit that produces intrinsic reward signals. At a high level, this neural machinery facilitates some of the functionality of the basal ganglia and procedural memory, simulating an internal reward-creation process [27]. Concretely, we refer to the above as an NGC dynamics (or generator) model, where the reward is calculated as a function of its error neurons.

In Figure 1 (Middle), we graphically depict the design of the NGC dynamics model used to generate epistemic rewards (or values meant to encourage exploration). The NGC dynamics circuit processes the current state/observation $\mathbf{o}_t$ and the current action $\mathbf{a}_t$ ($\mathbf{a}_t \in \mathcal{R}^{A \times 1}$), as produced by the motor-action model, and predicts the value of the future state $\mathbf{o}_{t+1}$. When provided with $\mathbf{z}^0 = \mathbf{o}_{t+1}$, the dynamics circuit runs the following for its layer-wise predictions:

$$\bar{\mathbf{z}}^2 = \mathbf{W}^3_a \cdot \mathbf{a}_t + \mathbf{W}^3_z \cdot \mathbf{z}^3_t + \mathbf{M}^3 \cdot \mathbf{m}_t + \mathbf{b}^2 \quad (14)$$

$$\bar{\mathbf{z}}^1 = \mathbf{W}^2 \cdot \phi(\mathbf{z}^2_t) + \mathbf{M}^2 \cdot \mathbf{m}_t + \mathbf{b}^1 \quad (15)$$

$$\hat{\mathbf{o}}_{t+1} = \bar{\mathbf{z}}^0 = g^0\Big(\mathbf{W}^1 \cdot \phi(\mathbf{z}^1_t) + \mathbf{M}^1 \cdot \mathbf{m}_t + \mathbf{b}^0\Big) \quad (16)$$

and leverages the NGC settling process (see Equation 1) to compute its internal state values, i.e., $\mathbf{z}^3_t, \mathbf{z}^2_t, \mathbf{z}^1_t$. (Note that $\mathbf{W}^3 = [\mathbf{W}^3_\mathbf{a}, \mathbf{W}^3_\mathbf{z}]$, where we have made clear the explicit distinction between observation-conditioned input synapses and action-conditioned input synapses.) Notice that this circuit's post-activation prediction functions for the internal layers are $g^2(\mathbf{v}) = g^1(\mathbf{v}) = \mathbf{v}$, and $\phi^2(\mathbf{v}) = \phi^1(\mathbf{v}) = \phi(\mathbf{v})$ (the same state and prediction activation function type is used across its layers). Once the above dynamics have been executed, the NGC dynamics model's synapses are adjusted via Hebbian synaptic updates (as in Sub-Section II-B). The parameters of the generator circuit are collected into the construct $\Theta_g = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \mathbf{b}^0, \mathbf{b}^1, \mathbf{b}^2\}$.

To generate the value of the epistemic reward [19], the dynamics model first settles to a prediction $\hat{\mathbf{o}}_{t+1}$ given the value of the next observation $\mathbf{o}_{t+1}$. After its settling process has finished, the activity signals of its (squared) error neurons are summed to obtain the circuit's epistemic reward signal:

$$\widehat{r}^{ep}_t = \sum_j (\mathbf{e}^0)^2_{j,1} + \sum_j (\mathbf{e}^1)^2_{j,1} + \sum_j (\mathbf{e}^2)^2_{j,1} \quad (17)$$

$$r^{ep}_t = \widehat{r}^{ep}_t / (r^{ep}_{max}) \quad \text{where } r^{ep}_{max} = \max(\widehat{r}^{ep}_1, \widehat{r}^{ep}_2, ..., \widehat{r}^{ep}_t) \quad (18)$$

where the epistemic reward signal is normalized to the range of $[0, 1]$ by tracking the maximum epistemic signal observed throughout the course of the simulation. This epistemic signal encourages the ActPC agent to take actions that explore environmental states/observations that it finds surprising.

F. The NGC Prior: A Model-Based Instrumental Signal

To produce a useful instrumental signal to guide the agent to goal states in the face of sparse rewards, we design a circuit that emulates a prior preference, a hallmark of active inference, without having to resort to hand-crafted reward functions. To do so, we learn a generative model of observation states collected from human/machine demonstrations (successful demonstrations on the target problems). This means that our prior (or instrumental) model can be considered to be a simple form of imitation learning [28]. Formally, our prior model learns to produce $p(\mathbf{o}_{t+1}|\mathbf{o}_t)$ at each step throughout a given demonstration episode.

Concretely, the prior preference circuit elicits the prediction dynamics below:

$$\bar{\mathbf{z}}^2 = \mathbf{W}^3 \cdot \mathbf{z}^3 + \mathbf{M}^3 \cdot \mathbf{m}_t + \mathbf{b}_2 \quad (19)$$

$$\bar{\mathbf{z}}^1 = \mathbf{W}^2 \cdot \phi(\mathbf{z}^2) + \mathbf{M}^2 \cdot \mathbf{m}_t + \mathbf{b}_1 \quad (20)$$

$$\hat{\mathbf{o}}_{t+1} = g^0(\bar{\mathbf{z}}^0 = \mathbf{W}^1 \cdot \phi(\mathbf{z}^1) + \mathbf{M}^1 \cdot \mathbf{m}_t + \mathbf{b}_0) \quad (21)$$

where, in contrast to the epistemic generative circuit, the prior preference module focuses on predicting the next observation $\mathbf{o}_{t+1}$ given only the previous observation $\mathbf{o}_t$. The parameters of the prior circuit are collected into the construct $\Theta_p = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \mathbf{M}^1, \mathbf{M}^2, \mathbf{M}^3, \mathbf{b}^0, \mathbf{b}^1, \mathbf{b}^2\}$.

An instrumental reward signal is produced in a similar fashion as that of the epistemic signal:

$$\widehat{r}^{in}_t = \sum_j (\mathbf{e}^0_{j,1})^2 + \sum_j (\mathbf{e}^1_{j,1})^2 + \sum_j (\mathbf{e}^2_{j,1})^2 \quad (22)$$

$$r^{in}_t = -\widehat{r}^{in}_t / (r^{in}_{max}) \quad \text{where } r^{in}_{max} = \max(\widehat{r}^{in}_1, \widehat{r}^{in}_2, ..., \widehat{r}^{in}_t) \quad (23)$$

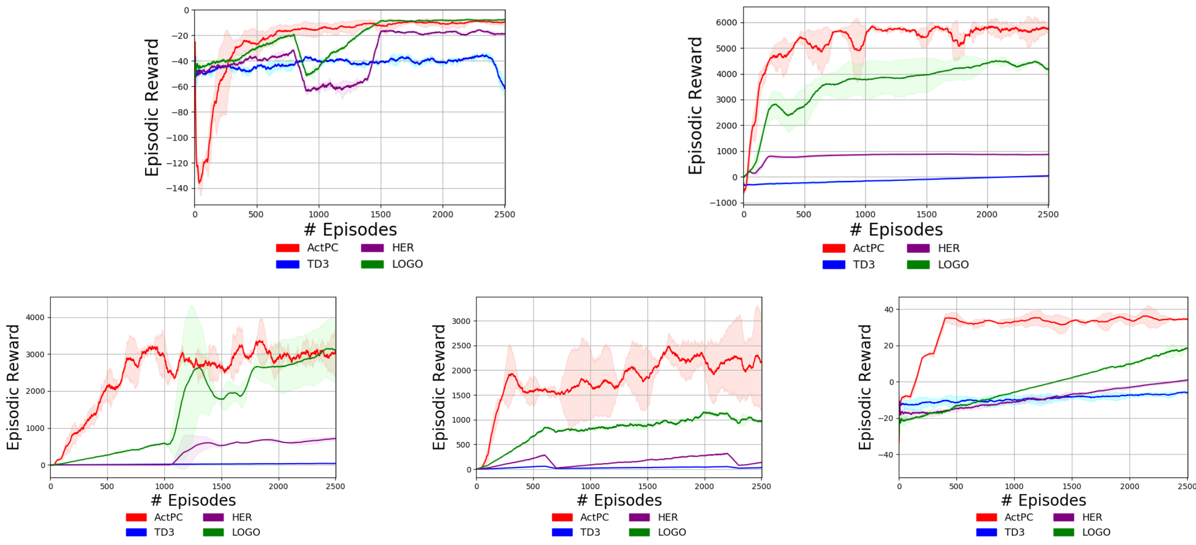

Fig. 2: Sparse robotic control problem results: (Top Left) reacher results, (Top Right) half-cheetah results, (Bottom Left) walker, (Bottom Middle) hopper, and (Bottom Right) swimmer (five-trial reward mean & std. dev. reported).

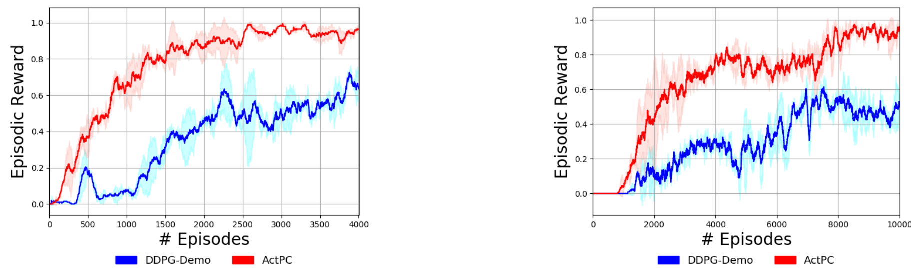

Fig. 3: (Left) The block-lifting task and (Right) can pick-and-place task results (five-trial reward mean & std. dev. reported).

where the instrumental reward signal is normalized in the same way as the epistemic signal as described in the last section. The instrumental signal induces goal-orienting/seeking behavior, mediating the agent's exploration of an environment by ensuring that such exploration serves its intent to solve a given problem or task. Note that the final reward signal r_t is calculated as follows: $r_t = \alpha_{in} r_t^{in} + \alpha_{ep} r_t^{ep}$ where α_{in} and α_{ep} (the reward weighting coefficients) are set to one in this work.

G. *Putting It All Together: The Neural Coding Agent*

Figure 1 (Left) depicts how the various NGC circuits described in the previous sub-sections, e.g., the policy, motor-actor, generator, prior circuits, and the memory replay buffers fit together to compose the full agent. At a high level, ActPC operates, given observation $\mathbf{o}_t$, according to the following steps: **(1)** the NGC motor-action circuit takes in $\mathbf{o}_t$ and produces a continuous action $\mathbf{a}_t$, **(2)** the ActPC agent receives observation $\mathbf{o}_{t+1}$ from the environment, i.e., the result of its action, **(3)** the NGC generator runs its dynamics (K steps of Equation 1) to find a set of hidden neural activity values that map $[\mathbf{a}, \mathbf{o}_t]$ to $\mathbf{o}_{t+1}$, produces epistemic signal r_t^{ep} (Equation 18), and then updates its own specific synapses using Equations 4-5, **(4)** the NGC prior circuit takes in $\mathbf{o}_t$ and runs its dynamics (K steps of Equation 1) to produce an instrumental signal r_t^{in} (Equation 23), and then the full reward signal is computed as $r_t = \alpha_{in} r_t^{in} + \alpha_{ep} r_t^{ep}$, **(5)** the NGC policy circuit takes in $\mathbf{a}_t$ and $\mathbf{o}_t$ and runs its dynamics (K steps of Equation 1) to find a set of hidden neural activities that allow a mapping from $\mathbf{o}_t$ to $\mathbf{t}_t$ (which contains r_t). Then the actor and policy circuits update their synapses (via Equations 4-5). Finally, **(6)** the ActPC agent transitions to $\mathbf{o}_{t+1}$ and moves back to step **(1)**. We refer the reader to the appendix for pseudocode depicting the full mechanics of ActPC, a discussion on connections to more general active inference biological process theory, policy gradients, and additional neuro-biological motivations behind the agent's design. Note that the synapses of all modules of ActPC, except for the (pre-trained) prior, are adapted at each time step. Finally, note that we leveraged *combined experience replay* [29] for the replay buffers (see Appendix for details), a scheme that fits well with ActPC's online processing scheme.

III. Experimental Results

In this section, we describe basic details of our experimental setup while the appendix contains extensive detail on hyper-parameters, task specifications, and related work.

A. *Investigated Models*

For the Mujoco robotic control tasks, we explored the following baselines: twin-delayed DDPG (TD3) [30], hindsight experience replay (HER) [31], and Learning Online with

TABLE I: Mujoco results (5-trial mean/std. dev. reported)

Reacher	**Avg. Return**	**R-Stability**
TD3	-61.682 ± 7.227	2.085 ± 0.323
HER	-18.772 ± 0.578	15.619 ± 0.357
LOGO	-7.583 ± 0.207	14.956 ± 1.542
ActPC	-9.269 ± 1.442	3.347 ± 1.116
Half-Cheetah	**Avg. Return**	**R-Stability**
TD3	36.988 ± 3.893	0.607 ± 0.026
HER	868.9 ± 19.827	0.046 ± 0.011
LOGO	4163.38 ± 260.745	0.085 ± 0.066
ActPC	5757.505 ± 130.922	0.088 ± 0.036
Walker	**Avg. Return**	**R-Stability**
TD3	39.071 ± 0.025	0.517 ± 0.022
HER	712.948 ± 57.917	0.119 ± 0.011
LOGO	3104.631 ± 905.941	0.096 ± 0.061
ActPC	3081.107 ± 668.267	0.568 ± 0.02
Hopper	**Avg. Return**	**R-Stability**
TD3	30.935 ± 0.65	0.503 ± 0.043
HER	133.638 ± 9.655	0.453 ± 0.037
LOGO	975.368 ± 24.273	0.461 ± 0.007
ActPC	2170.714 ± 918.793	0.349 ± 0.227
Swimmer	**Avg. Return**	**R-Stability**
TD3	-6.052 ± 0.678	10.349 ± 1.81
HER	0.956 ± 0.51	0.779 ± 0.129
LOGO	18.342 ± 1.221	0.516 ± 0.036
ActPC	34.498 ± 0.598	0.21 ± 0.008

TABLE II: Robosuite results (5-trial mean/std. dev. reported).

Block Lift	**Acc**	**R-Stability**
BC	100.0 ± 0.0	$--$
BC-RNN	100.0 ± 0.0	$--$
BCQ	100.0 ± 0.0	$--$
CQL	56.7 ± 40.3	$--$
HBC	100.0 ± 0.0	$--$
IRIS	100.0 ± 0.0	$--$
DDPG-Demo	63.5 ± 7.8	0.340 ± 0.043
ActPC	96.5 ± 2.1	0.048 ± 0.008
Can Place	**Acc**	**R-Stability**
BC	86.0 ± 4.3	$--$
BC-RNN	100.0 ± 0.0	$--$
BCQ	62.7 ± 8.2	$--$
CQL	22.0 ± 5.7	$--$
HBC	91.3 ± 2.5	$--$
IRIS	92.7 ± 0.9	$--$
DDPG-Demo	51.5 ± 3.5	0.351 ± 0.079
ActPC	94.0 ± 2.1	0.101 ± 0.028

Guidance Offline (LOGO) [32] (all models used demonstration data – see Appendix for descriptions of each as well as specific modifications made for this work). For the robosuite tasks, we compare to DDPG-Demo, which generalizes DDPG [22] to utilize demonstration data (we found it failed to perform without demonstrations).

B. Simulated Tasks

For all tasks, we measure/report two key quantities: 1) the episodic return (higher is better) and 2) R-stability, a novel metric that we developed for measuring an agent's online learning stability (lower is better - see appendix for details).

a) Mujoco Tasks: We start by experimenting with four challenging Mujoco-simulation tasks, namely the "reacher" (Reacher-v4), "half-cheetah" (Half-Cheetah-v2), "hopper" (Hopper-v2), and "walker" (Walker-v2), and "swimmer" (Swimmer-v2) tasks. All five tasks are converted to sparse reward variants. For all tasks except reacher, a reward r_t of 1 is awarded if a particular condition is met during an episode, whereas for reacher, $r_t = 1.0$ is awarded to the agent when the L2 distance between the arm fingertip and the goal state is less than 0.06 (else $r_t = 0$ or $r_t = -1$).

b) The Robosuite Tasks:: We next experiment with two difficult tasks taken from the high-quality, realistic simulator known as robosuite. In particular, we simulate the "block lifting" (Block Lift) and "can pick-and-place" (Can Place) tasks. Each problem is set to its sparse equivalent – $r_t = 1$ if the robotic arm reaches the goal state, otherwise, $r_t = 0$.

IV. Discussion

Based on the results of our simulations, we find that: (1) ActPC is able to perform well on both sets of tasks for all problems (both Mujoco and robosuite), (2) ActPC outperforms HER and TD3 on all five Mujoco tasks (and LOGO in 4/5 cases) while outperforming DDPG-Demo on both robosuite tasks (Figures 2 & 3), (3) offers generally good stability for most Mujoco tasks (it is best in 3/5 tasks, if we disregard the fact that TD3 performed poorly on the reacher task with a low return) and best stability for both robosuite tasks, and (4) is competitive with top-performing powerful baselines on the robosuite tasks, nearly reaching the same performance as behavior cloning (BC) and recurrent BC (BC-RNN) [33]. Outperforming other models such as Conservative Q-Learning (CQL) [33] on both tasks, and outperforming models such as hierarchical behavior cloning (HBC) [33], batch-constrained Q-learning (BCQ) [33], and IRIS on the harder can pick-and-place task. A detailed analysis, pseudocode, limitations, and computational complexity for Act-PC are presented in detail in the appendix.

V. Conclusions and Future Work

In this paper, we proposed active predictive coding (ActPC), a neuro-biologically-inspired learning process and agent system for tackling challenging sparse reward robotic control problems. Our experimental results, across two sets of robotic tasks demonstrate that learning powerful, goal-directed control agents is possible without backpropagation of errors (backprop). Furthermore, this work provides evidence that a backprop-free and neuro-biologically grounded scheme can offer a promising pathway toward developing agents that tackle the exploration-exploitation tradeoff dynamically. This could prove useful for research domains such as neurorobotics. ActPC, while potentially powerful as our results indicate, is not without its limitations (see Appendix for a detailed discussion), e.g., its missing other biological elements/properties, such as neural communication through spike trains (although related work has demonstrated that neural generative coding can be reformulated as a spiking neural system [34], a direction we intend to explore in future work). Future directions will include further studying and expanding the types of robotic control tasks that ActPC can tackle, investigating how ActPC can tackle the partially observable Markov decision process formulation of robotic control, and crucially generalizing ActPC to deal with the domain adaptation problem inherent to creating an embodied robotic agent for tackling real-world tasks.

References

[1] S. B. Furber, "Brain-inspired computing," *IET Computers & Digital Techniques*, vol. 10, no. 6, pp. 299–305, 2016.

[2] J. D. Kendall and S. Kumar, "The building blocks of a brain-inspired computer," *Applied Physics Reviews*, vol. 7, no. 1, p. 011305, 2020.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[4] M. Kawato and H. Gomi, "A computational model of four regions of the cerebellum based on feedback-error learning," *Biological cybernetics*, vol. 68, no. 2, pp. 95–103, 1992.

[5] H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," *Neural networks*, vol. 1, no. 3, pp. 251–265, 1988.

[6] K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *Journal of Physiology-Paris*, vol. 100, no. 1-3, pp. 70–87, 2006.

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

[8] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

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

[10] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," *arXiv preprint arXiv:1608.05343*, 2016.

[11] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, "Difference target propagation," in *Proceedings of the 2015th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ser. ECMLPKDD'15. Switzerland: Springer, 2015, pp. 498–515.

[12] M. Botvinick and M. Toussaint, "Planning as inference," *Trends in Cognitive Sciences*, vol. 16, no. 10, pp. 485–488, 2012.

[13] K. Friston, J. Mattout, and J. Kilner, "Action understanding and active inference," *Biological cybernetics*, vol. 104, no. 1, pp. 137–160, 2011.

[14] R. P. Rao and D. H. Ballard, "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." *Nat. Neurosci.*, vol. 2, no. 1, 1999.

[15] K. Friston, "A theory of cortical responses," *Philosophical transactions of the Royal Society B: Biological sciences*, vol. 360, no. 1456, pp. 815–836, 2005.

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

[17] A. Ororbia and D. Kifer, "The neural coding framework for learning generative models," *Nature communications*, vol. 13, no. 1, pp. 1–14, 2022.

[18] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[19] A. Ororbia and A. Mali, "Backprop-free reinforcement learning with active neural generative coding," *Proc. Conf. AAAI Artif. Intell.*, vol. 36, 2022.

[20] S. J. Gershman, "What does the free energy principle tell us about the brain?" *arXiv preprint arXiv:1901.07945*, 2019.

[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[24] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3878–3887.

[25] O. Hikosaka, K. Nakamura, and H. Nakahara, "Basal ganglia orient eyes to reward," *Journal of Neurophysiology*, vol. 95, no. 2, pp. 567–584, 2006.

[26] A. Rangel, C. Camerer, and P. R. Montague, "A framework for studying the neurobiology of value-based decision making," *Nature reviews neuroscience*, vol. 9, no. 7, pp. 545–556, 2008.

[27] W. Schultz, "Reward functions of the basal ganglia," *J. Neural Transm.*, vol. 123, no. 7, pp. 679–693, 2016.

[28] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[29] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.

[30] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[31] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[32] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," in *International Conference on Learning Representations*, 2021.

[33] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[34] A. Ororbia, "Spiking neural predictive coding for continual learning from data streams," *arXiv preprint arXiv:1908.08655*, 2019.