Learning Hierarchical Abstractions of Complex Dynamics using Active Predictive Coding

Vishwas Sathish
Paul G. Allen School of
Computer Science & Engineering,
University of Washington, Seattle
vsathish@cs.washington.edu

Rajesh P.N. Rao
Paul G. Allen School of
Computer Science & Engineering,
University of Washington, Seattle
rao@cs.washington.edu

***Abstract*—Learning robot dynamics for real-world environments is difficult, if not impossible, using a single monolithic state transition network. A more practical approach, inspired by sensory-motor hierarchies in the mammalian brain, is to model complex state transition dynamics as a sequence of simpler dynamics, which in turn are modeled using even simpler dynamics, and so on. We demonstrate the efficacy of such an approach using Active Predictive Coding (APC), a structured model for learning hierarchical dynamics and hierarchical policies that is inspired by the neocortex. APC leverages higher-level abstract states as priors for generating lower-level transition functions using hypernetworks. Additionally, APC learns a transition function for the dynamics between abstract states, leading to a tractable approach to learning complex dynamics in terms of a hierarchy of dynamically generated transition functions. We apply our approach to compositional navigation problems and show its capability for rapid planning and transfer to novel scenarios. In both traditional grid-world-style navigation problems as well as in the more complex Habitat vision-based navigation domain, APC learns to abstract the transition dynamics of a robot within and between different rooms in an unsupervised manner. Our results suggest that APC offers a promising framework for learning complex robot dynamics using a compositional and hierarchical approach.**

I. Introduction

Composing existing skills and knowledge to creatively generate solutions for new and complex problems is a fundamental attribute of human intelligence. Recent advances in generative AI and large language models are beginning to demonstrate attributes of human-like intelligence but fail at simple tasks like adding a few three digit numbers (Bubeck et al. [6], Bender et al. [5], Schmidhuber [39]) that rely on application of compositional knowledge. Similar observations hold true for traditional reinforcement learning (RL) and embodied AI agents (Lake et al. [27]).

There have been recent advances in hierarchical reinforcement learning with novel neural network architectures (Hafner et al. [19], Levy et al. [29], Kulkarni et al. [26]), but a wide gap still persists when it comes to using structured architectures for leveraging compositionality for rapid transfer of knowledge of robot dynamics and skills. These questions motivate us to ask - what are the fundamental computational principles behind biological neural networks, that allow them to exhibit compositional learning? And what would be the key takeaways from biological architectures for AI algorithms to learn and exhibit rapid generalization and transfer?

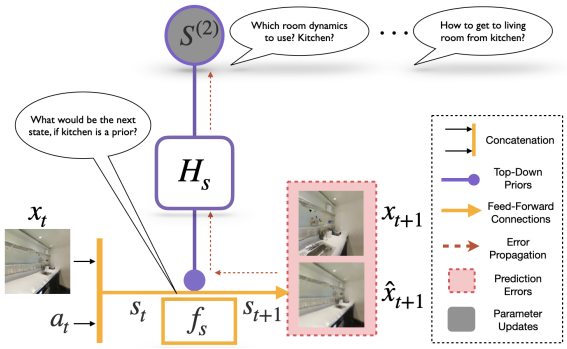

Fig. 1. **Learning and Inferring Hierarchical Dynamics**. A learnable higher-level latent state $S^{(2)}$ generates, using a hypernet H_s, a lower-level transition function f_s mapping current input x_t and lower-level action a_t to next input. Here, the input is an image (taken from the Habitat environment). Active predictive coding (APC) uses the sequence of prediction errors between the model prediction and the true output to update and infer in real time the higher-level latent state $S^{(2)}$ (here, representing an estimate for the current room). Complex dynamics is abstracted in an unsupervised manner and reused to model dynamics in other similar situations. Moreover, such abstractions allow hierarchical planning such as navigating at the level of rooms rather than using primitive actions, resulting in significant savings.

Predictive coding theories have garnered increasing attention in recent years as computational models of how the brain perceives and acts in the real world (Rao and Ballard [35], Friston and Kiebel [12], Keller and Mrsic-Flogel [24], Jiang and Rao [23]). In predictive coding, different areas of the neocortex together implement a hierarchical generative model of the world. Feedback connections from a higher to a lower level predict lower-level responses, and the prediction errors propagate via feedforward connections to update higher-level estimates. While the original formulation of predictive coding ignored actions, the recently introduced framework of active predictive coding (APC) [36] recognizes the importance of using actions for learning and modeling the dynamics of complex environments.

In Section II, we describe an algorithm to learn state abstractions, only using random trajectory rollouts of an agent (Fig. 1). In Section III we present results for abstract state inference and hierarchical planning. Finally, we present initial

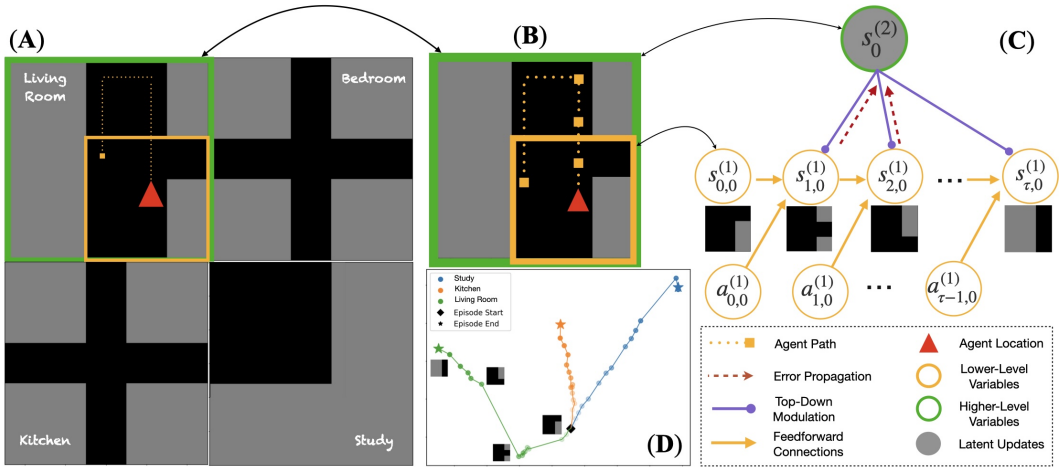

Fig. 2. **Inferring higher-level abstract states from lower-level dynamics**. **(A)** A home environment composed of simpler and, possibly repeating 5x5 elements ("rooms"). Gray areas represent walls or regions unreachable by the agent. The rooms are separated to show independent dynamics and an opportunity for reuse of transition functions between similar rooms (e.g., parts of the bedroom and kitchen). **(B)** The rooms correspond to reusable local dynamics that can be abstracted as the higher-level latent state vector $s^{(2)}$ and inferred by APC as the agent accumulates information from interactions with the local environment. **(C)** Latent state inference by unrolling the state transition graphical model over time. **(D)** 2-D TSNE plot of successive updates to a $d = 32$ dimensional latent state vector as the agent explores a room for $\tau = 15$ time steps. Note the convergence of inference to different parts of the latent space for different rooms

results from our ongoing work on a scalable APC model for complex environments like Habitat (Savva et al. [37]). Additional experiments and results are presented in the supplementary section. To summarize, the main contributions of this paper are:

1) A new compositional framework for learning complex dynamics using a hierarchy of simpler transition functions based on hypernetworks and APC.
2) A new approach for learning hypernetworks for generating transition functions on the fly via prediction errors.
3) A scalable APC architecture geared towards efficient planning and hierarchical state inference in real-world robotic environments.

II. Hierarchical Transition Dynamics with APC

In this section, we demonstrate how state abstractions and hierarchical transition functions can be learned by considering a 2-level APC model (Figure 1; Supplementary Figure 8). For the next two subsections, we assume a general case of top-down modulation where a hypernetwork generates K weights $\mathbf{w} = [w_1, w_2, ..., w_k]$ for composing a transition function as a weighted sum of K learnable matrices $\mathbf{M}$. Details regarding the parameterization and other versions of top-down modulation can be found in the supplementary section VI.

A. Inference

Consider an agent exploring its environment using actions defined by an exploration policy π. To make the example more concrete, assume that the agent is in a home environment made of rooms (kitchen, bedroom, etc.), as shown in Figure 2(A). A sequence of observations are generated from the sensory apparatus of the agent as it explores the environment. We assume that the underlying states are partially observable, resulting in a trajectory of observed states and actions over τ timesteps: $\mathcal{T}_{a\sim\pi} = \{s_0, a_0, s_1, a_1, ..., s_\tau\}$ [1]. Throughout the paper, we assume that the internal states s_t are based on encoded representations of inputs x_t (Figure 1) and integrate historically observed inputs via the recurrent network, a formulation in line with the recent trends in model-based RL (Hafner et al. [17, 18]). However, in APC, this recurrent network (implementing the lower-level transition function) is generated on the fly by the current higher-level abstract state $s^{(2)}$. Formally, $s_{t+1} \sim P(s_{t+1}|s_t, a_t, s^{(2)})$. Since our hierarchical transition models are task-independent, the rewards obtained in any particular task do not directly affect the transition models. However, in future work, we intend to explore incorporating reward prediction (in addition to state prediction) at the lower-level (Hafner et al. [18]).

The inference process involves making updates to beliefs over what room the agent is located in, i.e.. a kitchen, bedroom, etc., as evidence accumulates over an episode. This is formally computed by inferring the abstract high-level state $\mathbf{s}^{(2)}$ based on the minimization of prediction loss using gradient updates for each lower-level time step t (Equations 1, 2). The graphical model depicting this process is shown in Figure 2(C). A TSNE plot of $\mathbf{s}^{(2)}$ converging over specific episodes to represent different rooms is illustrated in 2(D). We investigate properties of the latent $\mathbf{s}^{(2)}$ space in Section III and provide further details in supplementary section VII.

$$\mathcal{L}_{t,s^{(2)}} = ||\hat{s}^{(1)}_{t+1} - s^{(1)}_{t+1}||^2_2 + \lambda||\mathbf{s}^{(2)}_T||^2_2 \quad (1)$$

[1]For convenience and readability of equations, we omit the subscripts and superscripts for variables throughout the paper, unless necessary: $s^{(1)}_{t,T}$, the lower-level state at time t and at a higher-level time period T, is replaced with s_t. Similarly, the higher-level state $s^{(2)}_T$ is replaced with $s^{(2)}$, when T remains constant.

$$\mathbf{s}^{(2)} \leftarrow \mathbf{s}^{(2)} - \eta \nabla_{s^{(2)}} \mathcal{L}_{t,s^{(2)}} \quad (2)$$

The first term in equation 1 computes the prediction loss. We typically use a decoder to transform predictions to the original observation space. The second term is an L_2 constraint on the abstract states which we found to work better in practice. $\eta = 0.05$ is the inference learning rate which is kept higher than the model learning rates. During the inference process, no update is made to the model parameters (Supplementary Figure 8(A)).

B. Learning

Learning the parameters of the hierarchical model is straightforward (Supplementary Figure 8(B)). After running the inference process for τ steps, latent states $s^{(2)}$ are frozen and used as inputs to the hypernetwork. For the same set of observations used during inference, prediction errors for τ timesteps are accumulated and the model parameters $\theta_{\mathcal{H}}$ and θ_f are updated in an unsupervised manner (here θ_f are the basis matrices $\mathbf{M}$ for composing transition functions (or RNN parameters when using an embedding method - Supplementary Section VI).

$$\mathcal{L}_\theta = \sum_{t=1}^{\tau} ||\hat{s}^{(1)}_{t+1} - s^{(1)}_{t+1}||^2_2 \quad (3)$$

$$\theta_{\mathcal{H}} \leftarrow \theta_{\mathcal{H}} - \eta_{\mathcal{H}} \nabla_{\theta_{\mathcal{H}}} \mathcal{L}_\theta; \quad \theta_f \leftarrow \theta_f - \eta_f \nabla_{\theta_f} \mathcal{L}_\theta \quad (4)$$

C. Scaling APC to Complex Environments

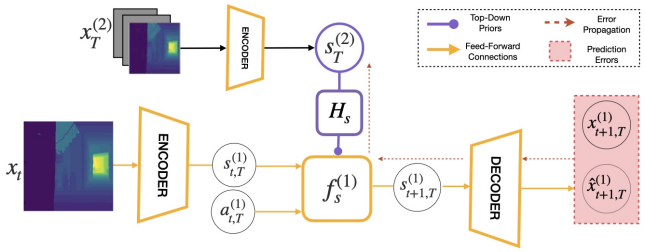

Fig. 3. **Scaling APC**: A scalable version of APC for complex image-based environments (original formulation in Supplementary Figure 8). The core idea of using abstract states $s^{(2)}$ to generate/modulate lower-level transition dynamics f_s remains the same. Here, instead of solely relying on prediction errors, an encoder is used for amortized inference to directly infer the abstract state. Multi-timestep inputs at the higher level $x^{(2)}$ are lower-resolution versions (24 x 24 images) of the lower-level inputs x_t (64 x 64 images).

Figure 3 depicts a modified version of the hypernetwork formulation for APC which relies on an RNN with a top-down embedding input. Prior work (Galanti and Wolf [13]) has shown that the embedding approach is functionally equivalent to using a hypernetwork, in that the modularity is emulated using an embedding vector:

$$\mathbf{e}_T = \mathcal{H}_\theta(\mathbf{s}^{(2)}_T) \quad (5)$$

$$\mathbf{h}_t = \tanh(\mathbf{s}_t W_1 + b_1 + \mathbf{h}_{t-1} W_2 + b_2 + \mathbf{e}_T W_3 + b_3) \quad (6)$$

$$\hat{s}_{t+1} = ReLU(\mathbf{h}_t W_4 + b_4) \quad (7)$$

While the original formulation of APC relied on $s^{(2)}$ updates via backpropagation of prediction errors, here we use a simple feedforward encoder to directly infer $s^{(2)}$ from a sequence of image inputs (amortized inference), providing significant improvements in training time and parallel processing.

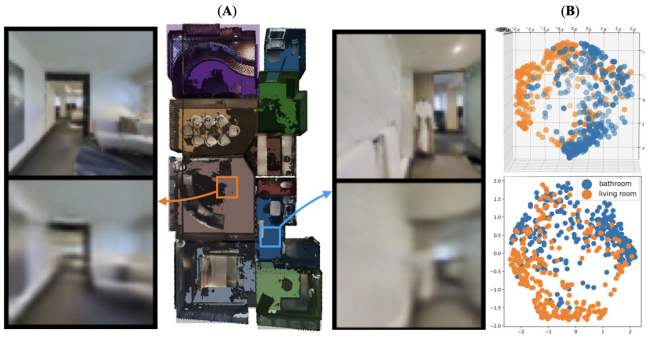

Fig. 4. **Results on Habitat**: Dynamics abstraction and next state prediction with $s^{(2)}$ as prior. The scaled APC model (Figure 3) was trained on randomly generated trajectories from an ego-centric home environment. **(A)** Rendering of a home used in our experiments. The render was taken from Matterport (Chang et al. [7]). The next step predicted model outputs (bottom) and ground truth reconstruction targets (top) are shown for two different rooms. **(B)** 2D and 3D PCA of $s^{(2)}$ vectors for episodes starting at 2 different rooms. With no training signal apart from dynamics prediction errors, $s^{(2)}$ shows separable clusters different rooms.

D. Higher level transition model and action abstractions

As discussed above, $s^{(2)}$ abstracts the transition dynamics at the lower-level (via propagation of prediction errors or an encoder). Significant efficiencies can be achieved by learning a transition function between abstract states, allowing higher-level planning and navigating to any goal in a compositional environment (Figure 2(A)). To learn to a transition function between abstract states, we introduce the idea of an **abstract action** $\mathbf{a}^{(2)}_T$ (similar to an option in hierarchical RL), which can be regarded as a latent action vector that generates/modulates the lower-level policy network (Supplementary Figure 7). Given abstract actions, we can define a transition dynamics for higher-level abstract states as $P(\mathbf{s}^{(2)}_{T+1} | \ \mathbf{s}^{(2)}_T, \mathbf{a}^{(2)}_T)$, or $f^{(2)}_s$. Here T represents timesteps at the higher-level in the hierarchy. Details regarding action abstractions in APC can be found in the Supplementary Section VI.

III. Experiments and Results

A. Abstract Transition Space

Supplementary Figure 9(A) shows the inference process for different gridworld rooms after training the APC dynamics model. Random trajectories of length $\tau = 15$ are drawn from different rooms and used for dynamics prediction task. Accurate estimates can be made in time, as the agent gathers more evidence. Supplementary Figure 9(B) shows one-shot inference for three rooms when the model is trained with shorter episodes of length $\tau = 5$. In the same figure, we also plot the PCA of final abstract states from 100 episodes for each environment. This shows that the one-shot inference result is indeed accurate. This fast inference method is useful for rapid planning with limited data. Details are in the Supplementary Section VII.

B. Zero-Shot Transfer to new Environments

A significant benefit of abstracting transition dynamics into a continuous latent space is in transfer. To illustrate this, we trained the APC dynamics model with two simple environments - a vertical and a horizontal hallway. Figure 5 shows the PCA of the higher-level abstract state space with blue and orange clusters representing the final inferred abstract states for the environments. We sampled points along the line joining the cluster centers and used the points as priors to generate a transition function at the lower level. Next state predictions were made using the generated function and a random policy. These predictions were used to reconstruct the dynamics and hence the environment captured by the transition function. These new environments ("rooms") are plotted as 5×5 grids in Figure 5. These new rooms, which were never seen by the model, demonstrates how the model can compose and transfer learned dynamics to new environments.

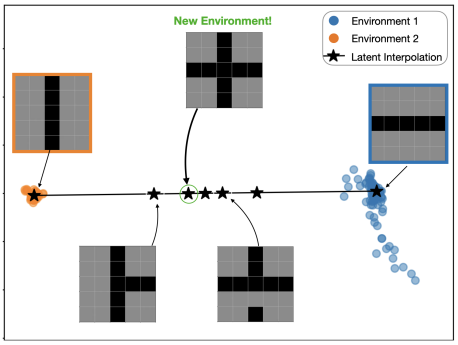

Fig. 5. **Zero shot transfer of dynamics** to new environments by interpolating the abstract states $s^{(2)}$. The dynamics for newly sampled abstract states (priors) are inferred from a model trained only on Environment 1 and 2 (Note that these rooms have different dynamics in a top-down setting). The inferred dynamics with interpolated $s^{(2)}$ priors are drawn out as new environments. This hints that the priors could be learning a smooth space spanning continuously changing transition functions.

C. Hierarchical RL and Planning

It is well-known that the learnt abstract states, along with well-defined abstract actions, can reduce the effective search space of an agent for reinforcement learning and planning, greatly reducing the complexity of the problem (Nachum et al. [33]). To demonstrate that this advantage accrues to APC, we performed simple experiments on a compositional gridworld environment (Supplementary Figure 11). A simple instantiation of APC's hierarchical transition model and hierarchical policy was learnt for this environment. Here, the abstract actions were assumed to be one of 8 possible subgoals (For details, refer to Supplementary Section VIII and Rao et al. [36]). We considered 2 tasks (1) Goal-reaching RL task where the goals can change at any point in time, and (2) Planning to reach a fixed goal from increasing distances. The baselines for these experiments are respectively: (1) A policy gradient model-free agent and (2) An MPC planner with full access to the transistion dynamics. Our results (Figure 6) show that APC is indeed robust to goal changes and can plan faster, as long as the abstract actions are well defined. Work on learning useful skills without hand-designed abstract actions (Eysenbach et al. [10]) is ongoing.

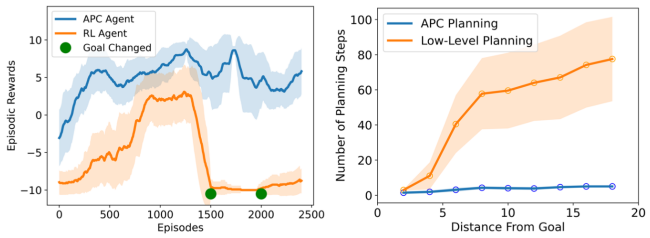

Fig. 6. Planning with APC on the grid world environment. (**Left**) Rewards collected over episodes as the goals are changed. RL agents are not robust to changing goals. (**Right**) Action steps taken to plan. With abstract actions, APC can plan exponentially faster due to the reduced sequence length. Figure taken from [36].

D. State Abstraction in Habitat 2.0

We now show that APC can be scaled to learn abstract states on realistic environments without using any labeled data. We use a modified APC model as seen in Figure 3). Dealing with high-dimensional image inputs require powerful encoders and decoders. For this experiment, we use a Residual Autoencoder, resnet18 (He et al. [21]) to encode and decode 64×64 RGB and depth images from Habitat 2.0 [37, 43] (replacing autoencoders with transformer-based ViT (Vaswani et al. [44], Dosovitskiy et al. [9]) or other architectures is straightforward). The sequential inputs $x_T^{(2)}$ to the higher level encoder are resized images. We task the scaled APC to predict single-step future states, given the current inputs and actions, and train for 500 epochs. The results are shown in Figure 4. The model directly infers $s^{(2)}$ from inputs, with clusters for 2 different rooms, showing that state abstractions can be learnt in complex image-based POMDP environments.

IV. Conclusion

This paper presents a new approach to learning transition dynamics for complex real-world environments based on a structured heirarchical model called active predictive coding (APC). APC is inspired by the architecture of the mammalian cortex and learns a hierarchy of transition functions using self-supervised learning based on prediction errors and hypernetworks. We applied APC to both traditional gridworlds and the more complex Habitat domain and showed that higher-level latent codes that generate transition dynamics for different environments form different clusters in the latent space. Furthermore, this continuous latent space exhibits a smooth transformation of transition functions, allowing APC to generate dynamics for new environments in a compositional manner. We introduce abstract actions (i.e., options) to allow transition functions to be learned for higher-level latent state spaces, giving rise to hierarchical world models. Our results demonstrate the efficacy of higher-level planning using the APC model by exploiting learned hierarchical world models and local reference frames. Our ongoing and future work is focused

on scaling the APC approach to larger-scale environments and RL benchmarks, and leveraging APC's compositional structure and ability to generate new transition functions on the fly to achieve fast transfer across environments.

References

[1] David Abel. A theory of abstraction in reinforcement learning, 2022.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1726–1734. AAAI Press, 2017.

[3] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012.

[4] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13:2003, 2003.

[5] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[6] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[8] Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: A path towards autonomous machine intelligence, 2023.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[10] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018.

[11] Katie A Ferguson and Jessica A Cardin. Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, 21(2):80–92, 2020.

[12] Karl Friston and Stefan Kiebel. Predictive coding under the free-energy principle. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364 (1521):1211–1221, 2009. doi: 10.1098/rstb.2008.0300. URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2008.0300.

[13] Tomer Galanti and Lior Wolf. On the modularity of hypernetworks, 2020.

[14] J Swaroop Guntupalli, Rajkumar Vasudeva Raju, Shrinu Kushagra, Carter Wendelken, Danny Sawyer, Ishan Deshpande, Guangyao Zhou, Miguel Lázaro-Gredilla, and Dileep George. Graph schemas as abstractions for transfer learning, inference, and planning. *arXiv preprint arXiv:2302.07350*, 2023.

[15] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[16] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks, 2016.

[17] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019.

[18] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[19] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35: 26091–26104, 2022.

[20] Jeff Hawkins. *A thousand brains: A new theory of intelligence*. Basic Books, 2021.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[22] Linxing Preston Jiang and Rajesh P. N. Rao. Dynamic predictive coding: A new model of hierarchical sequence learning and prediction in the cortex. *bioRxiv*, 2022. doi: 10.1101/2022.06.23.497415. URL https://www.biorxiv.org/content/early/2022/06/24/2022.06.23.497415.

[23] Linxing Preston Jiang and Rajesh P.N. Rao. Predictive coding theories of cortical function, November 2022. URL http://dx.doi.org/10.1093/acrefore/9780190264086.013.328.

[24] Georg B. Keller and Thomas D. Mrsic-Flogel. Predictive processing: A canonical cortical computation. *Neuron*, 100:424–435, 2018. URL https://api.semanticscholar.org/CorpusID:53105231.

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[26] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[27] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016. URL http://arxiv.org/abs/1604.00289.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[29] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight, 2017. URL https://arxiv.org/abs/1712.00948.

[30] Vincent Micheli, Eloi Alonso, and François Fleuret.

Transformers are sample-efficient world models, 2023.

[31] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention, 2014.

[32] Edvard I Moser, May-Britt Moser, and Bruce L McNaughton. Spatial representation in the hippocampal formation: a history. *Nature neuroscience*, 20(11):1448–1464, 2017.

[33] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning?, 2019.

[34] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

[35] Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2:79–87, 1999. URL https://api.semanticscholar.org/CorpusID:221608503.

[36] Rajesh PN Rao, Dimitrios C Gklezakos, and Vishwas Sathish. Active predictive coding: A unifying neural model for active perception, compositional learning, and hierarchical planning. *Neural Computation*, 36(1):1–32, 2023.

[37] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. *CoRR*, abs/1904.01201, 2019. URL http://arxiv.org/abs/1904.01201.

[38] J. Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 253–258 vol.2, 1990. doi: 10.1109/IJCNN.1990.137723.

[39] J. Schmidhuber. Learning to generate subgoals for action sequences. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 453 vol.2–, 1991. doi: 10.1109/IJCNN.1991.155375.

[40] Juergen Schmidhuber. A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. The MIT Press, 02 1991. ISBN 9780262256674. doi: 10.7551/mitpress/3115.003.0030. URL https://doi.org/10.7551/mitpress/3115.003.0030.

[41] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014. URL http://arxiv.org/abs/1404.7828.

[42] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[43] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John M. Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel X. Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. *CoRR*, abs/2106.14405, 2021. URL https://arxiv.org/abs/2106.14405.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[45] James CR Whittington, Joseph Warren, and Timothy EJ Behrens. Relating transformers to models and neural representations of the hippocampal formation. *arXiv preprint arXiv:2112.04035*, 2021.

Supplementary Materials

V. Related Work

In this section we briefly present the Active Predictive Coding (APC) architecture and its motivation in light of related works. For a more detailed discussion on the topic, please refer to (Rao et al. [36]). Figure 7 represents a general version of APC. We discuss the following relevant components in the model (1) Top-down Modulation, (2) Hierarchical World Models (3) Hierarchical Policy (4) Reference Frames.

Top-Down Modulation refers to an abstract state, conditioning a function that is usually working at a constrained spatio-temporal scale. Intuitively, the latent vectors "abstract" critical parts of the function representation to re-use learnt information in novel scenarios. For example, a child learns to lift a coffee mug and has no problem transferring that experience to picking up a water bottle. There is evidence from neuroscience that cortical neurons implement top-down gain modulation that allow such transfer of learnt behaviors (Ferguson and Cardin [11]). We propose variants of hypernetworks (Ha et al. [16], Galanti and Wolf [13]) as potential candidates to implement such abstractions in APC.

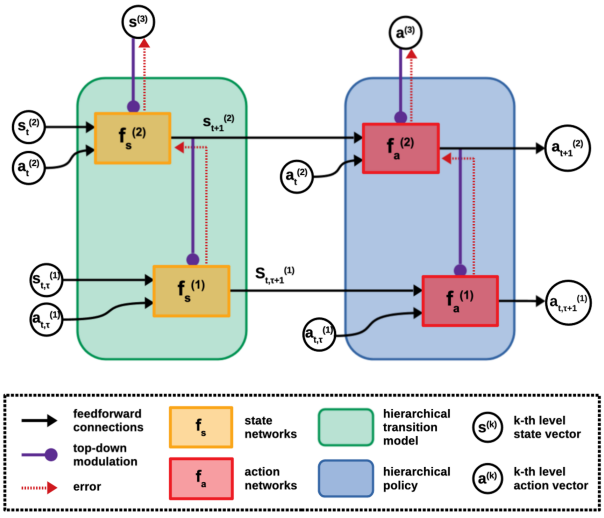

Fig. 7. Components of a 3-level Active Predictive Coding model. Top-down connections modulate the communication between different levels of hierarchy. Feedforward connections learn task-invariant transition models and task-specific policies.

Hierarchical Transition Models: (Ha and Schmidhuber [15]) introduces world models into model-based RL. Since then, powerful variants of world models have been successful at modeling more and more complex environment dynamics (Hafner et al. [18, 19], Micheli et al. [30]). Yet, the world models are limited in scope when exposed to novel environments. Graph schemas have gained popularity in recent years as potential computational principles governing complex functions like abstraction, transfer and planning in the brain (Guntupalli et al. [14], Moser et al. [32], Whittington et al. [45]). APC proposes hierarchical world models that learn abstractions of primitive transitions limited in space and time, and further learns to transition in the new abstract space with access to only unsupervised prediction errors. We believe hierarchical world model is a novel idea and present a concrete implementation with APC, in Section II.

Hierarchical Policies: Hierarchical Reinforcement Leaning and action abstractions have been popular since the first formulation of Options (Sutton et al. [42], Barto and Mahadevan [4], Schmidhuber [38, 39, 40]). With the introduction of deep neural networks (LeCun et al. [28], Schmidhuber [41]), many variants of hierarchical and deep reinforcement learning were developed (Bacon et al. [2], Hafner et al. [19], Kulkarni et al. [26]). There is also an extensive discussion on abstract states and actions in (Abel [1]). In similar lines, APC develops the notion of action abstractions as a sub-goal modulated policy. These are the blue shaded parts of APC in Figure 7. We dedicate Section 3.4 to action abstractions and do not discuss further about this formulation, since it is already dealt in detail, in (Rao et al. [36]).

Reference Frames: Working with smaller scales of space and time considerably reduces the complexity of the problem at hand. Biological agents naturally work with a local dynamics, constantly context switching between tasks and dynamics. Humans are exceptional at ignoring non-contextual noise and focusing their attention on the relevant task. There is considerable evidence from neuroscience that grid-cells in the cortex are involved in the implementation of reference frames that serve this function (Hawkins [20]). Evidence from (O'Keefe and Dostrovsky [34], Moser et al. [32]) also suggests that hippocampal and cortical circuits in rats and humans trigger spatial reference frames used for solving problems such as navigation and abstract reasoning. Past work has made attempts to simulate contexts with hard attention models (Mnih et al. [31]) We provide a plausible concrete implementation of reference frames in Section 2.

VI. Dynamics Abstraction Models

A. Top-Down Modulation

For a simple version of hierarchical abstraction of dynamics using APC, we consider two possible approaches. Both approaches use an approximation of hypernetworks, which are neural networks that generate the parameters of other neural networks (Ha et al. [16]). In our first approach, a hypernet predicts a weight vector $\mathbf{w}$ for combining, using a weighted sum, a set of learned basis matrices M to generate the state transition function f_s at the lower-level (Jiang and Rao [22]). Figure 8 shows the parameterization of the APC model.

$$\mathbf{w} = \mathcal{H}(\mathbf{s}_T^{(2)}) \tag{8}$$

$$f_s = \sum_{k=1}^{K} w_k \mathbf{M_k} \tag{9}$$

$$\hat{s}_{t+1} = ReLU(f_s(s_t, a_t)) \tag{10}$$

We also experimented with an embedding approach for top-down modulation where the hypernetwork predicts a vector

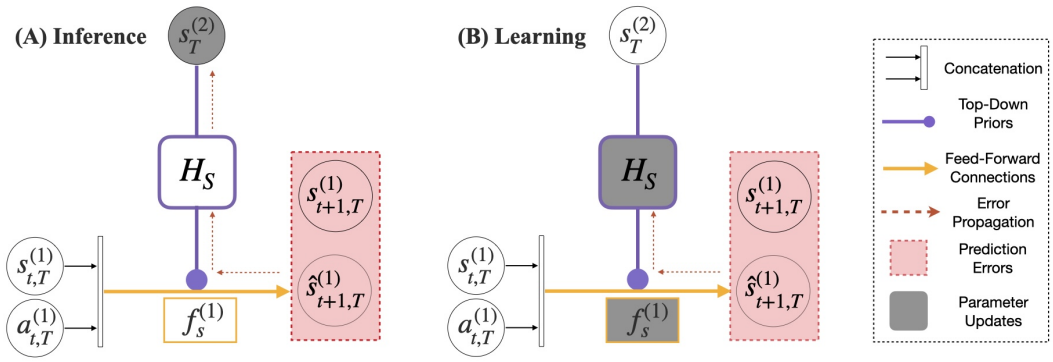

Fig. 8. Top-down modulation with Hypernetworks. **(A)** During inference, gradient updates to all the model parameters are switched off, except the higher-level latent code. Next state prediction errors accumulate and modify the latent via the backpropagation algorithm. **(B)** After running K inference steps, the latents are frozen and the model parameters are updated. The inputs to the transition function are the current state and action of the agent. This model is motivated by (Jiang and Rao [22])

.

embedding from the higher-level latent state. The set of matrices M is replaced by an RNN that takes as input the top-down embedding, the current state and action as inputs and predicts the next state. In practice, we found that adding additional decoders after the RNN prediction gave results comparable to the mixture of matrices method discussed above.

$$\mathbf{e} = \mathcal{H}_\theta(\mathbf{s}_T^{(2)}) \tag{11}$$

$$\mathbf{h_t} = \tanh(\mathbf{x_t}W_1 + b_1 + \mathbf{h_{t-1}}W_2 + b_2) \tag{12}$$

$$\hat{s}_{t+1} = ReLU(\mathbf{h_t}W_3 + b_3) \tag{13}$$

Where $\mathbf{x_t} = [\mathbf{e}, s_t, a_t]$ is the concatenated input at the lower-level and $[\theta, W_{1:3}, b_{1:3}]$ are the model parameters. Unlike traditional autoencoders (Kingma and Welling [25], Baldi [3]), and the scaled APC, this version does not have an explicit encoder mapping observations to a latent space. The latent codes, i.e., higher-level states, are directly inferred via backpropagation of prediction errors during inference (rather than being used solely for learning as in traditional neural networks). This is an insight from the purely generative models used in the brain Rao and Ballard [35], Jiang and Rao [23].

B. Action Abstractions

The action abstractions in APC represent subgoals or subtasks similar to the formulation in (Hafner et al. [19], Schmidhuber [39], Abel [1]). These abstract actions are tied to a context dynamics, since a particular action might not be relevant in all scenarios. For example, "Open the microwave" is a valid subgoal if the agent context is kitchen and not when the context is, say, a conference room. The latent codes for action abstractions can be learnt by a similar inference process discussed for state abstractions. For this paper however, we assume a fixed set of subgoals for each of our local reference frame dynamics. We leave the subgoal learning with APC as a future work and instead learn a policy, conditioned on a set of fixed subgoals represented as one hot vectors. Figure 11 taken from our previous work (Rao et al. [36]) shows these learnt policies for subgoals of two 3x3 maze environments.

VII. Additional Results: Few-Shot Inference of Abstract States

In this section, we provide experiment details for The results for the analysis of abstract state inference is shown in Figure 9. Our experiments with the learnt latent codes $\mathbf{s}^{(2)}$ was focused on studying the nature of the abstract transition space. This turned out to be useful when transfering learnt dynamics to novel environments. For our experiment setup, we collect episodic data for 5 room environments with different dynamics. The environment dynamics can be changed by placing the walls in different patterns. The hypernetwork used is a 4 layer deep neural net with 256 units at each layer. We use ReLU non-linear activation everywhere unless specified. The learning rate for inference was kept much higher at $\eta = 0.1$, whereas the learning rate for training both the hypernetwork and the transition function were $\eta_{\mathcal{H}} = \eta_f = 0.001$. For each environment, we collect episodes of length τ and feed it to our APC model. We experiment inference with episodes of lengths 2, 5, 15, 25 and 50. Typically, longer episodes perform better since the data available about the environment increases. For every episode, APC model first infers the latent code, freezes the final latent code and performs gradient updates to the model parameters using the prediction errors. Adam optimizer was used for both inference and training.

To choose a dimension for $s_{(2)} \in \mathcal{R}^d$, we run inference and training for $d = [4, 8, 16, 32, 64]$. Our intent was to create a balance between information capacity (neatly clustered latent codes) and prediction errors. (Dawid and LeCun [8]) notes that generative latent variable architectures can collapse if the latent codes have very high information capacity. In such cases, the transition function completely ignore the inputs s_t, a_t and learn to essentially push all the necessary information into the latent code. In our experiments, we found $d = 32$ to optimally minimize prediction errors while maintaining separable latent code clusters. The plots representing latent codes in this paper are 2-D PCA of $s^{(2)}$ originally in a 32 dimension space unless specified.

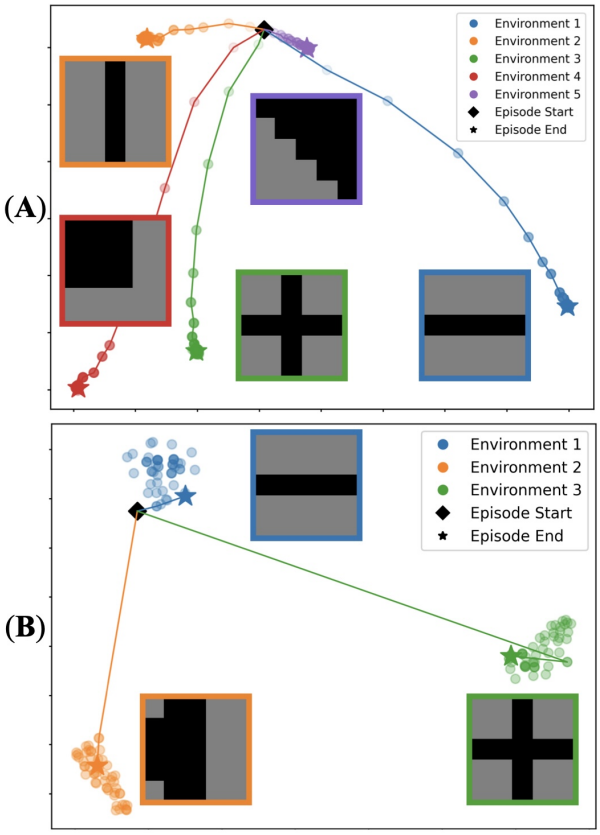

Fig. 9. Inferring the 5 x 5 room with partially observable patches. Both the figures are 2-D PCA of $d = 32$ dimension latent codes. **(A)** $\tau = 10$ step inference for different environments trained with episodes of length 10. **(B) One-Shot inference** for three different rooms. Final latent codes for 100 random episodes from each environment are also plotted in the background to validate the one-shot inference. Details in text.

VIII. Additional Results: Hierarchical Planning and Fast Transfer to New Compositions

We present the results for hierarchical planning from [36] in Figure 6 and discuss the experiment details here. We choose two (3 x 3) rooms as seen in the reference Figure 11 and construct larger environments with them. Abstract actions are learnt as policies with sub-goals as discussed in Section VI. Abstract state transitions are learnt by exploring in the environment and recording "higher-level" steps $(s_T^{(2)}, a_T^{(2)}, s_{T+1}^{(2)})$ and learning a transition function in this abstract space. Our aim with planning is to show the combinatorial advantage APC has over traditional planning and reinforcement learning that do not use reference frames. The planning problem with APC reduces to sequencing abstract actions $a_T^{(2)}$ instead of primitive actions a_t. With known dynamics model at both higher and lower levels, APC plans effectively even for long horizon goals (Figure 6(D)). Equipped with hierarchical world models, APC agents are also quick at learning to navigate in environments with changing goals, when compared against traditional RL agents using actor-critic methods (Figure 6(C)).

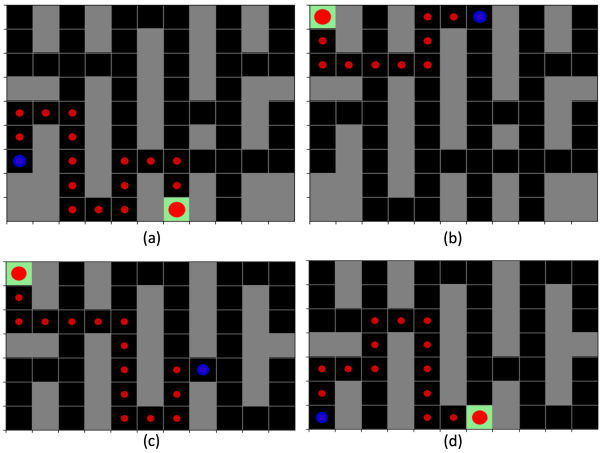

Fig. 10. Additional results for zero-shot transfer to new goals. Using hierarchical world models and planning, APC agent is able to remain robust to changing goals. Note that when a large compositional environment is changed, only transitions at the higher level must be re-learnt. Local reference frames allow transfer of dynamics at the lower level, as seen in previous figures. Figure taken from ([36]).

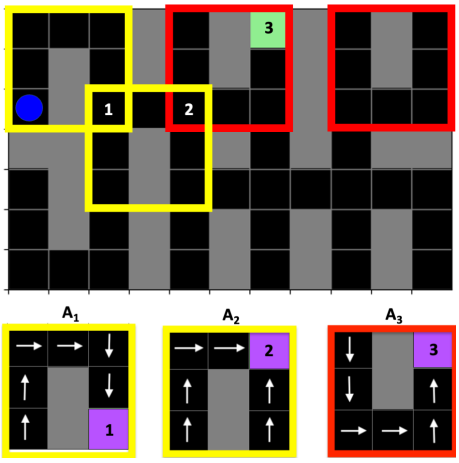

Fig. 11. Just 2 different abstract states $s^{(2)}$, (marked with red and yellow outlines) compose to create a complex looking environment. Here, abstract actions are subgoals that connect 2 different instantiations of abstract states. Actions $\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}$ represent policies with goals 1, 2 and 3 in two different room environment. Given a composition of the rooms, abstract actions could be sequenced to easily reach any goal. Figure from (([36])).