ARTICLE **Communicated by Rafal Bogacz**

Active Predictive Coding: A Unifying Neural Model for Active Perception, Compositional Learning, and Hierarchical Planning

Rajesh P. N. Rao
rao@cs.washington.edu
Dimitrios C. Gklezakos
gklezd@cs.washington.edu
Vishwas Sathish
vsathish@cs.washington.edu
Paul G. Allen School of Computer Science and Engineering and Center for Neurotechnology, University of Washington, Seattle, WA 98195, U.S.A.

There is growing interest in predictive coding as a model of how the brain learns through predictions and prediction errors. Predictive coding models have traditionally focused on sensory coding and perception. Here we introduce active predictive coding (APC) as a unifying model for perception, action, and cognition. The APC model addresses important open problems in cognitive science and AI, including (1) how we learn compositional representations (e.g., part-whole hierarchies for equivariant vision) and (2) how we solve large-scale planning problems, which are hard for traditional reinforcement learning, by composing complex state dynamics and abstract actions from simpler dynamics and primitive actions. By using hypernetworks, self-supervised learning, and reinforcement learning, APC learns hierarchical world models by combining task-invariant state transition networks and task-dependent policy networks at multiple abstraction levels. We illustrate the applicability of the APC model to active visual perception and hierarchical planning. Our results represent, to our knowledge, the first proof-of-concept demonstration of a unified approach to addressing the part-whole learning problem in vision, the nested reference frames learning problem in cognition, and the integrated state-action hierarchy learning problem in reinforcement learning.

1 Introduction

Predictive coding (Rao & Ballard, 1997, 1999; Rao, 1999; Keller & Mrsic-Flogel, 2018; Jiang & Rao, 2022b) has received increasing attention in recent years as a model of how the brain learns models of the world through prediction and self-supervised learning. In predictive coding, feedback connections from a higher to a lower level of a cortical neural network (e.g., the

Neural Computation **36, 1–32** (2024)
https://doi.org/10.1162/neco_a_01627

visual cortex) convey predictions of lower-level responses, and the prediction errors are conveyed via feedforward connections to correct the higher-level estimates, completing a prediction-error-correction cycle (see also Mumford, 1992). Such a model has provided explanations for a wide variety of neural and cognitive phenomena (Keller & Mrsic-Flogel, 2018; Jiang & Rao, 2022b). The layered architecture of the cortex is remarkably similar across cortical areas (Mountcastle, 1978), hinting at a common computational principle, with superficial layers receiving and processing sensory information and deeper layers conveying outputs to motor centers (Sherman & Guillery, 2013). The traditional predictive coding model focused on learning visual hierarchical representations and did not acknowledge the important role of actions in learning world models.

In this article, we introduce active predictive coding (APC), a new model of predictive coding that combines state and action networks at different abstraction levels to learn hierarchical internal models. The model provides a unified framework for addressing several important but seemingly unrelated problems in perception, action, and cognition as described below.

2 Related Work and Contributions

2.1 Part-Whole Learning Problem. Hinton and colleagues have posed the problem of how neural networks can learn to parse visual scenes into part-whole hierarchies by dynamically allocating nodes in a parse tree. They have explored networks that use a group of neurons to represent not only the presence of an object but also parameters such as position and orientation (Sabour et al., 2017; Kosiorek et al., 2019; Hinton et al., 2018; Hinton, 2021). Such equivariant models seek to overcome the inability of deep convolutional neural networks (CNNs) (Krizhevsky et al., 2012) to explain the images they classify in the way humans do, in terms of objects, parts and their locations.

Other approaches such as the ones presented in Burgess et al. (2019) or Greff et al. (2019) represent a scene by segmenting it into distinct objects and modeling them separately. Seitzer et al. (2023) demonstrate such an approach on more realistic data sets. Oquab et al. (2023) apply principal-component analysis (Pearson, 1901) to image-patch features learned from a self-supervised, transformer-based model. They show that the first few principal components capture information that can be used to segment an object from its background or identify different parts of the object regardless of pose or style.

2.2 Reference Frames Problem. In a parallel line of research, Hawkins and colleagues (Hawkins, 2021; Lewis et al., 2019; see also George & Hawkins, 2009; George et al., 2017; Guntupalli et al., 2023) have taken inspiration from the cortex and "grid cells" to propose that the brain uses object-centered reference frames (or "schemas") to represent objects, spatial

environments, and even abstract concepts. There is some evidence from hippocampal and cortical studies in rodents and humans for spatial (and more abstract) reference frames being used for solving problems such as navigation and abstract reasoning (O'Keefe & Dostrovsky, 1971; Moser et al., 2017; Constantinescu et al., 2016). The question of how such reference frames can be learned and used in a nested manner for hierarchical recognition and reasoning has remained open.

2.3 Integrated State-Action Hierarchy Learning Problem. A considerable literature exists on hierarchical reinforcement learning (see Hutsebaut-Buysse et al., 2022, for a recent survey), where the goal is to make traditional reinforcement learning (RL) algorithms more efficient through state and/or action abstraction. While one class of approaches relies on identifying particular states as "subgoals" and achieving these subgoals to solve a main task (Hafner et al., 2022), another class of approaches uses options (Sutton et al., 1999; Bacon et al., 2016), which are abstract actions that can be selected in particular states (in the option's "initiation set") and whose execution results in a sequence of primitive actions as prescribed by the option's lower-level policy. The problem of simultaneously learning state and action abstraction hierarchies has remained relatively less explored.

2.4 Hypernetworks. A hypernetwork (Ha et al., 2017) is an artificial neural network that generates parameters for another neural network called the primary network. The primary network is essentially a placeholder into which the generated weights are plugged in, to be used for solving a downstream task. In the simple case of a fully connected primary network of K layers, these parameters consist of the K weight matrices $\{W_j\}_{j=1}^{K}$ and bias vectors $\{b_j\}_{j=1}^{K}$. Hypernetworks are typically trained end-to-end.

Hypernetworks in their most general form (as defined above) are biologically implausible: neural networks in the brain cannot generate other neural networks ex nihilo for each input. Although we do not pursue biological plausibility in this article, we note that there are at least two ways to approximate hypernetworks in a biologically plausible manner. First, one can use a specialized type of hypernetwork: instead of generating a new network, a hypernetwork could modulate an existing "primary" neural network P, for example, by generating gain factors that multiply P's outputs and, by doing so, change the function being computed by the neural network P. Such "gain modulation" (Zipser & Andersen, 1988; Salinas & Abbott, 1996; Salinas & Sejnowski, 2001; Ferguson & Cardin, 2020; Shine et al., 2021; Stroud et al., 2018) appears to be common in the cortex, observed, for example, in the multiplicative modulation of tuning curves of visual cortical neurons during attention (McAdams & Maunsell, 1999) and in the changes in the input-output function of neurons in deep layers of the cortex due to

top-down modulatory inputs to their apical dendrites (Larkum et al., 2004). It is important to note that when gain modulation is mediated by neuromodulators such as dopamine, it is typically diffuse and not synapse- or neuron-specific, as needed to implement a full-fledged hypernetwork. However, even diffuse coarse-grained modulation of a recurrent primary network can in some cases achieve a performance similar to that of neuron-specific modulation, as shown by Stroud et al. (2018). Second, instead of generating parameters for a network, the hypernetwork can generate a top-down contextual input for the primary network P. P's input is augmented with the contextual input, thereby allowing the hypernetwork to modulate the function being computed by P (Yang et al., 2019; Eliasmith et al., 2012). Such an approach, known as the embedding approach in AI, can approximate the computational function of hypernetworks (Galanti & Wolf, 2020) in a biologically plausible manner (see Rao, 2022, for more details).

Although hypernetworks have been used extensively in AI, the problem of using hypernetworks for hierarchical abstraction of state and action functions has remained open.

2.5 Contributions of the Article. The APC model addresses the problems above in a unified manner using state/action embeddings and hypernetworks to dynamically generate and generalize over state and action networks at multiple hierarchical levels. The APC model contributes to a number of lines of research not connected before:

1. Perception, predictive coding, and reference frame learning: APC extends predictive coding and related neuroscience models of brain function (Rao & Ballard, 1999; Friston & Kiebel, 2009; Jiang et al., 2021; Jiang & Rao, 2022a) to hierarchical sensory-motor inference and learning, and connects these to learning nested reference frames (Hawkins, 2021) for perception and cognition.
2. Attention models: APC extends previous hard attention models such as the recurrent attention model (RAM) (Mnih et al., 2014) and attend-infer-repeat (AIR) (Eslami et al., 2016). As an active visual perception technique, it learns structured hierarchical strategies for sampling key parts of the visual scene.
3. Hierarchical planning and reinforcement learning: APC contributes to hierarchical planning and reinforcement learning (Hutsebaut-Buysse et al., 2022; Botvinick et al., 2009) by proposing a new way of simultaneously learning abstract macro-actions or options (Sutton et al., 1999) and abstract states.

2.6 General Applicability of the Model. When applied to vision, the APC model learns to hierarchically represent and parse images into parts and locations. When applied to RL problems, the model can exploit hypernetworks to (1) define a state hierarchy not merely through state

aggregation, but by abstracting transition dynamics at multiple levels, and (2) potentially generalize learned hierarchical states and actions (options) to novel scenarios via interpolation and extrapolation in the input embedding space of the hypernetworks. Our approach brings us closer to solving an important challenge in both AI and cognitive science (Lake et al., 2017): How can neural networks learn hierarchical compositional representations that allow new concepts to be created, recognized, and learned?

3 Active Predictive Coding

The APC model implements a hierarchical version of the traditional partially observable Markov decision process (POMDP) (Kaelbling et al., 1998; Rao, 2010). Figure 1a shows the canonical APC generative module. The module is self-similar and is separated into two distinct systems: the state system and the action system. Following the POMDP formulation, the state system captures the transition dynamics of the environment. The action system determines which action (actual or abstract) the agent will take toward solving the downstream task. The state system maintains historical context via the recurrent state vector $\mathbf{s}$ and the action system via the action vector $\mathbf{a}$. We denote the recurrent neural networks (RNNs) that generate these vectors by f_s and f_a, respectively. We can stack multiple such modules by allowing the current state and action vector at any given level to generate an entire new state network and action network respectively at the level below using hypernetworks (Ha et al., 2017).

Specifically, let $\mathbf{s}^{(i+1)}$ and $\mathbf{a}^{(i+1)}$ denote the state and action vectors at the higher level $i+1$. The state embedding vector $\mathbf{s}^{(i+1)}$ together with the function H_s^i (implemented as a hypernetwork) generates a lower-level state transition function f_s^i (Figure 1a, left). Similarly, the action embedding vector $\mathbf{a}^{(i+1)}$ together with the hypernetwork H_a^i generates a lower-level option/policy function f_a^i (see Figure 1a, right). Both f_s^i and f_a^i are implemented as RNNs. The state and action systems exchange information horizontally within each level as shown in Figure 1b for a three-level APC model. During inference (see below), the APC architecture employs a feedback mechanism via which the lower level can inform the higher level of its "findings" via prediction errors (see Figure 1b, red arrows).

The novel idea behind the APC approach is to imbue structure by abstracting lower-level state and action transitions via RNN (subprogram) generation and restricting the scale or extent of a subprogram in terms of temporal steps or the afforded action space. In our current implementation, the lower-level RNNs (subprograms) execute for a fixed number of time steps before returning control back to the higher level.[1] Although not

[1] Future implementations will explore the use of termination functions (Sutton et al., 1999; Eslami et al., 2016) to allow a variable number of time steps at each level and for each input.

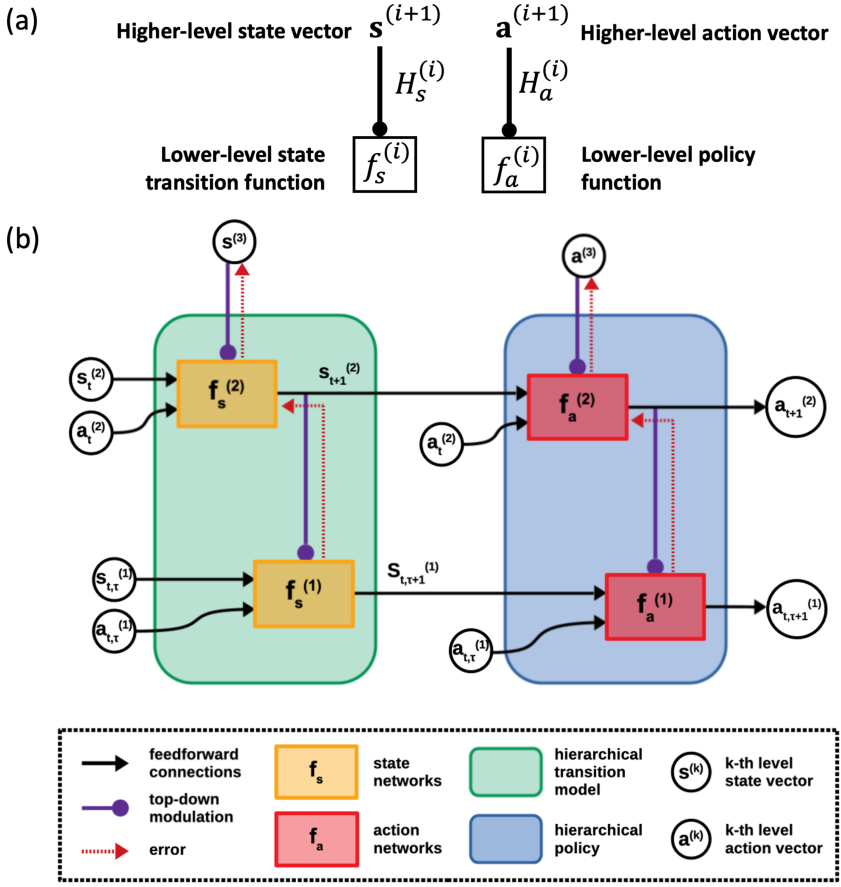


Figure 1: Active predictive coding model. (a) Canonical APC generative module. Lower-level functions are generated via hypernetworks based on current higher-level state and action-embedding vectors. All functions (in boxes) are implemented as recurrent neural networks (RNNs). Arrows with circular terminations generate function parameters (here, neural network weights and biases). (b) Generation and inference (via prediction errors) of states and actions in a three-level APC model. See the text for details.

the focus of this paper, we note that the components of the APC model described above could potentially be implemented with biologically plausible mechanisms within the laminar structure of the neocortex (Rao, 2022). Specifically, as mentioned in section 2.4 and discussed in more detail in Jiang & Rao (2022a) and Rao (2022), hypernetworks could be implemented via the neural mechanism of top-down gain modulation in cortical neurons (Larkum et al., 2004; Ferguson & Cardin, 2020).

3.1 Inference in the Active Predictive Coding Model. Inference involves estimating the state and action vectors at multiple levels based on the sequence of inputs produced by interacting with the environment in the context of a particular task or goal. For the rest of this article, we assume a two-level APC model (see Figure 1b without $\mathbf{s}^{(3)}$ and $\mathbf{a}^{(3)}$), which is sufficient to illustrate the basic capability of the proposed framework (we leave the exploration of deeper models to future work; see the discussion in section 6).

Given a two-level APC model, we assume the top level runs for T_2 steps (referred to as macrosteps). For each macrostep, the bottom level runs for T_1 "microsteps." To improve readability, instead of $f_s^{(2)}$, $f_a^{(2)}$ and $f_s^{(1)}$, $f_a^{(1)}$, we use the notations F_s, F_a and f_s, f_a to denote the top-level and bottom-level state and action functions, respectively (all functions are implemented by RNNs). Similarly, instead of $s^{(2)}, a^{(2)}$ and $s^{(1)}, a^{(1)}$, we use the notation S, A and s, a to denote the top-level and bottom-level state and action embedding vectors, respectively (we omit boldface notation for vectors for the rest of the article as well). These state and action vectors are estimated by the recurrent activity vectors of the respective state and action networks. We use the notation $f(;\theta)$ to denote a network parameterized by $\theta = \{W_l, b_l\}_{l=1}^{L}$, the weight matrices and biases for all the layers. Thus, the bottom-level state and action RNNs are denoted by $f_s(;\theta_s)$ and $f_a(;\theta_a)$, while their activity vectors are denoted by $s_{t,\tau}$ and $a_{t,\tau}$ respectively (t ranges over macrosteps, τ over microsteps).

At each macrostep t, the top-level state RNN F_s produces a new state embedding vector S_t based on the previous state and action embedding vectors. This higher-level state S_t and action A_{t-1} from the previous macrostep are fed to the action/policy RNN F_a (which is determined by the current task or goal) to produce the next action-embedding vector A_t (a macroaction/option/subgoal). The embedding vector A_t is used as input to a nonlinear function, implemented by the hypernetwork H_a, to dynamically generate the parameters $\theta_a(t)(= H_a(A_t))$ for the lower-level action RNN, which implements a policy to generate primitive actions suitable for achieving the subgoal associated with A_t.

The higher-level state S_t also defines a new reference frame for the lower level to operate over as follows: S_t (and any other state-relevant information) is fed as input to the state hypernetwork H_s to generate the lower-level parameters $\theta_s(t)(= H_s(S_t))$ specifying a dynamically generated bottom-level state RNN characterizing the state transition dynamics locally, for example, local parts and their transformations in vision (see Application I in section 4) and navigation dynamics in a local region of a building (see application II in section 5).

Each microstep proceeds in a manner similar to a macrostep. The bottom-level action RNN produces the next action $a_{t,\tau+1}$ based on the current lower-level state and previous action (see Figure 1b, lower right). This action (e.g., sensor/body movement or a lower-level abstract action) results in a new

input being generated by the environment for the bottom (and possibly higher) state network.

To predict an input $I_{t,\tau}$ at time step (t, τ), the lower-level state vector $s_{t,\tau}$ is fed to a generic decoder network D to generate the input prediction $\hat{I}_{t,\tau}$. This predicted input is compared to the actual input to generate a prediction error $\epsilon_{t,\tau} = I_{t,\tau} - \hat{I}_{t,\tau}$. Following the predictive coding model (Rao & Ballard, 1999), the prediction error is used to update the state vector via the state network: $s_{t,\tau+1} = f_s(s_{t,\tau}, a_{t,\tau}, \epsilon_{t,\tau}; \theta_s(t))$. Additionally, for inference of the top-level state, the top-level RNN activity vector is updated using information from the lower-level state vectors $s_{t,0:T_1}$, and the process continues.

3.2 Training the Active Predictive Coding Model. Since the state networks are task-agnostic and geared toward capturing the dynamics of the world, they are trained using self-supervised learning by minimizing prediction errors. For the results in this article, we used backpropagation, but other biologically plausible mechanisms for minimizing prediction errors may also be used (Rao & Ballard, 1999; Whittington & Bogacz, 2017; Lillicrap et al., 2020); such mechanisms have also been shown to emerge as a consequence of minimizing energy consumption (Ali et al., 2022). The action networks are trained to integrate the information provided by the state vectors toward a downstream task by minimizing the total expected task loss: this can be done using either reinforcement learning or planning with the help of the state networks. In application I, we illustrate the use of reinforcement learning,[2] while in application II, we illustrate the use of planning, but the APC framework is flexible and allows either approach for estimating actions. Algorithm 1 summarizes the APC training process, with further details for each application provided in the sections that follow.

4 Application I: Visual Perception

A long-standing problem in vision and cognitive science is (Hinton, 2021): How can neural networks learn intrinsic references frames for objects and concepts and parse inputs (e.g., images) into part-whole hierarchies? Human vision provides an important clue. Unlike convolutional nets, which need to process an entire scene, human vision is an active sensory-motor process, sampling the scene via eye movements to move the high-resolution fovea to task-relevant locations, accumulating evidence for or against competing visual hypotheses (Wedel et al., 2022). The APC model is well suited to emulating the sensory-motor nature of human vision, given its integrated state and action networks.

[2] For simplicity, we use the classic REINFORCE algorithm (Williams, 1992) with backpropagation, but more efficient algorithms such as proximal policy optimization (PPO) may also be used.

Algorithm 1: Active Predictive Coding Training Algorithm.

Assumptions: A two-level APC model

Parameters: θ_{RNN} for F_s, F_a, and θ_{NN} for H_s and H_a

Data: Episodic transitions from environment for navigation; or a set of glimpse images for visual reconstruction

Result: Trained two-level APC network that can efficiently navigate to a goal in the environment; or effectively reconstruct images with few glimpses

`// Initialization`

Initialize top-level state and action vectors S_0, A_0 randomly for navigation or via a random glimpse for reconstruction;

Optionally initialize bottom-level states and actions $s_{0,0}$ and $a_{0,0}$ using an initialization network;

`// Training`

for *a sample transition or glimpse in Data* **do**

  for $t \leftarrow 1$ **to** T_2 **do**

    Generate lower state network weights $\theta_s = H_s(S_t)$;

    Generate lower action network weights $\theta_a = H_a(A_t)$;

    for $\tau \leftarrow 1$ **to** T_1 **do**

      θ_s parameterizes f_s, and θ_a parameterizes f_a;

      Obtain gradients for lower-level transition model $f_s(s_{t,\tau+1}|s_{t,\tau}, a_{t,\tau})$ with respect to prediction loss over states/glimpses;

      Obtain gradients for lower-level policy $f_a(a_{t,\tau+1}|s_{t,\tau+1}, a_{t,\tau})$ via policy gradient loss over environment rewards or reconstruction success, or via planning to infer actions;

    end

    Obtain gradients for higher-level transition model $F_s(S_{t+1}|S_t, A_t)$ with respect to prediction loss;

    Obtain gradients for higher-level policy $F_a(A_{t+1}|S_{t+1}, A_t)$ with respect to policy gradient loss;

  end

  Adjust model parameters using the obtained gradients

end

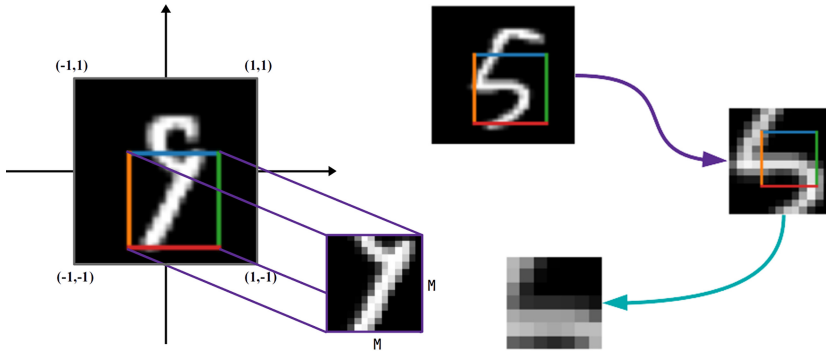


Figure 2: Reference frames for images. (a) The top level of the model picks a location and focuses on a subregion of a prespecified size (here, $M \times M$) centered at that location. This subregion is hypothesized by the model to contain a "higher-level part" of the object at a location relative to the original reference frame of the object (here, the digit 9). This subregion selected by the higher level in turn acts as the reference frame for the lower level. (b) Two-level model. The top level focuses on a subregion one-quarter the area of the initial input and fixes this region as the current reference frame. The next level focuses on locations within this local frame of reference and extracts sub-subregions, which contain subparts of the higher-level part at particular locations, all calculated relative to the local frame of reference. This hierarchical factoring of parts, subparts, and their transformations within local reference frames plays a critical role in endowing the APC model with compositionality.

For visual perception and part-whole learning, the actions in the APC model emulate eye movements (or attention) by moving a "glimpse sensor" (Mnih et al., 2014), which extracts high-resolution information about a small part of a larger input image. Ideally we would like our model to exhibit spatial convergence as we go up the representational hierarchy, capturing the inductive bias that an entity has a larger spatial extent than its constituent parts. The APC vision model implements this concept by using recursive object-centered reference frames. The top level of an APC architecture spans the entire image. At each step, the network chooses a subregion of the image to focus on (see Figure 2a). It then generates a lower-level image parser (comprising state-action subnetworks) and assigns this image subregion as the input to the lower level. The bottom-most level has direct access to the image via small-sized glimpses. The APC model performs a type of depth-first exploration of the representational graph, where each layer descends deeper into the graph with a new object-centered reference frame. These stacks of reference frames can be composed to derive the

absolute location of any sampled glimpse within the image. Figure 2b shows an example of recursive reference frame traversal down a two-level hierarchy.

Interactions with an image I (of size $N \times N$ pixels) are carried out through a glimpse sensor G. This sensor takes in a location l and a fixed-scale fraction m and extracts a square glimpse or patch $g = G(I, l, m)$ centered around l and of size $(mN) \times (mN)$. Since l is continuous, the sensor is implemented using a subdifferentiable bilinear interpolation module as introduced in Jaderberg et al. (2015). The image dimensions are normalized so that $l \in [-1, 1]$; m is hard-coded for each layer. Other transformations such as rotation and shear are ignored in the current version of this model, but incorporating them represents an obvious direction for future research.

Note that the APC model's use of a glimpse mechanism offers an additional degree of biological plausibility when compared to approaches that assume access to the whole image at once. Additionally, using a foveation emulator to intelligently select image locations for further processing potentially uses fewer neurons in the network and lowers energy expenditure compared to models that process all image locations such as CNNs (Krizhevsky et al., 2012) or vision transformers (ViTs; Dosovitskiy et al., 2021); this is because the latter scale quadratically with input size, while models using foveation such as APC have constant size regardless of input size.

4.1 Higher-Level Operation. The APC model we implemented for vision uses continuous feedback from the lower-level states. We use $S_{t,\tau}, A_{t,\tau}$ to denote the top-level vectors at macrostep t and microstep τ. Although the top-level states receive continuous feedback, the lower-level parameters $\theta_s(t)$ and $\theta_a(t)$ are generated at the beginning of each macrostep using only $S_{t,0}, A_{t,0}$ and stay fixed throughout. The initial lower-level state vector $s_{t,0}$ is initialized through a small neural network, using $S_{t,0}$. This state vector is then used together with f_a to initialize $a_{t,0}$.

At each macrostep t, the top-level action RNN updates its activity vector and generates two values via two networks: a location L_t and a macro-action (or option) z_t (see Figure 3). The location L_t is used to restrict the bottom level to a subregion $I_t^{(1)} = G(I, L_t, M)$ corresponding to a new frame of reference of scale M, centered around L_t (see Figure 2a). The option z_t is used as an embedding vector input to a nonlinear function, implemented by a hypernetwork H_a, to dynamically generate the parameters $\theta_a(t) = H_a(z_t)$ of the lower-level action RNN. For exploration during reinforcement learning, we treat the location network output of the top-level action RNN as a mean value $\bar{L}_t$ and add gaussian noise with fixed variance to sample an actual location: $L_t = \bar{L}_t + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

The state vector $S_{t,0}$ and location L_t are fed as inputs to the state hypernetwork H_s to generate the parameters $\theta_s(t)$ specifying a dynamically generated bottom-level state RNN for the current frame of reference. Figure 3 illustrates this top-down generation process.

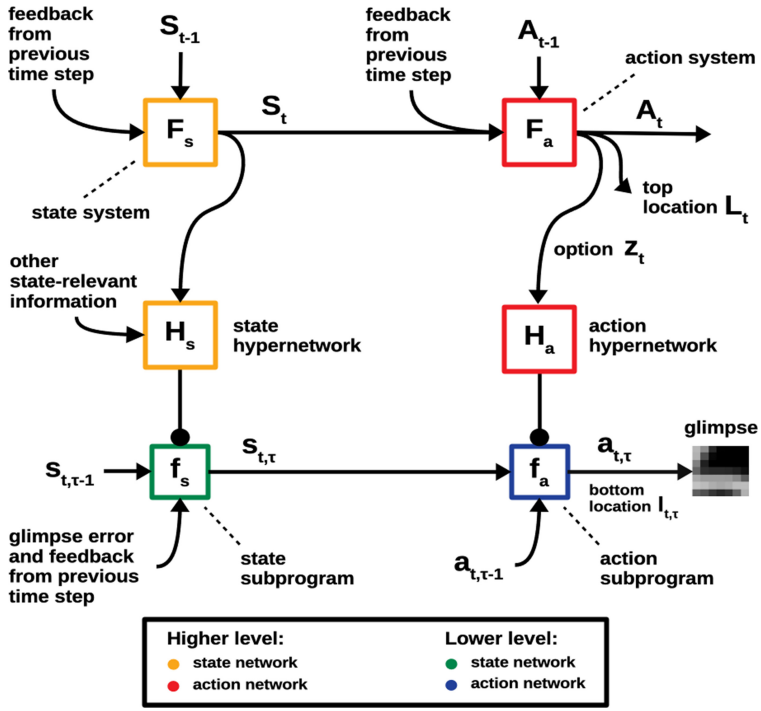


Figure 3: Active predictive coding: Generation of parts and subparts. Dynamic generation of bottom-level state RNN f_s and action RNN f_a (subprograms) from top-level state vector S_t and action vector A_t. This diagram elaborates the one in Figure 1b for a two-level APC model for vision. The generation process is related to the stacked frames of reference as shown in Figure 2b.

4.2 Lower-Level Operation. At the beginning of each microstep, the higher-level state $S_{t,0}$ is used to initialize the bottom-level state vector via a small feedforward network. Each microstep proceeds in a manner similar to a macrostep. The bottom-level action RNN produces the action vector $a_{t,\tau}$ based on the current state and past action, and a location $l_{t,\tau}$ is chosen as a function of $a_{t,\tau}$ (see Figure 3, lower right). This results in a glimpse image $g_{t,\tau} = G(I_t^{(1)}, l_{t,\tau}, m)$ of scale m centered around $l_{t,\tau}$ within the image subregion $I_t^{(1)}$ specified by the higher level. Figures 2b and 3 show the frames of reference and the corresponding image subregions across the two levels and how they relate to the operation of the two levels and state-action systems.

To predict the next glimpse image at the location specified by the action network, the lower-level state vector $s_{t,\tau}$, along with locations L_t and $l_{t,\tau}$, are fed to a generic decoder network D to generate the predicted glimpse $\hat{g}_{t,\tau}$.

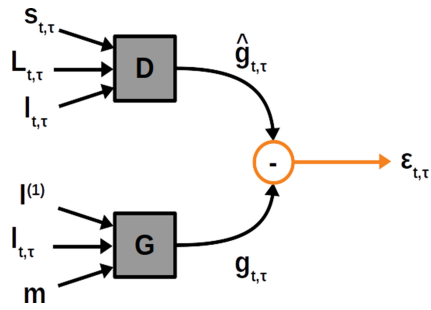


Figure 4: Active predictive coding: prediction. Computation of prediction error between predicted glimpse $\hat{g}$ generated by decoder D for the current time step (t, τ) and actual glimpse image g from the glimpse sensor after moving to the location $l_{t,\tau}$ produced by the action system.

This predicted glimpse is compared to the actual glimpse image to generate a prediction error $\epsilon_{t,\tau} = g_{t,\tau} - \hat{g}_{t,\tau}$ (see Figure 4). Following the predictive coding model (Rao & Ballard, 1999), the prediction error is used to update the state vector via the state network: $s_{t,\tau+1} = f_s(s_{t,\tau}, \epsilon_{t,\tau}, l_t; \theta^{(s)}(t))$. For exploration during reinforcement learning, we follow the same gaussian noise-based exploration strategy as the top level.

During each microstep, the top-level state RNN activity vector is updated using the bottom-level state vector and the top-level location:

$$S_{t,\tau+1} = S_{t,\tau} + F_s(S_{t,\tau}, s_{t,\tau+1}, L_t).$$

The top-level action RNN activity vector A_t is updated in a similar way,

$$A_{t,\tau+1} = A_{t,\tau} + F_a(S_{t,\tau+1}, a_{t,\tau+1}, L_t),$$

and the process continues. Note that we are using residual connections here to assist the learning process. This architecture can be readily extended to more levels by having F_s, F_a be dynamically generated by another parent level, and so on.

4.3 Training the Active Predictive Coding Network. The state and action systems are trained separately via different loss functions. The state system is trained to minimize prediction errors via backpropagation, while the action system is trained to minimize the total expected task loss via the reinforcement learning algorithm REINFORCE (Williams, 1992) together with backpropagation. In the implementation of the training procedure, whenever the state vectors at any given level are passed as input to that level's action network, the gradients for backpropagation are cut off. The

goal of the state prediction system is to predict the next state and is task-agnostic. The goal of the action system is to choose effective actions given past states and actions, so that the task loss is minimized.

4.3.1 Training the State System. The prediction error $\epsilon_{t,\tau}$ is given by

$$\epsilon_{t,\tau} = g_{t,\tau} - \hat{g}_{t,\tau} = G(I_t^{(1)}, l_{t,\tau}, m) - D(s_{t,\tau}, L_t, l_{t,\tau}). \tag{4.1}$$

The prediction error loss function is given by

$$L_{\text{pred}} = \sum_{t=1}^{T_2} \sum_{\tau=1}^{T_1} \|\epsilon_{t,\tau}\|_2^2. \tag{4.2}$$

At the end of a macrostep t, the higher level also reconstructs the current reference image $I_{\text{ref}}^{(1)}$, downsampled to the size of a lower-level glimpse, using a decoder D_{ref} with inputs $S_{t+1,0}$ and L_t. The corresponding loss term is $L_{\text{ref}} = \sum_{t=1}^{T_2} \|I_{\text{ref}}^{(1)} - D_{\text{ref}}(S_{t+1,0}, L_t)\|_2^2$. This term acts as a form of regularization and incentivizes the model to maintain information regarding the top frame of reference throughout the execution of the macrostep. Note that while $I_{\text{ref}}^{(1)}$ is available at training time, the generated error is not used as feedback and the reference image is not provided at test time. The total loss function for training the state networks at the two levels via backpropagation is given by

$$L_{\text{state}} = L_{\text{pred}} + L_{\text{ref}}. \tag{4.3}$$

4.3.2 Training the Action System. To apply the vision APC model to a given task (such as image reconstruction or classification), either the state or action RNN vectors can be provided as input to another neural network trained for the task. Here we use the action vectors. Let $A_{\text{out}}(t, \tau) = [A_{t,\tau}\ a_{t,\tau}]^T$ be the concatenation of top- and bottom-level action vectors for time step (t, τ). Let L_{task} be the task loss. Using just the final A_{out} (as in RAM; Mnih et al., 2014) for training actions has the shortcoming that the resulting reward function is sparse (the model is evaluated only after the final step). We use a dense, structured reward function (in our case, a dense loss function) as follows. For each microstep, we compute the marginal change in loss after the action for that step (i.e., fixating on a new location) has been executed:

$$v_{t,\tau} = L_{\text{task}}(A_{\text{out}}(t, \tau - 1)) - L_{\text{task}}(A_{\text{out}}(t, \tau)). \tag{4.4}$$

For example, if the task is reconstruction of an image, the reward is positive if the new action (new fixation location) reduced the reconstruction error.

For each macrostep, we compute the marginal change in loss due to the whole macrostep:

$$\nu_t = L_{\text{task}}(A_{\text{out}}(t-1, T_1)) - L_{\text{task}}(A_{\text{out}}(t, T_1)). \tag{4.5}$$

The top layer is trained using the cumulative reward from all future macrosteps $\Phi_t = \sum_{i=t}^{T_2} \nu_i$, while the bottom layer is trained using the cumulative reward from all future microsteps $\Phi_{t,\tau} = \sum_{i=t+1}^{T_2} \nu_i + \sum_{j=\tau}^{T_1} \nu_{t,j}$.

We use an adjusted version of the baseline-based variance-reduction technique introduced in Sutton et al. (2000) and used in Mnih et al. (2014). We learn two separate baselines, $b_{t,\tau} = \mathbb{E}[\Phi_{t,\tau}]$ and $b_t = \mathbb{E}[\Phi_t]$, and use the baseline-removed cumulative rewards $\Phi_{t,\tau} - b_{t,\tau}$ and $\Phi_t - b_t$ for training.

The REINFORCE loss is given by

$$L_{\text{RL}} = -\sum_{t=1}^{T_2} \underbrace{\left(\log P(L_t | A_t; \theta_L)(\Phi_t - b_t) + \sum_{\tau=1}^{T_1} \log P(l_{t,\tau} | a_{t,\tau}; \theta_l)(\Phi_{t,\tau} - b_{t,\tau}) \right)}_{\text{Action log-probabilities}}. \tag{4.6}$$

As mentioned earlier, to allow exploration during training with REINFORCE, the locations at each macro- or microstep were the location network's output plus gaussian noise. Therefore, the logarithmic probability terms above reduce to the squared Euclidean distances between the mean and the sampled locations.

We combine the REINFORCE loss with a dense version of the task loss to get the combined loss function for the action networks:

$$L_{\text{action}} = \underbrace{L_{\text{RL}}}_{\text{Location networks}} + \underbrace{\sum_t \sum_\tau L_{\text{task}}(A_{\text{out}}(t, \tau))}_{\text{Action sub-system minus location networks}}. \tag{4.7}$$

For example, if the task is reconstruction, the second term in the combined loss allows minimization of the reconstruction error at every time step. Overall, the combined loss function increases the performance of the intermediate action vectors from step to step in the context of the task, producing more interpretable results.

Ideally the model would avoid generating locations exceeding the boundaries of the image. Several implementations of RAM (Mnih et al., 2014) use clipping or the hyperbolic tangent activation function. In practice, we found constraining the locations via an appropriate penalty to be more effective. We calculate a threshold c so that if a glimpse is centered c units away from the image boundary ($l \in [-1.0 + c, 1.0 - c]$), then the glimpse

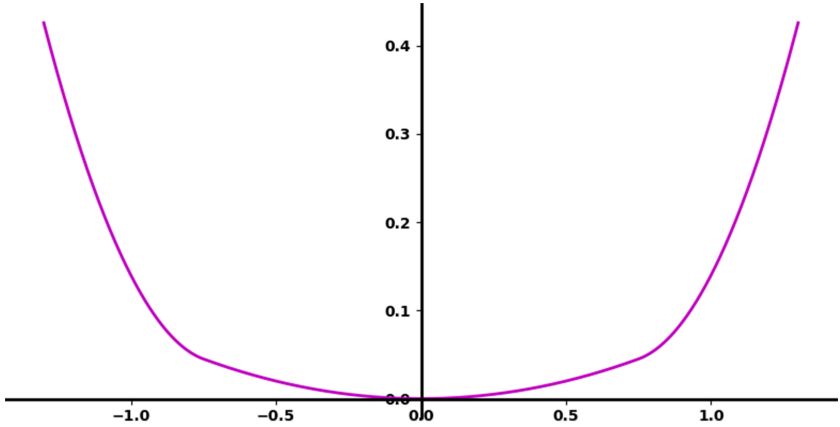


Figure 5: APC location penalty function for $c = 0.75$.

resides entirely within that boundary. We use a thresholded version of ℓ_2 normalization,

$$L_{\text{reg}}(l) = (\text{LRelu}(l - c))^2 + (\text{LRelu}(-l - c))^2 - 2(\alpha c)^2,$$

where LRelu is the leaky rectified linear unit (He et al., 2015) with $\alpha = 0.2$. The structure of this penalty, which is added to the loss function equation 4.7, can be seen in Figure 5.

4.4 Results. We first tested the APC model on the task of sequential part/location prediction and image reconstruction of objects in the following data sets[3]:

1. MNIST: Original MNIST data set of 10 classes of handwritten digits.
2. Fashion-MNIST (FMNIST): Instead of digits, the data set consists of 10 classes of clothing items.
3. Omniglot: 1623 hand-written characters from 50 alphabets, with 20 samples per character.
4. affNIST: MNIST digits embedded in a 40×40 pixel frame and transformed via random affine transformations.

For our APC models, we used three macro- and three microsteps (except four macrosteps for Omniglot and affNIST). A single dense layer, together with an initial random glimpse, was used to initialize the state and action vectors of the top level. While we evaluate the model on a reconstruction

[3] Code for the APC model is available at https://github.com/gklezd/apc-vision.

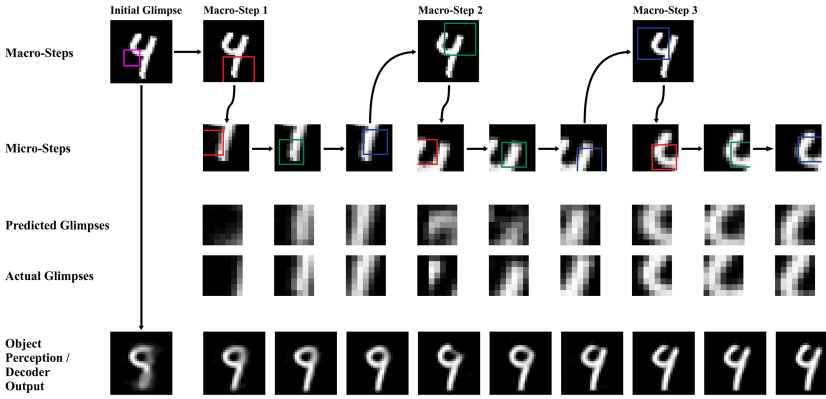


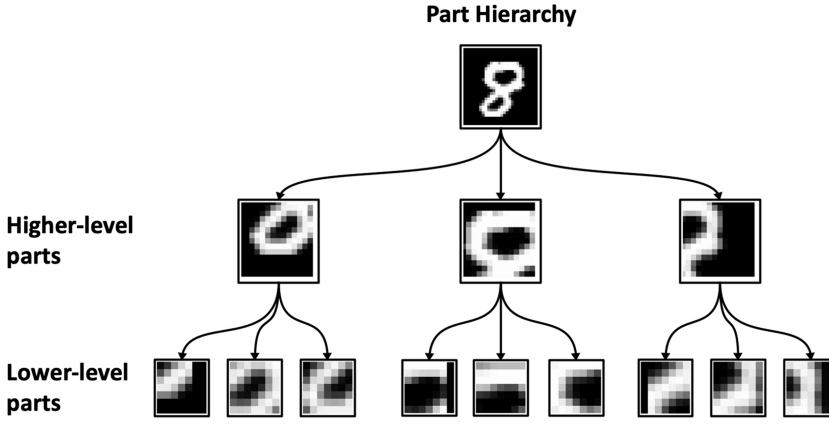
Figure 6: Learned two-level parsing strategy and an illustration of perceptual stability in the presence of eye movements. (First row) Initialization glimpse (purple box) and sampled top-level reference frames (red, green, blue boxes). (Second row) Sampled bottom-level parts within each top-level frame, (Third and fourth rows) Predicted versus actual parts/glimpses. (Fifth row) "Perception" of the model (object reconstructed from current network state) over time.

task, the architecture is general and can be used with various downstream tasks (see section 4.3.2).

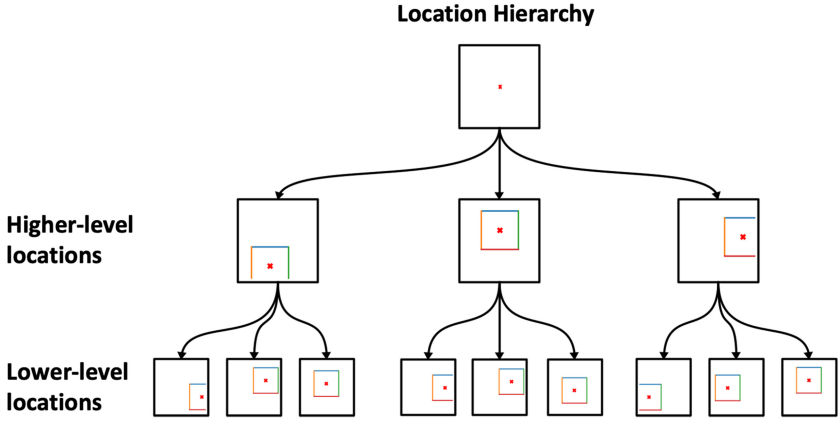
4.4.1 Parsing Images and Perceptual Stability. Figure 6 shows an example of a parsing strategy learned by a two-level APC model for an MNIST digit. The top level learned to cover the input image sufficiently, while the bottom level learned to parse subparts inside the reference frame computed by the higher level. Figure 6 also suggests an explanation for why human perception can appear stable despite dramatic changes in our retinal images as our eyes move to sample a scene: the last row of the figure shows how the model maintains a visual hypothesis that is gradually refined and does not exhibit the kind of rapid changes seen in the sampled images ("Actual Glimpses" in Figure 6).

Figure 7 shows a learned part-whole hierarchy for an MNIST input in the form of a parse tree of "parts" and "subparts" (here, strokes and mini-strokes) with locations. The model learns different parsing strategies for different classes of objects (see Figure 8).

4.4.2 Prediction of Parts and Pattern Completion. To investigate the predictive and generative ability of the model, we had the model "hallucinate" different parts of an object by setting the prediction error input to the lower-level network to zero. This disconnects the model from the input, forcing it to predict the next sequence of parts and "complete" the object. Figure 9a shows that the model has learned to generate plausible predictions of parts given an initial glimpse.



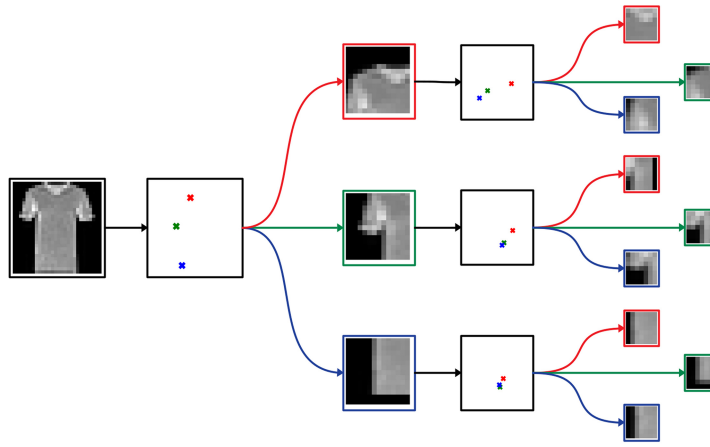
(a)



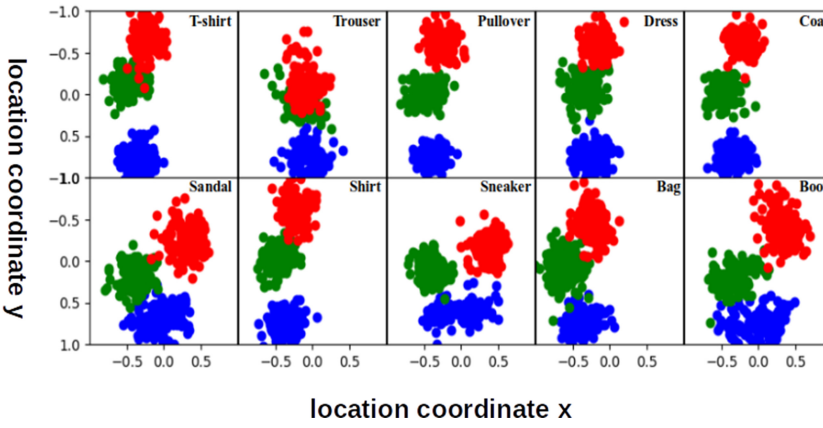
(b)

Figure 7: Example parse tree with inferred locations of parts. Hierarchy of (a) sampled parts and (b) sampled locations, inducing a hierarchy of reference frames.

4.4.3 Transfer Learning. We tested transfer learning for reconstruction of unseen character classes for the Omniglot data set. We trained a two-level APC model to reconstruct examples from 85% of classes from each Omniglot alphabet. The rest of the classes were used to test transfer: the trained model had to generate new "programs" (via the state and action hypernets) to predict parts for new character classes for each alphabet. The model successfully performed this task (see Table 1 and Figure 9b).



(a)



(b)

Figure 8: Class-based hierarchical representation of object parts and locations. (a) Parts and subparts recognized by a two-level APC network trained on the Fashion-MNIST (FMNIST) data set for an input image of a T-shirt. The order of sampled locations within each frame of reference is red, green, and blue. (b) Each panel shows the top-level part locations selected by the trained APC network in panel a for all classes. Note the differences in the network's action strategies between vertically symmetric items and footwear. The locations are normalized to the $[-1, 1]$ range, with -1 being the left-most (or top) and 1 the right-most (or bottom) edges of the image.

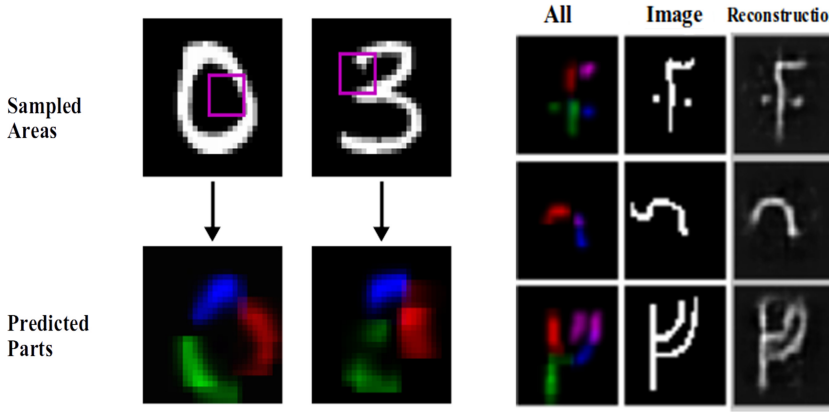


Figure 9: Prediction of parts, pattern completion and transfer learning. (a) Given only an initialization glimpse (purple box) for an input image (here, a 0 and a 3 from MNIST), an APC model trained on MNIST predicts its best guess of the parts of the object and their locations (colored segments in row below). (b) APC model trained on Omniglot can transfer its learned knowledge to predict parts of previously unseen character classes. (Left column). All the predicted parts. (Middle column) Input from a novel class. (Right column) APC model reconstruction.

Table 1: Ablation Studies: Reconstruction Mean-Squared-Error (per Pixel) for Different Models across Data Sets.

	MNIST	FMNIST	Om-Tst	Om-Trn	affNIST
RB	0.0097	0.0138	0.0222	0.0220	0.0116
APC-1	0.0072	0.0107	0.0205	0.0207	0.0055
APC-2	0.0070	0.0124	0.0191	0.0193	0.0063

Notes: See text for details. FMNIST, Om-Tst, and Om-Trn denote Fashion-MNIST, the Omniglot test and transfer data sets, respectively.

4.4.4 Ablation Studies. To test the utility of having two levels of abstraction, we compared the reconstruction performance of the two-level APC model (APC-2) to a one-level model (APC-1) and a randomized baseline model (RB), which samples glimpses (same size as APC-1 and APC-2) from T i.i.d. locations (T is the same value as for APC-1 and APC-2, that is, 9 for MNIST/FMNIST, 12 for Omniglot), extracts an average feature vector and feeds this to a feedforward network to reconstruct the image. As shown in Table 1, both APC models clearly outperform RB, indicating that they learn to sample glimpses in an intelligent manner. APC-2 and APC-1 have

Table 2: Ablation Studies: Number of Learnable Parameters for Different Models across Data Sets.

	MNIST	FMNIST	Om-Tst	Om-Trn	affNIST
RB	2.09M	2.09M	2.15M	2.15M	2.30M
APC-1	2.06M	2.06M	2.13M	2.13M	2.27M
APC-2	2.22M	2.22M	2.73M	2.73M	2.43M

Notes: See text for details. FMNIST, Om-Tst, and Om-Trn denote Fashion-MNIST, the Omniglot test and transfer data sets, respectively.

comparable performance on MNIST. APC-2 clearly outperforms APC-1 on the standard and transfer Omniglot tasks, demonstrating the advantage of dynamically generating "programs" to parse novel characters. APC-1 outperforms APC-2 and RB by a wide margin on the FMNIST task. For this data set, sampling the borders of the image is a very effective strategy. APC-1 also performs better on affNIST, a data set for which an effective strategy requires initial exploration to locate the digit. Note that APC-2 is more restricted in terms of sampled glimpse locations, since for each macrostep, these are confined within the respective top-level frame of reference. However, employing nested references frames endows APC-2 with the power of hierarchical abstraction and compositionality, which pays off in transfer learning tasks as seen in our Omniglot examples.

Table 2 shows the number of learnable parameters, which is comparable across models and data sets. Note that a comparison with models such as Attend, Infer, Repeat (AIR; Eslami et al., 2016) or variational autoencoders (VAEs; Kingma & Welling, 2014), also tasked with image reconstruction, would not be a fair comparison since these models have access to the entire image at once while the APC model has to learn how to intelligently sample only a subset of the input.

5 Application II: Hierarchical Planning

We now show that the same APC model we used above for learning part-whole hierarchies can also be used for a very different problem: learning hierarchical world models for efficient planning. We introduce a new compositional, scalable "multirooms" navigation task to illustrate this.

Consider the problem of navigating from any starting location to any goal location on a given floor of a large building such as the one in Figure 10A (gray: walls; blue circle: agent; green square: current goal). In the traditional (nonhierarchical) reinforcement learning (RL) approach for solving such a problem, the states are the discrete locations in the grid, and the actions are going north (N), east (E), south (S), or west (W). A large reward ($+10$) is received at the goal location, with a small negative reward (-0.1) for each action to encourage shortest paths.

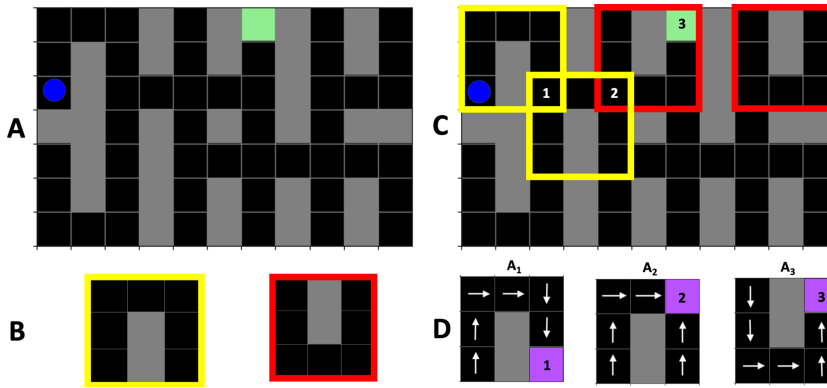


Figure 10: The multi-rooms navigation problem and state-action hierarchy. The problem of navigating in a large building with multiple rooms and corridors (A) (blue: agent location; gray: walls; green: goal) can be reduced to planning using high-level states (B, C) and high-level actions (D). See text for details.

5.1 Problems with Traditional Reinforcement Learning. The traditional RL approach suffers from the following problems: The first is *sample inefficiency.* As the environment gets larger, the number of interactions with the environment required to learn the value function becomes impractically large. The second is *risk of catastrophic consequences:* taking actual actions in the real world to estimate the value function might have catastrophic consequences (injury or death). The third is *inflexibility:* for every new goal or task, a new value function needs to be learned. Hierarchical reinforcement learning and safe reinforcement learning have been proposed as ways of addressing some of these problems (Sutton et al., 1999; Botvinick et al., 2009; Garcıa & Fernández, 2015; Kulkarni et al., 2016; Nachum et al., 2019; Hafner et al., 2022). The APC model differs from previous approaches in asserting the need for abstracting not only actions (policies) but also state transition functions in a hierarchical manner, as depicted in Figure 1.

5.2 How the APC Model Solves These Problems. Just as an object (e.g., an MNIST digit) consists of the same parts (e.g., strokes) occurring at different locations, the multi-rooms environment in Figure 10A is also made up of the same two components ("Room types" R1 and R2), shown in Figure 10B, occurring at different locations (some example locations highlighted by yellow and red boxes in Figure 10C).

These components form part of the higher-level states in the APC and are defined by state embedding vectors (say, $S1$ and $S2$), which can be trained to generate, via the hypernet H_s (see Figure 1), the lower-level transition functions f_s for rooms R1 and R2, respectively. Next, similar to how the APC

model was able to reconstruct an image using top-level action embedding vectors to generate policies and actions (locations) to compose parts using strokes, the APC model can compute top-level action embedding vectors A_i (option vectors) for the multi-rooms world that generate, via hypernet H_a (see Figure 1), bottom-level policies f_a that produce primitive actions (N, E, S, W) to reach a goal i encoded by A_i (note that we use the subscript for A here and in the next two sections to denote a particular goal rather than time).

5.3 Local Reference Frames Allow Policy Reuse and Transfer. Figure 10D illustrates the bottom-level policies for three such action-embedding vectors A_1, A_2, and A_3, which generate policies for reaching goal locations 1, 2, and 3, respectively. Note that the A_i are defined with respect to higher-level state $S1$ or $S2$ corresponding to room type R1 or R2. Defining these policies to operate within the local reference frame of the higher-level state $S1$ or $S2$ (regardless of global location in the building) confers the APC model with enormous flexibility because the same policy can be reused at multiple locations to solve local tasks (here, reach subgoals within R1 or R2).

For example, to solve the navigation problem in Figure 10C, the APC model only needs to plan and execute 3 higher-level actions or options: A_1 followed by A_2 followed by A_3, compared to planning a sequence of 12 lower-level actions to reach the same goal. Finally, since the A_i embedding space of options is continuous, the APC model offers an unprecedented opportunity to exploit properties of this embedding space (such as smoothness) to interpolate or extrapolate to create and explore new options for transfer learning; this possibility will be explored in future work.

5.4 Results. For simplicity, we assume the higher-level states capture 3×3 local reference frames and are defined by an embedding vector generating the transition function for "room type" R1 or R2, along with the location for this local reference frame in the global frame of the building. The lower-level action network is trained to map a higher-level action embedding vector A_i to a lower-level policy that navigates the agent to a particular goal location i within R1 or R2. For the current example, eight embedding vectors $A_1, \ldots, A_8$ were trained, using REINFORCE-based RL (Williams, 1992) to generate via the hypernet H_a eight lower-level policies to navigate to each of the four corners of room types R1 and R2. The higher-level state network F_s was trained to predict the next higher-level state (decoded as an image of room type R1 or R2, plus its location) given the current higher-level state and higher-level action.

The trained higher-level state network F_s was used for planning at each step a sequence of four higher-level actions using "random-sampling shooting" model-predictive control (MPC) (Richards, 2004): random state-action trajectories of length 4 were generated using F_s by starting from the current

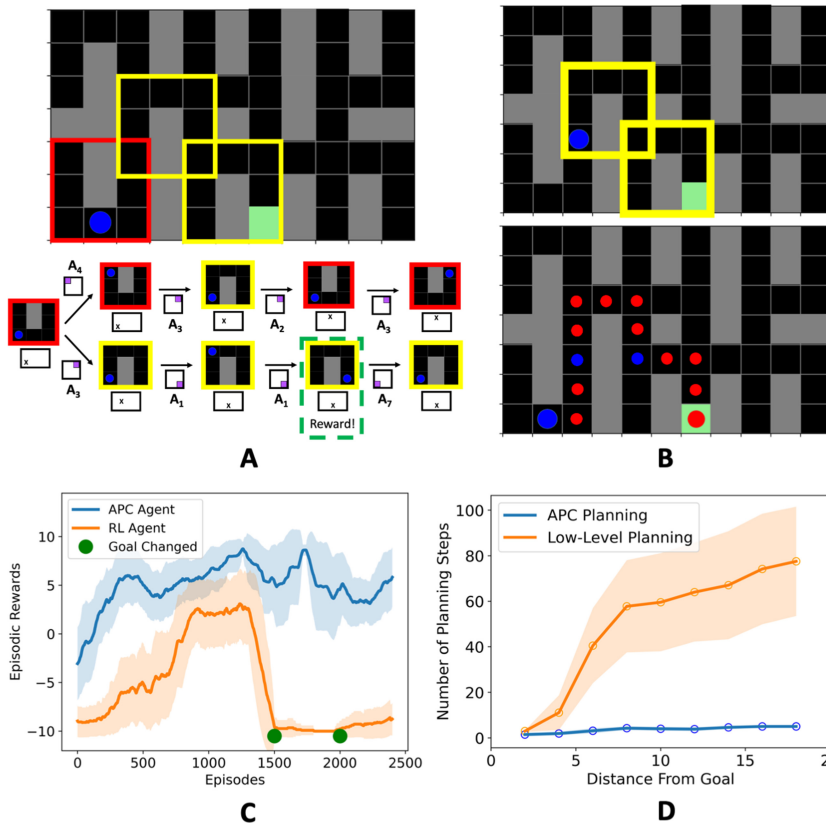


Figure 11: Planning and model predictive control. To navigate to the green goal in the top part of panel A, the agent (blue) uses its learned high-level state network to sample N high-level state-action trajectories ($N = 2$ in the bottom part of panel A), picks the sequence with highest total reward, executes this sequence's first action to reach the location in the top part of panel B and repeats to reach the goal with three high-level actions (see the bottom part of panel B). Small red dot: intermediate location; small blue dot: intermediate goal. In panel A, bottom, a high-level state is depicted by predicted R1 or R2 image and its location X in the global frame, action by its goal (purple) in a square local frame. (C, D) APC model performance. See the text for details.

state and picking one of the four random actions A_i for each next state; the action sequence with the highest total reward was selected, and its first action was executed. Figures 11A and 11B show an example of this high-level planning and MPC process using the trained APC model. Such an approach to planning is closely related to the ideas of planning by inference (Attias,

2003; Verma & Rao, 2005, 2006; Botvinick & Toussaint, 2012; Levine, 2018) and active inference (Friston et al., 2011, 2017; Fountas et al., 2020).

We compared the two-level APC model with both a heuristic lower-level-only planning algorithm and a REINFORCE-based RL algorithm using primitive states and actions. The task involved navigating to a randomly selected goal location in a building environment (as in Figure 10A), with the goal location changing after some number of episodes.

Figure 11C shows how the APC model, after an initial period spent on learning the hypernet H_a to generate the lower-level options, is able to cope with goal changes and successfully navigate to each new goal by sequencing high-level actions (+10 reward for goal; −0.1 per primitive action). The RL algorithm experiences a drop in performance after a goal change and does not recover even after 500 episodes.

Figure 11D demonstrates the efficacy of APC's higher-level planning compared to lower-level planning (MPC using random sequences of four primitive future actions; Euclidean distance heuristic): the average number of planning steps to reach the goal increases dramatically for larger distances from the goal for lower-level compared to higher-level planning.

Additional results and details regarding the application of the APC model to hierarchical planning can be found in the Supplementary Information section online at https://doi.org/10.1162/neco_a_01627.

6 Discussion and Conclusion

Our results represent a first proof-of-concept demonstration of how the APC model can offer a unified approach to modeling a diverse set of problems that have previously required very different approaches, for example, active vision and part-whole learning (Hinton, 2021), learning reference frames (Hawkins, 2021), and planning using state-action hierarchies (Hutsebaut-Buysse et al., 2022).

The APC model for active vision employs eye movements for visual sampling, and performs end-to-end learning and parsing of part-whole hierarchies from images. The incremental parsing of a scene by the APC model using state-action recurrent networks also suggests an explanation for why our visual perception remains stable despite the staccato nature of our eye movements. We showed how the same framework can also be used to solve a complex navigation task by modeling a large environment as being composed of simpler components and using hierarchical states and actions for efficient planning.

Our results relied on assumptions such as a fixed number of time steps at each level, a two-level hierarchy, hard-coded glimpse operators for active vision, and preidentified higher-level states and actions for planning. Future work will involve relaxing these assumptions, comparing the APC model to other part-whole learning approaches using different metrics, and

employing more sophisticated planning and RL methods to scale the model to more complex tasks.

We expect adding more levels to the model to boost its capability by allowing it to learn even more abstract state-action representations; this capability is not limited by sensor resolution or action space dimensionality, though the benefits of adding additional levels will depend on the complexity and compositional nature of the environment and problem. Adding more levels would potentially allow the model to handle more complex environmental dynamics and solve more complex tasks via a hierarchical divide-and-conquer strategy, breaking down the dynamics into compositions of simpler transition functions and breaking down long and complex sequences of low-level actions into simpler compositions of macro-actions. For example, adding a third level to our network for the navigation problem would allow learning of third-level abstract actions composed of sequences of macro-actions; these could be used, for example, for hierarchical planning between buildings in a larger campus environment. More broadly, within the domain of reinforcement learning problems, adding a third level to the APC model could enable learning sequences of macro-actions to solve complex problems composed of multiple subproblems, each solved by a macro-action. In the image parsing problem, adding a third level could, for example, allow frames of reference of larger spatial extent, transformations of the agent's point of view in three-dimensional space relative to the object (such that each such transformation would result in a different two-dimensional view of the object), and other higher-level abstractions of object and scene properties.

Models based on transformers (Vaswani et al., 2017) have recently demonstrated tremendous promise in various vision-related tasks (Dosovitskiy et al., 2021). Soft attention models such as transformers and hard attention models such as ours represent two approaches with significant conceptual differences; the first assumes access to the whole input at once, while the latter uses a policy to sequentially and intelligently sample only parts of the input. The plain version of transformers scales quadratically with the input, while sequential models such as APC or RAM (Mnih et al., 2014) can be interpreted as having constant size regardless of the input size. While numerous recent approaches (Dao et al., 2022) are aimed at reducing the computational demands of transformers, an alternative approach (and a promising future research direction) would combine them into a hybrid model. Such a model would dynamically combine information (soft attention) from a restricted subset of the input (hard attention).

Another potentially fruitful direction for future research is to formulate a fully probabilistic version of the APC model and derive methods for approximate inference of posterior probability distributions over states and actions (see Levine, 2018 and Friston et al., 2017). Such a probabilistic version of the APC model would allow uncertainty-based reasoning and action selection within a hierarchical POMDP framework.

We began this article with the goal of identifying a unifying computational framework inspired by the theoretical problems posed by various researchers in section 2. We regard APC as one such framework, capable of tackling a diverse set of problems that previously were studied in different subareas of AI and cognitive science (vision, reinforcement learning, navigation, and compositional learning). Thus, rather than comparing performance on a single benchmark or problem, we provided proof-of-concept results demonstrating the diverse computational abilities of the same framework across multiple domains.

The APC model is inspired by the growing interest in predictive coding as a model for understanding cortical computation (Rao & Ballard, 1999; Keller & Mrsic-Flogel, 2018; Jiang & Rao, 2022b). The APC model augments the traditional predictive coding framework with state-action hierarchies and shows how these can be learned using hypernetworks. More broadly, our results showing diverse applicability of the same APC framework across multiple domains lend strong computational support to the long-cherished and much-discussed hypothesis that there may be a common computational principle operating across the cortex (Creutzfeldt, 1977; Mountcastle, 1978; Hawkins, 2021; Rao, 2022).

Acknowledgments

We thank the reviewers for their constructive suggestions. We also thank Ares Fisher, Preston Jiang, and Prashant Rangarajan for helpful discussions during the course of this work. This research was supported by the National Science Foundation EFRI grant 2223495, the Defense Advanced Research Projects Agency under contract HR001120C0021, a Weill Neurohub Investigator grant, a UW + Amazon Science Hub grant, and a Frameworks grant from the Templeton World Charity Foundation. The opinions expressed in this article are our own and do not necessarily reflect the views of the funders.

References

Ali, A., Ahmad, N., de Groot, E., Johannes van Gerven, M., & Kietzmann, T. (2022). Predictive coding is a consequence of energy efficiency in recurrent neural networks. *Patterns*, *3*(12), 100639. 10.1016/j.patter.2022.100639

Attias, H. (2003). Planning by probabilistic inference. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics* (pp. 9–16).

Bacon, P.-L., Harb, J., & Precup, D. (2016). The option-critic architecture. *CoRR*, abs/1609.05140.

Botvinick, M., & Toussaint, M. (2012). Planning as inference. *Trends in Cognitive Sciences*, *16*(10), 485–488. 10.1016/j.tics.2012.08.006

Botvinick, M. M., Niv, Y., & Barto, A. G. (2009). Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, *113*(3), 262–280. 10.1016/j.cognition.2008.08.011

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M. M., & Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *CoRR*, abs/1901.11390.

Constantinescu, A., O'Reilly, J., & Behrens, T. (2016). Organizing conceptual knowledge in humans with a gridlike code. *Science*, *352*(6292), 1464–1468. 10.1126/science.aaf0941

Creutzfeldt, O. D. (1977). Generality of the functional structure of the neocortex. *Naturwissenschaften*, *64*, 507–517. 10.1007/BF00483547

Dao, T., Fu, D., Ermon, S., Rudra, A., & Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with IO-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems*, *35* (pp. 16344–16359). Curran.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, . . . Houlsby, N. (2021). An image is worth 16 × 16 words: Transformers for image recognition at scale. In *Proceedings of the 9th International Conference on Learning Representations*.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, *338*, 1202–1205. 10.1126/science.1225266

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., & Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), *Advances in neural information processing systems*, *29*. Curran.

Ferguson, K. A., & Cardin, J. A. (2020). Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, *21*(2), 80–92. 10.1038/s41583-019-0253-y

Fountas, Z., Sajid, N., Mediano, P., & Friston, K. (2020). Deep active inference agents using Monte-Carlo methods. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, *33* (pp. 11662–11675). Curran.

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., & Pezzulo, G. (2017). Active inference: A process theory. *Neural Computation*, *29*(1), 1–49. 10.1162/NECO_a_00912

Friston, K., & Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *364*, 1211–1221. 10.1098/rstb.2008.0300

Friston, K., Mattout, J., & Kilner, J. (2011). Action understanding and active inference. *Biological Cybernetics*, *104*(1), 137–160. 10.1007/s00422-011-0424-z

Galanti, T., & Wolf, L. (2020). On the modularity of hypernetworks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems*, *33* (pp. 10409–10419), Curran.

Garcıa, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480.

George, D., & Hawkins, J. (2009). Towards a mathematical theory of cortical microcircuits. *PLOS Computational Biology*, *5*, e1000532. 10.1371/journal.pcbi.1000532

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., . . . Phoenix, D. S. (2017). A generative vision model that trains with high data

efficiency and breaks text-based captchas. *Science*, *358*(6368), eaag2612. 10.1126/science.aag2612

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., . . . Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning*.

Guntupalli, J. S., Raju, R. V., Kushagra, S., Wendelken, C., Sawyer, D., Deshpande, I., . . . George, D. (2023). *Graph schemas as abstractions for transfer learning, inference, and planning*. arXiv:2302.07350.

Ha, D., Dai, A. M., & Le, Q. V. (2017). Hypernetworks. In *Proceedings of the 5th International Conference on Learning Representations*.

Hafner, D., Lee, K.-H., Fischer, I., & Abbeel, P. (2022). *Deep hierarchical planning from pixels.* arXiv:2206.04114.

Hawkins, J. (2021). *A thousand brains: A new theory of intelligence*. Basic Books.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034).

Hinton, G. E. (2021). How to represent part-whole hierarchies in a neural network. *CoRR*, abs/2102.12627.

Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing. In *Proceedings of the 6th International Conference on Learning Representations*.

Hutsebaut-Buysse, M., Mets, K., & Latré, S. (2022). Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, *100*, 172–221. 10.3390/make4010009

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28*. Curran.

Jiang, L., Gklezakos, D. C., & Rao, R. P. N. (2021). *Dynamic predictive coding with hypernetworks.* bioRxiv:2021.02.22.432194.

Jiang, L., & Rao, R. P. N. (2022a). *Dynamic predictive coding: A new model of hierarchical sequence learning and prediction in the cortex.* bioRxiv:2022.06.23.497415.

Jiang, L., & Rao, R. P. N. (2022b). Predictive coding theories of cortical function. In *Oxford research encyclopedia of neuroscience*. Oxford University Press.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1–2), 99–134. 10.1016/S0004-3702(98)00023-X

Keller, B., & Mrsic-Flogel, T. D. (2018). Predictive processing: A canonical cortical computation. *Neuron*, *100*(2), 424–435. 10.1016/j.neuron.2018.10.003

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations.*

Kosiorek, A., Sabour, S., Teh, Y. W., & Hinton, G. E. (2019). Stacked capsule autoencoders. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, *32*. Curran.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *25*. Curran.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., & Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), *Advances in neural information processing systems*, *29*. Curran.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*, e253.

Larkum, M. E., Senn, W., & Lüscher, H.-R. (2004). Top-down dendritic input increases the gain of layer 5 pyramidal neurons. *Cerebral Cortex*, *14*(10), 1059–1070. 10.1093/cercor/bhh065

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909.

Lewis, M., Purdy, S., Ahmad, S., & Hawkins, J. (2019). Locations in the neocortex: A theory of sensorimotor object recognition using cortical grid cells. *Frontiers in Neural Circuits*, *13*. 10.3389/fncir.2019.00022

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, *21*(6), 335–346. 10.1038/s41583-020-0277-3

McAdams, C. J., & Maunsell, J. H. R. (1999). Effects of attention on orientation–tuning functions of single neurons in macaque cortical area v4. *Journal of Neuroscience*, *19*, 431–441. 10.1523/JNEUROSCI.19-01-00431.1999

Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*, *27*. Curran.

Moser, E., Moser, M., & McNaughton, B. (2017). Spatial representation in the hippocampal formation: A history. *Nature Neuroscience*, *20*(11), 1448–1464. 10.1038/nn.4653

Mountcastle, V. (1978). An organizing principle for cerebral function: The unit model and the distributed system. In G. Edelman & V. Mountcastle (Eds.), *The mindful brain* (pp. 7–50). MIT Press.

Mumford, D. (1992). On the computational architecture of the neocortex: II The role of cortico-cortical loops. *Biological Cybernetics*, *66*, 241–251. 10.1007/BF00198477

Nachum, O., Tang, H., Lu, X., Gu, S., Lee, H., & Levine, S. (2019). *Why does hierarchy (sometimes) work so well in reinforcement learning?* arXiv:1901.10618 (cs.LG).

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., . . . Bojanowski, P. (2023). *DINOv2: Learning robust visual features without supervision*. arXiv:2304.071193(cs,CV).

O'Keefe, R. J., & Dostrovsky, J. (1971). The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, *34*,171–175.

Pearson, K. (1901). LIII. *On lines and planes of closest fit to systems of points in space*, *2*(11), 559–572.

Rao, R. P. N. (1999). An optimal estimation approach to visual perception and learning. *Vision Research*, *39*(11), 1963–1989. 10.1016/S0042-6989(98)00279-X

Rao, R. P. N. (2010). Decision making under uncertainty: A neural model based on partially observable Markov decision processes. *Frontiers in Computational Neuroscience*, *4*, 146. 10.3389/fncom.2010.00146

Rao, R. P. N. (2022). *A sensory-motor theory of the neocortex based on active predictive coding*. bioRxiv:2022.12.30.522267.

Rao, R., & Ballard, D. (1997). Dynamic model of visual recognition predicts neural response properties in the visual cortex. *Neural Computation*, *9*(4), 721–763. 10.1162/neco.1997.9.4.721

Rao, R., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, *2*,79–87. 10.1038/4580

Richards, A. (2004). *Robust constrained model predictive control*. PhD diss., MIT.

Sabour, S., Frosst, N., & Hinton, G. E. (2017). *Dynamic routing between capsules*. arXiv:1710.09829.

Salinas, E., & Abbott, L. F. (1996). A model of multiplicative neural responses in parietal cortex. *Proceedings of the National Academy of Sciences*, *93*, 11956–1196. 10.1073/pnas.93.21.11956

Salinas, E., & Sejnowski, T. J. (2001). Gain modulation in the central nervous system: Where behavior, neurophysiology, and computation meet. *Neuroscientist*, *7*, 430–440. 10.1177/107385840100700512

Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., . . . Locatello, F. (2023). *Bridging the gap to real-world object-centric learning*. arXiv:2209.14860 (cs.CV).

Sherman, S. M., & Guillery, R. W. (2013). *Functional connections of cortical areas: A new view from the thalamus*. MIT Press.

Shine, J. M., Müller, E. J., Munn, B., Cabral, J., Moran, R. J., & Breakspear, M. (2021). Computational models link cellular mechanisms of neuromodulation to large-scale neural dynamics. *Nature Neuroscience*, *24*(6), 765–776. 10.1038/s41593-021-00824-6

Stroud, J. P., Porter, M. A., Hennequin, G., & Vogels, T. P. (2018). Motor primitives in space and time via targeted gain modulation in cortical networks. *Nature Neuroscience*, *21*(12), 1774–1783. 10.1038/s41593-018-0276-0

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, & K. Müller (Eds.), *Advances in neural information processing systems*, *12*. MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1), 181–211. 10.1016/S0004-3702(99)00052-1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, . . . Polosukhin, I. (2017). Attention is all you need. In I. Guyon, Y. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*, *30*. Curran.

Verma, D., & Rao, R. P. (2005). Goal-based imitation as probabilistic inference over graphical models. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems*, *18*. MIT Press.

Verma, D., & Rao, R. P. N. (2006). Planning and acting in uncertain environments using probabilistic inference. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2382–2387).

Wedel, M., Pieters, R., & van der Lans, R. (2022). Modeling eye movements during decision making: A review. *Psychometrika*, *88*(2), 697–729. 10.1007/s11336-022-09876-4

Whittington, J. C. R., & Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local Hebbian synaptic plasticity. *Neural Computation*, *29*(5), 1229–1262. 10.1162/NECO_a_00949

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T., & Wang, X.-J. (2019). Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience*, *22*(2), 297–306. 10.1038/s41593-018-0310-2

Zipser, D., & Andersen, R. A. (1988). A backpropagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, *331*, 679–684. 10.1038/331679a0

Received May 22, 2023; accepted September 20, 2023.