# Auto-Encoding Recurrent Representations

**Chris Nota**
Autonomous Learning Lab
University of Massachusetts
Amherst, MA 01002
cnota@cs.umass.edu

**Clement Wong**
iRobot Corporation
Bedford, MA 01730
cwong@irobot.com

**Philip S. Thomas**
Autonomous Learning Lab
University of Massachusetts
Amherst, MA 01002
pthomas@cs.umass.edu

## Abstract

We introduce a new approach for learning a task-independent Markovian representation for reinforcement learning using a specialized recurrent conditional variational auto-encoder. Unlike most existing approaches, the representation can be learned without using backpropagation through time. This gives the approach several computational properties that are desirable in the reinforcement learning setting. The resulting representation is highly flexible. In addition to its use as an input to an actor-critic network, we present preliminary results showing that the representation can be used to reconstruct short-term visual memories, predict environment dynamics, and associate new observations with previous experiences. We apply the method to a simulated and real-world home robotics task.

**Keywords:** reinforcement learning, memory, representation learning

**Acknowledgements**

We would like to thank iRobot Corporation for generously supporting this work and providing the robots. We would especially like to thank Kshitij Bichave, Deepak Sharma, Billy Zhou, and Letian Chen for their feedback and technical support on the project, Danielle Dean for supporting and encouraging research initiatives including this one, and Alberto Soragna for his patient technical support in interfacing with the robots.

# 1 Introduction

Most *reinforcement learning* (RL) algorithms depend on the *Markov property* [1, 2, 3]. When this assumption is satisfied by the environment, it is possible for an agent to behave optimally based on its immediate observations of the environment. For such problems, the agent does not need any kind of short-term memory. However, the Markov assumption is violated by many real-world problems, and such problems require some form of short-term memory to solve.

For example, consider a robot vacuum such as the iRobot Roomba j7+. In order to optimally navigate a home, it must remember attributes of the environment as it encounters them, such as the layout of the home, the location of the charging dock, and the position of various obstacles. In this paper, we develop a method for learning a representation of the environment for which the Markov assumption holds by modeling a form of short-term memory using a specialized *conditional variational auto-encoder* (CVAE) [4]. The encoder network compresses the agent's history into a fixed-length vector representation which can be decoded to reconstruct the original history, as shown in Figure 1. This vector is passed to the RL policy, which allows the robot to act based on all of the information contained in the representation. Unlike most existing methods, this approach does *not* rely on backpropagation through time (BPTT) [5].

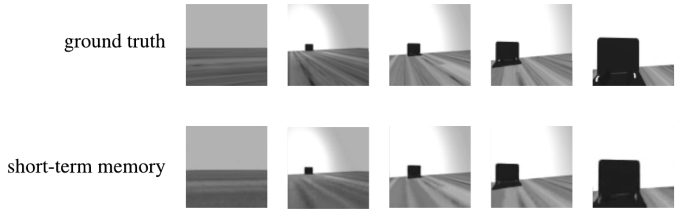


Figure 1: Reconstruction of the short-term memory of a simulated robot vacuum with a front-facing camera docking using the proposed method. The history of observations is recursively compressed into a vector which is given as input to a reinforcement learning policy. This vector can be recursively decoded to reconstruct the original camera images.

The method we propose produces a highly flexible representation that can be used for many purposes, such as:

1. The representation can be used as an input to a reinforcement learning algorithm.
2. The representation can be unrolled backwards in time in order to reproduce recent experiences by the agent.
3. The representation can be rolled *forward* in time to serve as a predictive model of the environment.
4. Given an observation, the model can predict representations which were likely to correspond to that observation. These representations can then, in turn, be unrolled backwards or forwards in time in order to deduce the likely context of an observation, thus serving as a form of long-term *associative* memory.

While in this paper we focus on conventional reinforcement learning methods, we consider the flexibility of the representation to be particularly motivating as it opens new possibilities involving the design of intelligent agents. The lack of dependence of BPTT is another feature that we find compelling because it removes some of the computational difficulties that arrise when combining BPTT with RL. Instead of taking as input long sequences of experiences, as with BPTT, our approach applies updates over individual transitions, which fits better within the paradigm of *temporal difference* (TD) learning [3, 6]. In our approach, information propagates "forward" in time in a similar manner to the way TD-errors propagate backwards in time. The key idea is to auto-encode the agent's internal recurrent state alongside its external observations. In reference to this, we call our approach *auto-encoding recurrent representations* (AERR).

# 2 Notation

A *partially observable Markov decision process* (POMDP) [2] is the tuple, $\mathcal{S}, \mathcal{A}, P, R, d_0, \Omega, O)$, where $\mathcal{S}$ is the state set, $\mathcal{A}$ is the action set, $P : \mathcal{S} \times A \times S \to [0, 1]$ is the transition function, $R : \mathcal{S} \times A \to \mathbb{R}$ is the reward function, $d_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution, $\Omega$ is the observation set, and $O : \mathcal{S} \times \Omega \to [0, 1]$ is the observation function. We consider the episodic setting, beginning at time $t = 0$. The state, action, reward, and observation at time $t$ are represented by the random variables $S_t, A_t, R_t$, and $O_t$ respectively. The initial state, $S_0$, is sampled from $d_0$. At each timestep, $t$, the agent makes an observation, $O_t \sim O(S_t, \cdot)$, and chooses an action, $A_t$, in a way that is described later. Finally, the next state is sampled from the transition function, $S_{t+1} \sim P(S_t, A_t, \cdot)$, and the agent is given a reward, $R_t = R(S_t, A_t)$. The episode ends when the agent enters a special state called the *terminal absorbing state*, $s_\infty$.

We consider an agent which maintains some internal state, $Z_t \in \mathcal{Z}$, where $\mathcal{Z}$ is the set of internal states and $Z_t$ is the internal state at time $t$. The internal state is generated by a (typically parameterized) function called the *encoder*, $f : \mathcal{Z} \times \Omega \times A \times \mathcal{Z} \to [0, 1]$, such that $Z_t \sim f(Z_{t-1}, O_t, A_t, \cdot)$. $Z_{-1}$ is defined to be some arbitrary constant. Finally, an action is sampled according to a parameterized *policy*, $\pi : \mathcal{Z} \times \mathcal{A} \to [0, 1]$, such that $A_t \sim \pi(Z_t, \cdot)$. Typically, the goal of the agent is to learn parameters for both the encoder and the policy that maximize the expected sum of rewards, $\mathbb{E}[\sum_{t=0}^{T} R_t]$,

where $T$ is the time horizon defining the maximum length of an episode. The goal of this paper is to introduce a novel approach to learning an encoder that allows the agent to solve POMDPs by producing a sequence of internal states (or representation) that is both Markovian and that are positioned in $\mathcal{Z}$ in such as way that is conducive to training $\pi$.

## 3 Markovian Representations Using Short-Term Memory

In this section, we introduce the basic intuition to our approach. Consider some arbitrary non-Markovian sequence of random variables, $(X_0, X_1, \ldots, X_T)$, where all $X_t$ are in some space $\mathcal{X}$. Our goal is to learn some encoder, $f$, as described in the previous section, such that the joint sequence $(X_0, Z_0, \ldots, X_T, Z_T)$ is Markovian with respect to $Z_t$, i.e.:

$$\Pr(X_{t+1}, Z_{t+1}|Z_t) = \Pr(X_{t+1}, Z_{t+1}|X_0, Z_0, \ldots, X_t, Z_t). \tag{1}$$

Here we consider an arbitrary space $\mathcal{X}$ such that for all $t$, $X_t \in \mathcal{X}$. In the RL case this is the tuple $(O_t, A_t)$. The basic idea is to find an $f$ that ensures we can always reconstruct $(X_t, Z_{t-1})$ based on $Z_t$ alone. Specifically, we wish to ensure the existence of a *decoder*, $g : \mathcal{Z} \to \mathcal{X} \times \mathcal{Z}$, such that $(X_t, Z_{t-1}) = Z_t$. The existence of such a $g$ is sufficient to prove that the joint sequence is Markovian. That is:

**Lemma 1.** *If there exists some function $g$ such that $g(Z_t) = (X_t, Z_{t-1})$, then the sequence $(X_0, Z_0, \ldots)$ is Markovian.*

*Proof.* This can be shown by induction:

$$\begin{aligned}
\Pr(X_{t+1}, Z_{t+1}|Z_t) &= \Pr(X_{t+1}, Z_{t+1}|g(Z_t), Z_t) \\
&= \Pr(X_{t+1}, Z_{t+1}|Z_{t-1}, X_t, Z_t) \\
&= \Pr(X_{t+1}, Z_{t+1}|g(Z_{t-1}), Z_{t-1}, X_t, Z_t) \\
&\vdots \\
&= \Pr(X_{t+1}, Z_{t+1}|X_0, Z_0, \ldots, X_t, Z_t).
\end{aligned}$$

□

Our basic approach is to simultaneously learn $f$ and $g$ such that $g = f^{-1}$. This scheme is recognizable as variant of an auto-encoder [7]. The resulting representation is recurrent, in that we repeatedly call the function $f$ in order to produce each successive $Z_t$.

We argue that unlike existing schemes, auto-encoding individual samples of $(X_t, Z_{t-1})$ is sufficient for learning a representation that satisfies Lemma 1 and there is no need to reply on BPTT. Instead, the scheme can be thought of as a type of bootstrapping method like TD learning—specifically, it is a bootstrapping method because the decoder, $g$, learns to predict a representation of $Z_{t-1}$ that is initially an arbitrary estimate which does not itself satisfy Lemma 1. Rather than bootstrapping value backwards through time (as in TD), our method bootstraps information forwards through time. The learned representation, $Z_t$, can be thought of as a type of short term memory because it can be used to reconstruct the entire sequence $(X_0, \ldots, X_t)$. Thus, $Z_t$ can be thought of as the agent's "memory" at time $t$ and $g$ is a function that lets us inspect that memory.

## 4 Short-Term Memory Using a Conditional Variational Auto-Encoder

We consider representing both the encoder $f$ and decoder $g$ using stochastic functions parameterized by vectors $\theta$ and $\phi$ respectively. Recall Lemma 1, which gives us the requirement that $g(Z_t) = (X_t, Z_{t-1})$. Our approach is to satisfy that requirement approximately by finding

$$\max_{\theta,\phi} \mathbb{E}_D \Big[ P\big(g_\phi(f_\theta(X_t, Z_{t-1})) = (X_t, Z_{t-1})\big) \Big] \tag{2}$$

over some dataset $D$. However, computing the entire expected probability is intractable for most common parameterizations of $f$ and $g$ because of the need to integrate over all possible outputs of $f$. One approach for sidestepping this issue is to apply the *variational auto-encoder* (VAE) framework [8]. In this approach, we introduced an assumed "prior" distribution over the target latent variable, $Z_t$, and try to find

$$\max_{\theta,\phi} \mathbb{E}_D \Big[ \ln p(X_t, Z_{t-1}|Z_t) - D_{\mathrm{KL}}(q(Z_t|X_t, Z_{t-1})\|p(Z_t)) \Big], \tag{3}$$

where $p(X_t, Z_t - 1|Z_t) = P(g_\phi(Z_t) = (X_t, Z_{t-1})|Z_t)$, $q(Z_t) = P(f_\theta(X_t, Z_{t-1}) = Z_t|X_t, Z_{t-1})$ and $p(Z_t)$ is the prior. While we use the reparameterization trick [8] to backpropagate through $Z_t$, as previously mentioned, we do not backpropagate through $Z_{t-1}$.

Instead, we can consider two methods for propagating information forwards through time. In the first case, consider an agent performing completely "online" updates, that is, updates in which we only consider the most recent transition (as in traditional RL methods). Each time we perform an update, a little bit of information contained in $Z_{t-1}$ will be propagated forward into our new $Z_t$. If we were to recompute $Z_t$ using the update $f_\theta$, then during the next update, this information would in turn be propagated into $Z_{t+1}$, and so on. In the second case, consider a fixed replay buffer. Supposing that it was created using complete trajectories, each individual $Z_t$ will be referenced twice: in the transition $(X_t, Z_{t-1}, X_{t+1}, Z_t)$, and in the transition $(X_{t+1}, Z_t, X_{t+2}, Z_{t+1})$. We can ensure that the information is propagated forward by updating the value of $Z_t$ in the latter transition. Thus, in either case there is no need to rely upon BPTT. While these arguments are intuitive and supported by preliminary experiments, more rigorous demonstrations are left for future work.

We next consider the question: How can we make the learned representation more useful for downstream tasks? The basic idea is to add additional priors, turning our VAE into a conditional VAE [4]. What priors could be useful? We consider three priors in particular. The resulting systems are shown schematically in Figure 2.

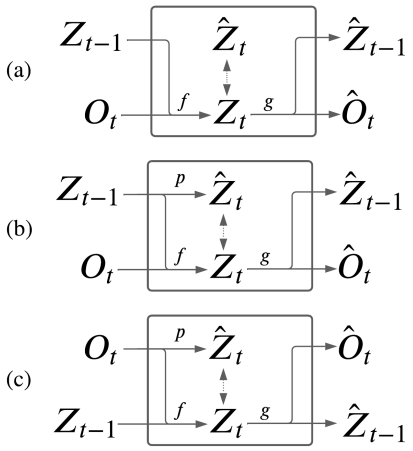


Figure 2: Three types of AERR units. The dotted line indicates the use of the KL divergence. (a) shows the system with an unconditioned prior. (b) shows the system with the forward dynamics prior. (c) shows the system with the associated memory prior.

**Unconditioned Prior:** The first prior we consider is simply the unconditional $p(Z_t)$, as used in the original VAE [8]. Most commonly this is taken to be a fixed standard Gaussian. This prior works well because it is computationally convenient and ensures that the latent values stay within a reasonable magnitude.

**Forward Dynamics:** The next prior we consider is $p(Z_t|Z_{t-1})$. Incorporating this as a Gaussian prior ensures that all $Z_t$ which follow from the same $Z_{t-1}$ are located in a similar place in latent space. This is desirable for two reasons: 1) This may make it easier for our decoder to predict $Z_{t-1}$, and 2) It is likely that for many downstream tasks, much of the information necessary to make a decision will be contained in $Z_{t-1}$. Therefore, locating all successors to $Z_{t-1}$ in latent space may make it easier for these downstream tasks. This prior is also recognizable as a stochastic model of the latent *forward dynamics* of the system. That is, by sampling from $p(Z_t|Z_{t-1})$ repeatedly, we can generate samples of the latent dynamics of the system. By then using our decoder, $p(X_t|Z_t)$, we can generate the observations associated with the latent sequence. This model could be potentially useful, for example, for model-based RL.

**Associative Memory:** The third prior we consider is what we call the *associative memory prior*, $p(Z_t|O_t)$. This prior learns an association between an observation $O_t$ and a distribution over latent states $Z_t$. Again, if we use a Gaussian prior, the model will attempt to group all $Z_t$ associated with a particular $O_t$ close to each other in latent space. For example of how this could be useful, consider a reinforcement learning task that is Markov with respect to $O_t$ (i.e., an MDP). The optimal action will depend only on $O_t$. By grouping all latent states associated with $O_t$ together in latent space, it is potentially easier for the policy to learn the optimal action. This model also gives us an interesting "memory-like" function. Suppose the agent makes some observation $O_t$. We can then sample from $p(Z_t|O_t)$. $Z_t$ can then be unrolled using the decoder to generate trajectories that "could have" led to $X_t$, when combined with the forward dynamics prior, $Z_t$ could be used to predict which latent states might follow from $X_t$.

**Combined Prior:** For our final system, we include all of these priors in the loss function, allowing us to retain the benefits of each of them. During our experiments, we did not notice any destructive interference.

## 5 Experiments

We performed some preliminary experiments using the AERR framework on a real and simulated docking task involving a modified iRobot Roomba J7+ robot. Images of the robot completing the task in both experimental setups can be found in Figures 3 and 4. In addition to the performance of the robot on the task, we present some qualitative results involving the quality of the learned inverse dynamics, forward dynamics, and associative memory. Overall, we found that the robot was able to learn the docking task as quickly and as well using the AERR approach as with the baseline algorithm in both the simulator and in the real world, and that AERR was able to accurately model the environment.

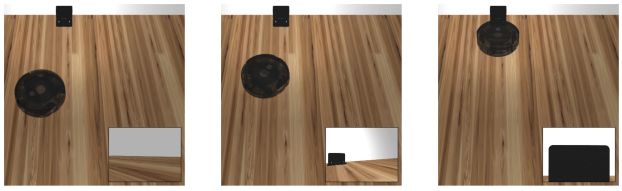

Figure 3: A sample episode from the docking simulator using a fully trained policy. The perspective of the front-facing camera is shown in the bottom-right of each image. (a) The robot is initialized in a random position. (b) The robot turns to face the dock. (c) The robot successfully docks.

Figure 4: A sample episode from the real-world docking environment using a fully trained policy, similar to the simulated episode in Figure 3. We see the same basic steps: (a) The robot is initialized in a random position. (b) The robot turns to face the dock. (c) The robot successfully docks.

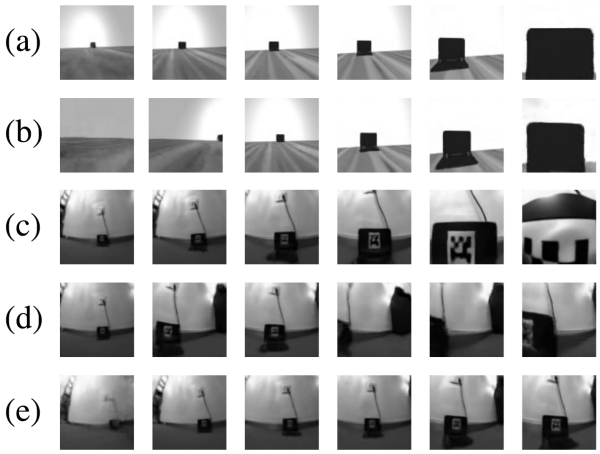

Figure 5: Sample episodes generated by the latent model. First, a latent vector is sampled from an initial distribution. Then, predicted internal states are recursively sampled from the forward model. The images for each internal state are generated using the decoder.

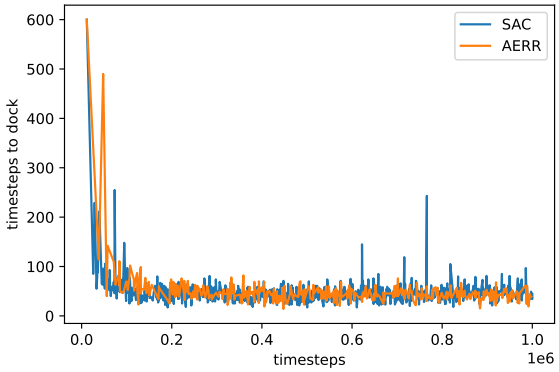


Figure 6: The performance of the AERR and baseline SAC algorithm on the simulated environment, in terms of the time taken to successfully dock. While we found these results to be stable over multiple runs, we caution that the preliminary results presented here represent a single run and that the level of uncertainty has not been quantified.

Both the baseline approach and the AERR approach used variants of the soft actor-critic (SAC) [9]. In the baseline approach, we modified the network to use a convolution network shared between the actor and critic networks. For the AERR, we instead used the output of the encoder as the input to both the actor and critic networks. The encoder accepts as input an $84 \times 84$ grayscale image, the previous action, a boolean indicating whether the input is an initial state, and several previous latent states. In order to reduce reconstruction errors resulting from repeated calls to the decoder, instead of only auto-encoding $Z_{t-1}$, we also auto-encoded $Z_{t-2}, Z_{t-4}, Z_{t-8}, \ldots, Z_{t-32}$. The image is processed by a 4 convolution layers with ReLU activations and finally a fully connected layer. In order to speed learning, the final layer outputs two vectors of length $512$, representing the mean and log standard deviation of each component of a multivariate normal distribution with a diagonal covariance matrix.

Figure 3 shows the performance of AERR and SAC on the simulated environment. Overall, the results were similar. While we found the robot was able to dock successfully after training in the real world, as in Figure 4, we have not yet quantified these results. In our opinion, the more interested results are the qualitative results shown in Figures 1 and 5. Figure 1 shows samples generated by the decoder starting with the $Z_t$ taken at the final timestep. In this example, the decoder produced a nearly pixel perfect recreation of the episode. We examined $20$ such episodes and found that the model was able to reproduce all $20$ with similar accuracy. Figure 5 shows episodes generated by the forward model, including samples from the real-world model. We were able to produce similar samples using the associative memory prior, but had to exclude them in the interest of space.

Overall, while our results are preliminary, we believe they support the overall viability of the proposed approach. In particular, we find the fact the episodes are able to reconstructed from the latent vector and the fact that the latent vector can be successfully used to train an agent to be compelling evidence of the viability of the approach.

## References


[1] A. A. Markov, "The theory of algorithms," *Trudy Matematicheskogo Instituta Imeni VA Steklova*, vol. 42, pp. 3–375, 1954.

[2] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[5] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[6] R. S. Sutton and A. G. Barto, "Time-derivative models of pavlovian reinforcement.," 1990.

[7] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, 1993.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.