# LEARNING POWERFUL POLICIES BY USING CONSISTENT DYNAMICS MODEL

**Shagun Sodhani, Anirudh Goyal, Tristan Deleu, Jian Tang**
Mila, University of Montreal
sshagunsodhani@gmail.com

**Yoshua Bengio**
CIFAR Senior Fellow and Mila, University of Montreal

**Sergey Levine**
University of California, Berkeley

## ABSTRACT

Model-based Reinforcement Learning approaches have the promise of being sample efficient. Much of the progress in learning dynamics models in RL has been made by learning models via supervised learning. There is enough evidence that humans build a model of the environment, not only by observing the environment but also by interacting with the environment. Interaction with the environment allows humans to carry out *experiments*: taking actions that help uncover true causal relationships which can be used for building better dynamics models. Analogously, we would expect such interactions to be helpful for a learning agent while learning to model the environment dynamics. In this paper, we build upon this intuition, by using an auxiliary cost function to ensure consistency between what the agent observes (by acting in the real world) and what it imagines (by acting in the "learned" world). We consider several tasks - Mujoco based control tasks and Atari games - and show that the proposed approach helps to train powerful policies and better dynamics models.

## 1 INTRODUCTION

Reinforcement Learning (RL) consists of two fundamental problems: *learning* and *planning*. *Learning* refers to improving the agent's policy by interacting with the environment while *planning* refers to improving the policy without interacting with the environment. These problems evolve into the dichotomy of *model-free* methods (which primarily rely on *learning*) and *model-based* methods (which primarily rely on *planning*). While *model-free* methods have shown many successes (1; 2; 3), their high sample complexity remains a major challenge. In contrast, model-based RL methods aim to improve the sample efficiency by learning a dynamics model of the environment. But these methods have several caveats. If the policy takes the learner to an unexplored state in the environment, the learner's model could make errors in estimating the environment dynamics, leading to sub-optimal behavior. This problem is referred to as the model-bias problem (4).

To make predictions about the future, dynamics models are unrolled step by step leading to "compounding errors" (5; 6): an error in modeling the environment at time $t$ affects the predicted observations at all subsequent steps. In the model-based approaches, the dynamics model is usually trained with supervised learning techniques and the state transition tuples (collected as the agent acts in the environment) become the supervising dataset. Hence the process of learning the model has no control over what kind of data is produced for its training. That is, from the perspective of learning the dynamics model, the agent just observes the environment and does not "interact" with it. On the other hand, there's enough evidence that humans learn the environment dynamics not just by observing the environment but also by interacting with the environment (7; 8). Interaction is useful as it allows the agent to carry out experiments in the real world which is clearly a desirable characteristic when building dynamics models.

This leads to an interesting possibility. Consider a learning agent training to optimize an expected returns signal in a given environment. At a given timestep $t$, the agent is in some state $s_t \in S$ (State space). It takes an action $a_t \in A$ (action space) according to its policy $a_t \sim \pi_t(a_t|s_t)$, receives a

reward $r_t$ (from the environment) and transitions to a new state $s_{t+1}$. The agent is trying to maximize its expected returns and has two pathways for improving its behaviour:

1. ***Close-loop* path:** The agent interacts with the environment acting in the real world at every step. The agent starts in state $s_0$ and is in state $s_t$ at time $t$. It chooses an action $a_t$ to perform (using its policy $\pi_t$), performs the chosen action, and receives a reward $r_t$. It then observes the environment to obtain the new state $s_{t+1}$, uses this state to decide which action $a_{t+1}$ to perform next and so on.
2. ***Open-loop* path:** The agent interacts with the learned dynamics model by *imagining* to act and predicts the future observations (or future belief state in case of state space models). The agent starts in state $s_0$ and is in state $s_t$ at time $t$. Note that the agent "imagines" itself to be in state $s^I_t$ and can not access the true state. It chooses an action $a_t$ to perform (using its policy $\pi_t$), acts in the "learner's" world (dynamics model) and imagines to transition to the new state $s^I_{t+1}$. The current "imagined" state is used to predict the next "imagined" state. During these "imagined" roll-outs, the agent only interacts with the learner's "world" and not with the environment.

The agent could use both the pathways simultaneously. It could, in parallel, (i) build a model of the environment (*dynamics* model) and (ii) engage in interaction with the real environment as shown in Figure 1. As such, the two pathways may not be *consistent* given the challenges in learning a multi-step dynamics model. By *consistent*, we mean the behavior of state transitions along the two paths should be indistinguishable. Had the two pathways would be *consistent* and we could say that the learner's dynamics model is grounded in reality. To that end, our contributions are the following:

1. We propose to ensure consistency by using an auxiliary loss which explicitly matches the generative behavior (from the open loop) and the observed behavior (from the closed loop) as closely as possible.
2. We evaluate our approach on 7 Mujoco based continuous control tasks and 4 Atari games and observe that the proposed approach helps to train more powerful policies.
3. We compare our proposed approach to the state-of-the-art state space models (9) and show that the proposed method outperforms the sophisticated baselines despite being very straightforward.

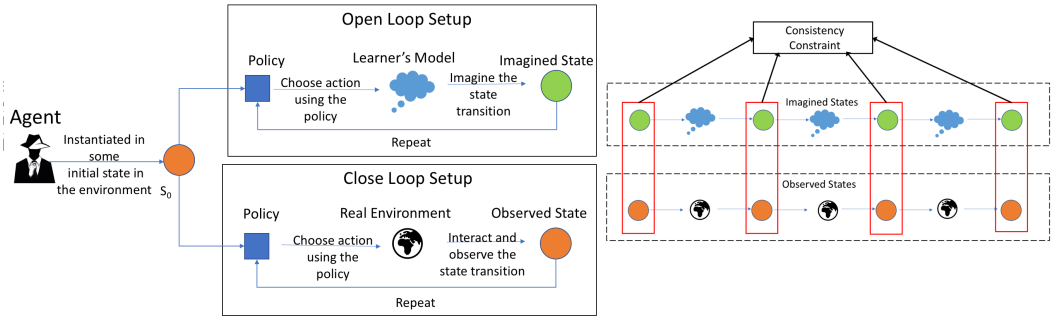



Figure 1: The agent, in parallel, (i) Builds a model of the world and (ii) Engages in an interaction with the world. The agent can now learn the model dynamics while interacting with the environment. We show that making these two pathways consistent helps in simultaneously learning a better policy and a more powerful generative model.

## 2 Consistency Constraint

We impose the consistency constraint by encoding the state transitions (during both open-loop and closed-loop) into fixed-size real vectors using recurrent networks and enforce the output of the recurrent networks to be similar in the two cases. Encoding the sequence can be seen as abstracting out the per-step state transitions into how the dynamics of the environment evolve over time. This way,

we do not focus on mimicking each state but the high-level dynamics of the state transitions and the dynamics model focuses only on information that makes the multi-step predictions indistinguishable from the actual observations from the environment (figure 1). We minimize the $L2$ error between the encoding of predicted future observations as coming from the learner's dynamics model (during open-loop) and the encoding of the future observations as coming from the environment (during closed loop).

Let us assume that the agent started in state $s_0$ and that $a_{0:T-1}$ denote the sequence of actions that the agent takes in the environment from time $t = 0$ to $T - 1$ resulting in state sequence $s_{1:T}$ that the agent transitions through. Alternatively, the agent could have "imagined" a trajectory of state transitions by performing the actions $a_{0:T-1}$ in the learner's dynamics model. This would result in the sequence of states $s^I_{1:T}$. The consistency loss is computed as follows:

$$l_{cc}(\theta, \phi) = \|enc(s_{1:T})) - enc(s^I_{1:T}))\| \tag{1}$$

where $\|\|$ denotes the L2 norm, $enc(s_{1:T})) = RNN([s_1, s_2, ..., s_T])$ and $enc(s^I_{1:T})) = RNN([s^I_1, s^I_2, ..., s^I_T])$. The agent which is trained with the *consistency constraint* is referred to as the *consistent dynamics* agent. The overall loss for such a learning agent can be written as follows:

$$l_{total}(\theta, \phi) = l_{rl}(\theta, \phi) + \alpha l_{cc}(\theta, \phi) \tag{2}$$

$\theta$ refers to the parameters of the agent's transition model $\hat{f}$ and $\phi$ refers to the parameters of the agent's policy $\pi$. The first component, $l_{rl}(\theta, \phi)$, corresponds to the RL objective i.e maximizing expected return and is referred to as the *RL loss*. The second component, $l_{cc}(\theta, \phi)$, corresponds to the loss associated with the *consistency constraint* and is referred to as *consistency loss*. $\alpha$ is a hyper-parameter to scale the *consistency loss component* with respect to the *RL loss*.

We consider both observation space models (where the environment is modeled as a Markov Decision Process) and state-space models where the learning agent encodes the observation into a high-dimensional latent space. State space models are useful when the observation space is high dimensional, as in case of pixel-space observations. For the state space models, the agent learns to model the environment dynamics in the latent space.

## 3 Rationale Behind Using Consistency Loss

Our goal is to provide a mechanism for the agent to have a direct "interaction" between the policy and the dynamics model. This interaction is different from the standard RL approaches where the trajectories sampled by the policy are used to train the dynamics model. In those cases, the model has no control over what kind of data is produced for its training and there is no ("direct") mechanism for the dynamics model to affect the policy, hence a "direct interaction" between the policy and the model is missing. A practical instantiation of this idea is the *consistency loss* where we ensure consistency between the predictions (from the dynamics model) and the actual observations (from the environment). This simple baseline works surprisingly well compared to the state-of-the-art methods (as demonstrated by our experiments).

Our approach is different from just learning a k-step prediction model as in our case, we have two learning signals for the policy: The one from the reinforcement learning loss (to maximize return) and the other due to consistency constraint. This provides a mechanism where learning a model can itself change the policy (thus "interacting" with the policy). In the standard case, the state transition pairs (collected as the agent acts in the environment) become the supervising dataset for learning the model and there is no feedback from the model learning process to the policy.

## 4 Experimental Results

We evaluate how well does the proposed *Consistent Dynamics* model compares against the state-of-the-art approaches for observation space models and state space models. All the results are reported after averaging over 3 random seeds. Note that our simplistic approach outperforms the state-of-the-art *Learning to Query* model (9).

### 4.1 OBSERVATION SPACE MODELS

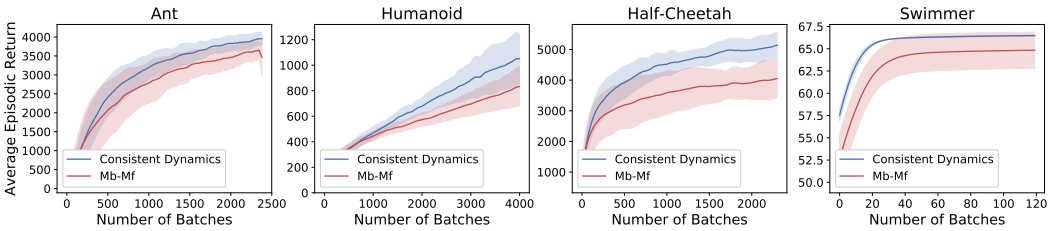



Figure 2: Comparison of the average episodic returns, for *Mb-Mf* agent and *consistent dynamics* agent on the Ant, Humanoid, Half-Cheetah and Swimmer environments (respectively). Note that the results are averaged over 100 batches for Ant, Humanoid and Half-Cheetah and 10 batches for Swimmer.

We use the hybrid Model-based and Model-free (*Mb-Mf*) algorithm (10) as the baseline model. The policy and the dynamics model are learned jointly. We consider 4 Mujoco environments from RLLab (11): Ant ($S \in R^{41}$, $A \in R^{8}$), Humanoid ($S \in R^{142}$, $A \in R^{21}$), Half-Cheetah ($S \in R^{23}$, $A \in R^{6}$) and Swimmer ($S \in R^{17}$, $A \in R^{3}$). For computing the consistency loss, the learner's dynamics model is unrolled for $k = 20$ steps and GRU model is used(12).

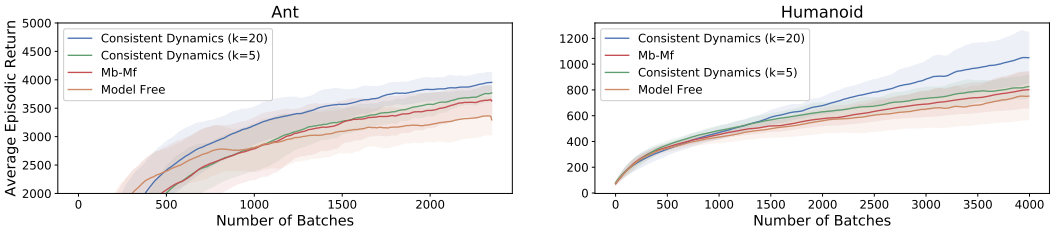



Figure 3: Average episodic return on Ant and Humanoid environments, for a model-free agent, the Mb-Mf agent without any consistency constraint, and the Consistent Dynamics (Mb-Mf + consistency constraint) that are trained with a consistency constraint over time horizons of length 5 and 20. Note that the results are averaged over 100 batches for both Ant and Humanoid.

Figure 2 compares the average episodic returns for the baseline *Mb-Mf* model (does not use consistency loss) and the proposed *consistent dynamics* model (*Mb-Mf* model + consistency loss). Using consistency helps to learn a better policy in fewer updates for all the environments.

We study the effect of changing $k$ (during training) on the average episodic return for the Ant and Humanoid tasks, by training the agents with $k \in \{5, 20\}$. As an ablation, we also include the case of training the policy without using a model, in a fully model-free fashion. We would expect that a smaller value of $k$ would push the average episodic return of the *consistent dynamics* model closer to the *Mb-Mf* case. Figure 3 shows that a higher value of $k$ ($k = 20$) leads to better returns for both tasks.

### 4.2 STATE SPACE MODELS

We use the state-of-the-art *Learning to Query* model (9) as the baseline. We train an expert policy for sampling high-reward trajectories which are used to train the policy (using imitation learning) and the dynamics model (by max-likelihood). We consider 3 continuous control tasks from the OpenAI Gym suite (13): Half-Cheetah, Fetch-Push and Reacher. During the open loop, the dynamics model is unrolled for $k = 10$ steps for Half-Cheetah and $k = 5$ for other tasks.

In figure 4, we compare the imitation learning loss for the *Consistent Dynamics* agent (*Learning to Query* agent with consistency loss) and the baseline (*Learning to Query* agent) and show that consistency constraint helps to learn a more powerful policy. Sampling from recurrent dynamics model is prone to *compounding errors* as even small prediction errors can compound when sampling

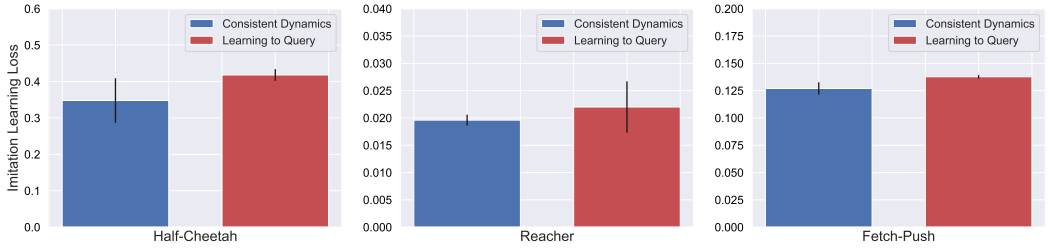
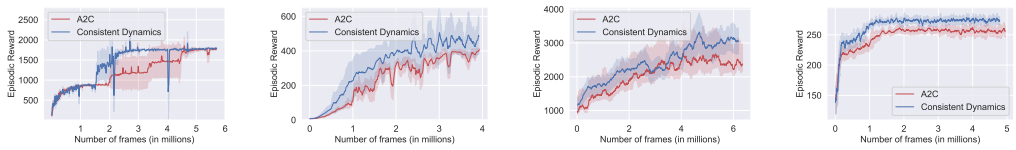



Figure 4: Comparison of imitation learning loss for the *Consistent Dynamics* agent (ie *Learning to Query* agent + consistency loss) and the baseline (just *Learning to Query*) for Half-Cheetah and Reacher environments. The plot for Fetch-Push environment is in the appendix. The bars represents the values corresponding to the trained agent, averaged over the last 50 batches of training. Using consistency constraint leads to a more powerful policy.

for a large number of steps. We evaluate the robustness of the proposed model by unrolling the model for much longer (50 timesteps) than it was trained on (10 timesteps). We observe that the auxiliary cost (which is not solely focused on predicting the next observation) helps to learn a better model




(a) Seaquest (b) Breakout (c) MsPacman (d) Frostbite

Figure 5: Comparison of average episodic return on four Atari environments (Seaquest, Breakout, MsPacman and Frostbite respectively), for the *Consistent Dynamics* agent (ie *A2C* agent + consistency loss) and the baseline (just *A2C*). Using consistency constraint leads to a more powerful policy. Note that the results are average over 100 episodes.

### 4.3 Atari Environment

We evaluate the proposed approach on Atari games (14) using A2C as the baseline model and by adding consistency loss to A2C to obtain the *Consistent Dynamics* model. Specifically, we consider four environments - Seaquest, Breakout, MsPacman, and Frostbite and show that the proposed approach is more sample efficient as compared to the A2C approach thus demonstrating the applicability of our approach to different environments and learning algorithms.

## 5 Conclusion

In this work, we formulate a way to ensure consistency between the predictions of a dynamics model and the real observations from the environment thus allowing the agent to learn powerful policies. We apply an auxiliary loss which encourages the state transitions in the actual environment to be indistinguishable from state transitions in the learner's dynamics model. We consider both observation space and state space models and show that the proposed method outperforms the sophisticated baselines despite being quite simple.

## References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[2] V. Mnih, A. Puigdomènech Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *ArXiv e-prints*, February 2016.

[3] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust Region Policy Optimization. *ArXiv e-prints*, February 2015.

[4] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[5] Erik Talvitie. Model regularization for stable sample rollouts. In *UAI*, 2014.

[6] A. Lamb, A. Goyal, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. *ArXiv e-prints*, October 2016.

[7] Claire Cook, Noah D Goodman, and Laura E Schulz. Where science starts: Spontaneous experiments in preschoolers exploratory play. *Cognition*, 120(3):341–349, 2011.

[8] Bryan C Daniels and Ilya Nemenman. Automated adaptive inference of phenomenological dynamical models. *Nature communications*, 6:8133, 2015.

[9] L. Buesing, T. Weber, S. Racaniere, S. M. A. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, and D. Wierstra. Learning and Querying Fast Generative Models for Reinforcement Learning. *ArXiv e-prints*, February 2018.

[10] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. *ArXiv e-prints*, August 2017.

[11] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. *ArXiv e-prints*, April 2016.

[12] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv e-prints*, June 2014.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[14] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.