Foresee then Evaluate: Decomposing Value Estimation with Latent Future Prediction

Hongyao Tang,[1*] Zhaopeng Meng,[1] Guangyong Chen,[3] Pengfei Chen,[4] Chen Chen,[2] Yaodong Yang,[2] Luo Zhang,[1] Wulong Liu,[2] Jianye Hao[1,2†]

[1]College of Intelligence and Computing, Tianjin University, [2]Noah's Ark Lab, Huawei,
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, [4]The Chinese University of Hong Kong
{bluecontra,mengzp,luozhang}@tju.edu.cn, gy.chen@siat.ac.cn, pfchen@cse.cuhk.edu.hk,
{chenchen9,yang.yaodong,liuwulong,haojianye}@huawei.com

Abstract

Value function is the central notion of Reinforcement Learning (RL). Value estimation, especially with function approximation, can be challenging since it involves the stochasticity of environmental dynamics and reward signals that can be sparse and delayed in some cases. A typical model-free RL algorithm usually estimates the values of a policy by Temporal Difference (TD) or Monte Carlo (MC) algorithms directly from rewards, without explicitly taking dynamics into consideration. In this paper, we propose Value Decomposition with Future Prediction (VDFP), providing an explicit two-step understanding of the value estimation process: 1) first foresee the latent future, 2) and then evaluate it. We analytically decompose the value function into a latent future dynamics part and a policy-independent trajectory return part, inducing a way to model latent dynamics and returns separately in value estimation. Further, we derive a practical deep RL algorithm, consisting of a convolutional model to learn compact trajectory representation from past experiences, a conditional variational auto-encoder to predict the latent future dynamics and a convex return model that evaluates trajectory representation. In experiments, we empirically demonstrate the effectiveness of our approach for both off-policy and on-policy RL in several OpenAI Gym continuous control tasks as well as a few challenging variants with delayed reward.

1 Introduction

Reinforcement learning (RL) is a promising approach to obtain the optimal policy in sequential decision-making problems. One of the most appealing characteristics of RL is that policy can be learned in a model-free fashion, without the access to environment models. Value functions play an important role in model-free RL (Sutton and Barto 1988), which are usually used to derive a policy implicitly in value-based methods (Mnih et al. 2015) or guide the policy updates in policy-based methods (Schulman et al. 2015; Silver et al. 2014). With deep neural networks, value functions can be well approximated even for continuous state and action space, making it practical for model-free RL to deal with more challenging tasks (Lillicrap et al. 2015; Mnih et al. 2015; Silver et al. 2016; Vinyals et al. 2019; Hafner et al. 2020; Schreck, Coley, and Bishop 2019).

Value functions define the expected cumulative rewards (i.e., returns) of a policy, indicating how a state or taking an action under a state could be beneficial when performing the policy. A value function is usually estimated directly from rewards through Monte Carlo (MC) or Temporal Difference (TD) algorithms (Sutton and Barto 1988), without explicitly dealing with the entanglement of reward signals and environmental dynamics. However, learning with such entanglement can be challenging in practical problems due to the complex interplay between highly stochastic environmental dynamics and noisy and even delayed rewards. Besides, it is apparent that the dynamics information is not well considered and utilized during the learning process of value functions. Intuitively, human beings usually not only learn the reward feedback of their behaviors but also understand the dynamics of the environment impacted by their polices. To evaluate a policy, it can commonly be the case of the following two steps: first *foresee* how the environment could change afterwards, and then *evaluate* how beneficial the dynamics could be. Similar ideas called *Prospective Brain* are also studied in cognitive behavior and neuroscience (Atance and O'Neill 2001; Schacter, Addis, and Buckner 2007; Schacter and Addis 2007). Therefore, we argue that it can be important to disentangle the dynamics and returns of value function for better value estimation and then ensure effective policy improvement.

Following the above inspiration, in this paper we consider that value estimation can be explicitly conducted in a two-step way: 1) an agent first predicts the following environmental dynamics regarding a specific policy in a latent representation space, 2) then it evaluates the value of the predicted latent future. Accordingly, we look into the value function and re-write it in a composite form of: 1) a reward-independent predictive dynamics function, which defines the expected representation of future state-action trajectory; and 2) a policy-independent trajectory return function that maps any trajectory (representation) to its discounted cumulative reward. This provides a new decomposed view of value function and induces a trajectory-based way to disentangle the dynamics and returns accordingly, namely Value De-

[*]Work partially done as an intern at Noah's Ark Lab, Huawei.
[†]Corresponding author.

composition with Future Prediction (**VDFP**). VDFP allows the decoupled two parts to be modeled separately and may alleviate the complexity of taking them as a whole. Further, we propose a practical implementation of VDFP, consisting of a convolutional model to learn latent trajectory representation, a conditional variational dynamics model to predict and a convex trajectory return model to evaluate. At last, we derive practical algorithms for both off-policy and on-policy model-free Deep RL by replacing conventional value estimation with VDFP.

Key contributions of this work are summarized as follows.

- We propose an explicit decomposition of value function (i.e., VDFP), in a form of the composition between future dynamics prediction and trajectory return estimation. It allows value estimation to be performed in a decoupling fashion flexibly and effectively.
- We propose a conditional Variational Auto-Encoder (VAE) (Higgins et al. 2017; Kingma and Welling 2014) to model the underlying distribution of future trajectory and then use it for prediction in a latent representation space.
- Our algorithms derived from VDFP outperforms their counterparts with conventional value estimation for both off-policy and on-policy RL in continuous control tasks. Moreover, VDFP shows significant effectiveness and robustness under challenging delayed reward settings.

For reproducibility, we conduct experiments on commonly adopted OpenAI gym continuous control tasks (Brockman et al. 2016; Todorov, Erez, and Tassa 2012) and perform ablation studies for each contribution. Source codes are available at https://github.com/bluecontra/AAAI2021-VDFP.

2 Related Work

Thinking about the future has been considered as an integral component of human cognition (Atance and O'Neill 2001; Schacter and Addis 2007). In neuroscience, the concept of the prospective brain (Schacter, Addis, and Buckner 2007) indicates that a crucial function of the brain is to integrate information past experiences and to construct mental simulations about possible future events. One related work in RL that adopts the idea of future prediction is (Dosovitskiy and Koltun 2017), in which a supervised model is trained to predict the residuals of goal-related measurements at a set of temporal offsets in the future. With a manually designed goal vector, actions are chosen through maximizing the predicted outcomes. Dynamics prediction models are also studied in model-based RL to learn the dynamics model of the environment for synthetic experience generation or planning (Atkeson and Schaal 1997; Sutton 1991). SimPLe (Kaiser et al. 2019) and Dreamer (Hafner et al. 2020) are proposed to learn one-step predictive world models that are used to train a policy within the simulated environment. Multi-steps and long-term future are also modeled in (Hafner et al. 2018; Ke et al. 2019) with recurrent variational dynamics models, after which actions are chosen through online planning, e.g., Model-Predictive Control (MPC). In contrast to seek for a world model, in this paper we care about imperfect latent future prediction underneath the model-free value estimation process, which are closer to human nature in our opinion.

Most model-free deep RL algorithms approximate value functions with deep neural networks to generalize value estimates in large state and action space, e.g., DDPG (Lillicrap et al. 2015), Proximal Policy Optimization (PPO) (Schulman et al. 2017), and Advantage Actor-Critic (A2C) (Mnih et al. 2016). A well-approximated value function can induce effective policy improvement in Generalized Policy Iteration (GPI) (Sutton and Barto 1988). Value functions are usually learned from rewards through TD or MC algorithms, without explicitly considering the dynamics. In this paper, we propose a decomposed value estimation algorithm that explicitly models the underlying dynamics.

A similar idea that decompose the value estimation is the Successor Representation (SR) for TD learning (Dayan 1993). SR assumes that the immediate reward function is a dot-product of state representation and a weight vector, and then the value function can be factored into the expected representation of state occupancy by dividing the weight vector. Thereafter, Deep Successor Representation (DSR) (Kulkarni et al. 2016) extends the idea of SR based on Deep Q-Network (DQN) (Mnih et al. 2015). The SR function is updated with TD backups and the weight vector is approximated from states and immediate rewards in collected experiences. The idea of SR is also further developed in transfer learning (Barreto et al. 2017, 2018). In this paper, we derive the composite function form of value function from the perspective of trajectory-based future prediction, and we show that SR can be viewed as a special linear case of our derivation. In contrast to using TD to learn SR function, we use a conditional VAE to model the latent distribution of trajectory representation. Besides, we focus on the trajectory return instead of immediate reward, which can be more effective and robust for practical reward settings (e.g., sparse and delayed rewards). In a nutshell, our work differs from SR in both start points and concrete algorithms.

3 Background

Consider a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho_0, \gamma, T \rangle$, defined with a state set $\mathcal{S}$, an action set $\mathcal{A}$, transition function $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_{\in[0,1]}$, reward function $\mathcal{R}: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, initial state distribution $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}_{\in[0,1]}$, discounted factor $\gamma \in [0,1]$, and finite horizon T. The agent interacts with the MDP at discrete timesteps by performing its policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, generating a trajectory of states and actions, $\tau_{0:T} = (s_0, a_0, \dots, s_T, a_T)$, where $s_0 \sim \rho_0(s_0)$, $a_t \sim \pi(s_t)$ and $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. An RL agent's objective is to maximize the expected discounted cumulative reward: $J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t | \pi\right]$ where $r_t = \mathcal{R}(s_t, a_t)$.

In RL, state-action value function Q is defined as the expected cumulative discounted reward for selecting action a in state s, then following a policy π afterwards:

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, a_0 = a; \pi\right]. \quad (1)$$

State value function $V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s; \pi\right]$ defines the value of states under policy π similarly.

For continuous control, a parameterized policy π_θ, with parameters θ, can be updated by taking the gradient of the objective $\nabla_\theta J(\pi_\theta)$. In actor-critic methods, a deterministic policy (actor) can be updated with the deterministic policy gradient (DPG) theorem (Silver et al. 2014):

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\pi_\theta(s)}\right], \quad (2)$$

where ρ^π is the discounted state distribution under policy π. Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015) is one of the most representative RL algorithm for continuous policy learning. The critic network (Q-function) is approximated with off-policy TD learning and the policy is updated through the Chain Rule in Equation 2.

4 Value Decomposition of Future Prediction

Value estimation faces the coupling of environmental dynamics and reward signals. It can be challenging to obtain accurate value functions in complex problems with stochastic dynamics and sparse or delayed reward. In this section, we look into value functions and propose a way to decompose the dynamics and returns from the perspective of latent future trajectory prediction.

We first consider a trajectory representation function f that maps any trajectory to its latent representation, i.e., $m_{t:t+k} = f(\tau_{t:t+k})$ for $\tau_{t:t+k} = (s_t, a_t, ..., s_{t+k}, a_{t+k})$ with $k \geq 0$. We then define the trajectory return function U and the predictive dynamics function P as follows:

Definition 1 *The trajectory return function U defines the cumulative discounted reward of any trajectory $\tau_{t:t+k}$ with the representation $m_{t:t+k} = f(\tau_{t:t+k})$:*

$$U\big(f(\tau_{t:t+k})\big) = U(m_{t:t+k}) = \sum_{t'=t}^{t+k} \gamma^{t'-t} r_{t'}. \quad (3)$$

Since U does not depend on a particular policy, it can be viewed as a partial model of the environment that evaluates the overall utility of a trajectory.

Definition 2 *Given the representation function f, the predictive dynamics function P denotes the expected representation of the future trajectory for performing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, then following a policy π:*

$$\begin{aligned} P^\pi(s,a) &= \mathbb{E}\big[f(\tau_{0:T})|s_0 = s, a_0 = a; \pi\big] \\ &= \mathbb{E}\big[m_{0:T}|s_0 = s, a_0 = a; \pi\big]. \end{aligned} \quad (4)$$

Similar to the definition of Q-function, P is associated to policy π, except for the expectation imposed on the trajectory representation. It predicts how the states and actions would evolve afterwards, which is independent with reward.

With above definitions, we derive the following lemma:

Lemma 1 *Given a policy π, the following lower bound of the Q-function holds for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, when function U is convex:*

$$Q^\pi(s,a) \geq U\big(P^\pi(s,a)\big). \quad (5)$$

The equality is strictly established when U is linear.

The proof can be obtained with *Jensen's Inequality* Complete proof can be found in Supplementary Material A. Similar conclusion can also be obtained for state value function V and we focus on Q-function in the rest of the paper.

Lemma 1 provides a lower-bound approximation of the Q-function as a composite function of U and P. When U is a linear function, the equality guarantees that we can also obtain the optimal policy through optimizing the composite function. Note that a linear U actually does not limit the representation power of the whole composite function, since the input of U, i.e., trajectory representation, can be non-linear. Moreover, we suggest that Successor Representation (SR) (Dayan 1993; Kulkarni et al. 2016) can be considered as a special linear case when U is a weight vector and P represents the expected discounted successor state occupancy. For the case that U is a convex function (e.g., a single fully connected layer with ReLU activation), Lemma 1 indicates a lower-bound optimization of the Q-function by maximizing the composite function. There is no theoretical guarantee for the optimality of the learned policy in such cases, although we also found comparable results for convex functions in our experiments (see Ablation 6.3).

The above modeling induces an understanding that the Q-function takes an explicit two-step estimation: 1) it first predicts the expected future dynamics under the policy in terms of latent trajectory representation (i.e., P), 2) then evaluates the utility of latent prediction (i.e., U). This coincides our intuition that humans not only learn from rewards but also understand the dynamics. This provides us a way to decompose the value estimation process by dealing with P and U separately, namely Value Decomposition of Future Prediction (**VDFP**). The decoupling of environmental dynamics and returns reduces the complexity of value estimation and provides flexibility in training and using the decomposed two parts. With a compact trajectory representation, prediction and evaluation of future dynamics can be efficiently carried out in a low-dimensional latent space. Moreover, VDFP draws a connection between model-free RL and model-based RL since the composite function in Lemma 1 indicates an evidence of learning partial or imperfect environment models during model-free value estimation.

Finally, when value estimation is conducted with the composite function approximation in Lemma 1, we can obtain the value-decomposed deterministic policy gradient $\nabla_\theta \tilde{J}(\pi_\theta)$ by extending Equation 2 with the Chain Rule:

$$\begin{aligned} \nabla_\theta \tilde{J}(\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}\big[&\nabla_\theta \pi_\theta(s) \nabla_a P^\pi(s,a)|_{a=\pi_\theta(s)} \\ &\cdot \nabla_m U(m)|_{m=P^\pi(s,a)}\big]. \end{aligned} \quad (6)$$

5 Deep Reinforcement Learning with VDFP

In this section, we implement VDFP proposed in previous section with modern deep learning techniques and then derive a practical model-free Deep RL algorithm from it.

5.1 State-Action Trajectory Representation

The first thing to consider is the representation of state-action trajectory. To derive a practical algorithm, an effective and compact representation function is necessary because:

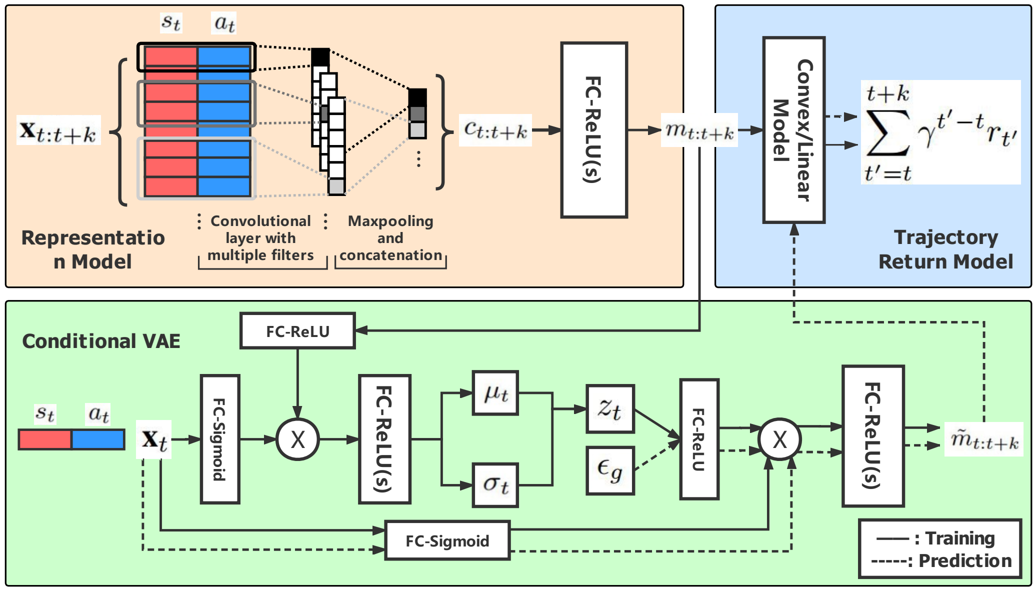


Figure 1: The overall network structure of our models consists of: representation model (*orange*), trajectory return model (*blue*) and conditional VAE (*green*). We abbreviate the Fully-Connected layers as FC (with certain activation) and use $\otimes$ to denote the element-wise product operation. The solid lines illustrate the flow of training process and the dashed lines illustrate the two-step prediction of decomposed value estimation with from the generative decoder (P) to the trajectory return model (U).

1) the trajectories may have variant length, 2) and there may be irrelevant features in states and actions which might hinder the estimation of the cumulative discounted reward of the trajectory. We propose using Convolutional Neural Networks (CNNs) to learn a representation model f^{CNN} of the state-action trajectory, similar to the use for sentence representation in (Kim 2014). In our experiments, we found that this way achieves faster training and better performance than the popular sequential model LSTM (Hochreiter and Schmidhuber 1997) (see Ablations 6.3).

An illustration of f^{CNN} is shown in the orange part of Figure 1. Let $x_t \in \mathbb{R}^l$ be the l-dimensional feature vector of a state-action pair (s_t, a_t). A trajectory $\tau_{t:t+k}$ (padded if necessary) is represented as $\mathbf{x}_{t:t+k} = x_t \oplus x_{t+1} \oplus \cdots \oplus x_{t+k}$, where $\oplus$ is the concatenation operator. A feature $c^i_{t:t+k}$ of trajectory $\tau_{t:t+k}$ can be generated via a convolution operation which involves a filter $\mathbf{w}^i \in \mathbb{R}^{h^i \times l}$ that applied to a window of h^i state-action pairs, and then a max-pooling:

$$\begin{aligned} &c^i_{t:t+k} = \max\{c^i_t, c^i_{t+1} \dots, c^i_{t+k-h^i+1}\}, \\ &\text{where } c^i_j = \text{ReLU}(\mathbf{w}^i \cdot \mathbf{x}_{j:j+h^i-1} + b). \end{aligned} \tag{7}$$

We apply multiple filters $\{\mathbf{w}^i\}^n_{i=1}$ on trajectory $\tau_{t:t+k}$ similarly to generate the n-dimensional feature vector $c_{t:t+k}$, then obtain the trajectory representation $m_{t:t+k}$ after feeding $c_{t:t+k}$ through a Multi-Layer Perceptron (MLP):

$$\begin{aligned} c_{t:t+k} &= \text{concat}(c^1_{t:t+k}, c^2_{t:t+k}, \dots, c^n_{t:t+k}), \\ m_{t:t+k} &= f^{\text{CNN}}(\tau_{t:t+k}) = \text{MLP}(c_{t:t+k}). \end{aligned} \tag{8}$$

Through sliding multiple convolutional filters of different scales along the trajectory, effective features can be extracted with max-pooling thereafter. Intuitively, it is like to scan the trajectories and remember the most relevant parts.

5.2 Trajectory Return Model

With a compact trajectory representation, we consider the trajectory return function U that maps a trajectory representation into its discounted cumulative rewards. As discussed in Section 4, Q-function can be exactly estimated by the composite function when U is linear without decreasing representation power. Therefore, we use a linear U, as shown in the blue part of Figure 1, for following demonstration.

The representation model f^{CNN} and return model U^{Linear} can be trained together with respect to their parameters ω, by minimizing the mean square error of trajectory returns with mini-batch samples from the experience buffer $\mathcal{D}$:

$$\mathcal{L}^{\text{Ret}}(\omega) = \mathbb{E}_{\tau \sim \mathcal{D}}\Big[\big(U^{\text{Linear}}(f^{\text{CNN}}(\tau_{t:t+k})) - y_{t:t+k}\big)^2\Big], \tag{9}$$

where $y_{t:t+k} = \sum_{t'=t}^{t+k} \gamma^{t'-t} r_{t'}$ is the corresponding return of the sampled trajectory $\tau_{t:t+k}$.

Generally, we can use any convex functions for the trajectory return function U. For the simplest case, one can use a single fully-connected layer with ReLU or Leaky-ReLU activation. More sophisticated models like Input Convex Neural Networks (ICNNs) (Amos, Xu, and Kolter 2017; Chen, Shi, and Zhang 2019) can also be considered. The results for using such convex models can be seen in Ablation 6.3.

5.3 Conditional Variational Dynamics Model

One key component of VDFP is the predictive dynamics function P. It is non-trivial to model the expected representation of future trajectory since it reflects the long-term interplay between stochastic environment transition and agent's policy. The most straightforward way to implement the predictive dynamics function is to use a MLP P^{MLP} that takes the state and action as input and predicts the expected representation of future trajectory. However, such a deterministic model performs poorly since it is not able to capture the stochasticity of possible future trajectories. In this paper, we propose a conditional Variational Auto-Encoder (VAE) to capture the underlying distribution of future trajectory representation conditioned on the state and action, achieving significant improvement over P^{MLP} in our experiments. We also provide more insights and experiments on the choice of conditional VAE in Supplementary Material B.

The conditional VAE consists of two parts, an encoder network $q_\phi(z_t|m_{t:t+k}, s_t, a_t)$ and a decoder network $p_\varphi(m_{t:t+k}|z_t, s_t, a_t)$ with variational parameters ϕ and generative parameters φ respectively. With a chosen prior, generally using the multivariate Gaussian distribution $\mathcal{N}(0, I)$, the encoder approximates the conditional posteriors of latent variable z_t from trajectory representation instances, producing a Gaussian distribution with mean μ_t and standard deviation σ_t. The decoder takes a given latent variable as input and generates a representation of future trajectory $\tilde{m}_{t:t+k}$ conditioned on the state-action pair. The structure of the conditional VAE is illustrated in the green part of Figure 1. Specially, we use an element-wise product operation to emphasize an explicit relation between the condition stream and trajectory representation stream.

During the training process of the conditional VAE, the encoder infers a latent distribution $\mathcal{N}(\mu_t, \sigma_t)$ of the trajectory representation $m_{t:t+k}$ conditioned on state s_t and action a_t. A latent variable then is sampled with reparameterization trick, i.e., $z_t = \mu_t + \sigma_t \cdot \mathcal{N}(0, I)$, which is taken as part of input by the decoder to reconstruct the trajectory representation. This process models the underlying stochasticity of potential future trajectories. We train the conditional VAE with respect to the variational lower bound (Kingma and Welling 2014), in a form of the reconstruction loss along with a KL divergence regularization term:

$$\begin{aligned}\mathcal{L}^{\mathrm{VAE}}(\phi, \varphi) = & \mathbb{E}_{\tau_{t:t+k} \sim \mathcal{D}} \Big[\|m_{t:t+k} - \tilde{m}_{t:t+k}\|_2^2 \\ & + \beta D_{\mathrm{KL}}\big(\mathcal{N}(\mu_t, \sigma_t) \| \mathcal{N}(0, I)\big)\Big],\end{aligned} \tag{10}$$

where $m_{t:t+k}$ is obtained from the representation model (Equation 8). We use a weight $\beta > 1$ to encourage VAE to discover disentangled latent factors for better inference quality, which is also known as a β-VAE (Burgess et al. 2018; Higgins et al. 2017). See Ablation (Section 6.3) for the influence of different values of β.

After the training via instance-to-instance reconstruction, we can use the well-trained decoder to generate the instances of trajectory representation with latent variables sampled from the prior $\mathcal{N}(0, I)$. However, this differs from the expected representation we expect for the predictive dynamics model (Equation 4). To narrow down this discrepancy, we propose using a clipped generative noise during the generation process, which allows us to obtain high-quality prediction of expected future trajectories. Finally, the predictive dynamics model P^{VAE} can be viewed as the generation process of the decoder with a clipped generative noise ϵ_g:

$$\begin{aligned}\tilde{m}_{t:t+k} &= P^{\mathrm{VAE}}(s, a, \epsilon_g), \\ \text{where } \epsilon_g &\sim \mathrm{Clip}\big(\mathcal{N}(0, I), -c, c\big) \text{ and } c \geq 0.\end{aligned} \tag{11}$$

Empirically, we found such a clipping can achieve similar results to Monte Carlo sampling but with only one sample. A further discussion on clip value c is in Ablation 6.3.

5.4 Overall Algorithm

With the three modules introduced above, now we derive deep RL algorithms from VDFP. We build our algorithm on off-policy RL algorithm DDPG, by replacing the conventional critic (Q-function) with a decomposed one, namely Value Decomposed DDPG (**VD-DDPG**). Accordingly, the actor (policy) is updated through value decomposed deterministic policy gradient (Equation 6) with respect to the VDFP critic. Note our algorithm does not use target networks for both the actor and critic since no TD target is calculated here. The complete algorithm of VD-DDPG can be found in Supplementary Material C.1 Algorithm 1.

For on-policy RL, we also equip PPO with VDFP for the value estimation of V-function and obtain Value Decomposed PPO (**VD-PPO**). Details and empirical results are omitted and can be found in Supplementary Material C.2.

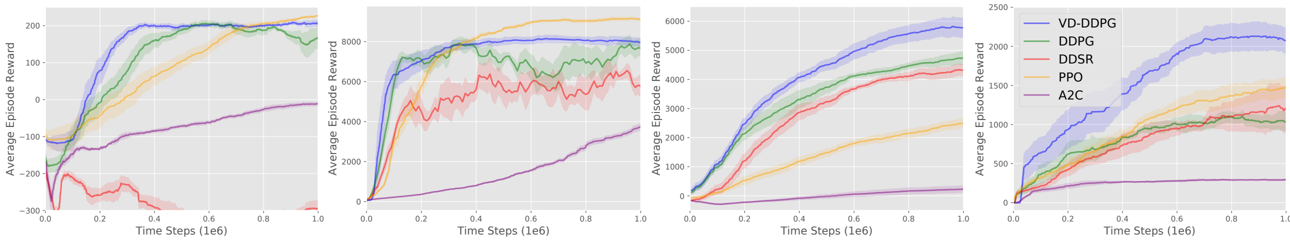
6 Experiments

We conduct our experiments on serveral representative continuous control tasks in OpenAI gym (Brockman et al. 2016; Todorov, Erez, and Tassa 2012). We make no modifications to the original environments or reward functions (except the delay reward modification in Section 6.2).

6.1 Evaluation

To evaluate the effectiveness of VDFP, we focus on four continuous control tasks: one Box2D task LunarLander-v2 and three MuJoCo InvertedDoublePendulum-v1, HalfCheetah-v1 and Walker2d-v1 (Fujimoto, v. Hoof, and Meger 2018; Fujimoto, Meger, and Precup 2018; Schulman et al. 2015, 2017; Peng et al. 2019). We compare our derived approach VD-DDPG with DDPG (Lillicrap et al. 2015), PPO (Schulman et al. 2017), and A2C (Mnih et al. 2016), as well as the Deterministic DSR (DDSR), a variant of DSR (Kulkarni et al. 2016) for continuous control. Since DSR is originally proposed for discrete action problems, we implement DDSR based on DDPG according to the author's codes for DSR on GitHub. VD-DDPG, DDSR and DDPG only differs in the implementation of the critic network to ensure a fair comparison. For PPO and A2C, we adopt non-paralleled implementation and use GAE (Schulman et al. 2016) with $\lambda = 0.95$ for stable policy gradient. Each task is run for 1 million timesteps and the results are reported over 5 random seeds of Gym simulator and network initialization.

For VD-DDPG, we set the KL weight $\beta = 1000$ as in (Burgess et al. 2018) and the clip value c as 0.2. A max trajectory lengthof 256 is used expect using 64 already ensures a good performance for HalfCheetah-v1. An exploration noise sampled from $\mathcal{N}(0, 0.1)$ (Fujimoto, v. Hoof, and



(a) LunarLander-v2 (b) InvertedDoublePendulum-v1 (c) HalfCheetah-v1 (d) Walker2d-v1

Figure 2: Learning curves of algorithms in several OpenAI Gym continuous tasks. Results are average episode reward ($\sum_{t=0}^{T} r_t$) over recent 100 episodes. The shaded region denotes half a standard deviation of average evaluation over 5 trials.

	HalfCheetah-v1				Walker2d-v1			
Algorithm	d = 16	d = 32	d = 64	d = 128	d = 16	d = 32	d = 64	d = 128
VD-DDPG	**4823** (17%↓)	**4677** (**20%**↓)	**3667** (**37%**↓)	**2479** (**57%**↓)	**1822** (14%↓)	**1824** (14%↓)	**1628** (24%↓)	**570** (73%↓)
DDPG	2675 (43%↓)	1546 (67%↓)	1176 (75%↓)	909 (81%↓)	653 (41%↓)	583 (47%↓)	404 (64%↓)	325 (71%↓)
DDSR	1337 (69%↓)	860 (80%↓)	832 (81%↓)	-1 (100%↓)	249 (80%↓)	212 (83%↓)	146 (88%↓)	125 (90%↓)
PPO	1894 (24%↓)	1721 (31%↓)	982 (61%↓)	689 (72%↓)	629 (39%↓)	351 (76%↓)	274 (81%↓)	210 (86%↓)
A2C	207 (**9%**↓)	111 (51%↓)	-35 (116%↓)	-144 (163%↓)	287 (**2%**↓)	281 (**4%**↓)	276 (**6%**↓)	212 (**27%**↓)

Table 1: Performance of algorithms in HalfCheetah-v1 and Walker2d-v1 under the first delayed reward setting. Different delay steps (d) are listed from left to right. Results are max Average Episode Reward over 5 trials of 1 million timesteps. Downarrow (↓) represents the percentage of performance decrease compared with the results under common setting in Figure 2 (i.e., $d = 0$).

Meger 2018) is added to each action selected by the deterministic policy of DDPG, DDSR and VD-DDPG. The discounted factor is 0.99 and we use Adam Optimizer (Kingma and Ba 2015) for all algorithms. Exact experimental details of algorithms are provided in Supplementary Material D.

The learning curves of algorithms are shown in Figure 2. We can observe that VD-DDPG matches or outperforms other algorithms in both final performance and learning speed across all four tasks. Especially, VD-DDPG shows a clear margin over its off-policy counterparts, i.e., DDPG and DDSR. This indicates that VDFP brings better value estimation and thus more effective policy improvement. Similar results are also found in the comparison between VD-PPO and PPO (see Supplementary Material C.2).

6.2 Delayed Reward

One potential advantage of VDFP is to alleviate the difficulty of value estimation especially in complex scenarios. We further demonstrate the significant effectiveness and robustness of VDFP under delayed reward settings. We consider two representative delayed reward settings in real-world scenarios: 1) multi-step accumulated rewards are given at sparse time intervals; 2) each one-step reward is delayed for certain timesteps. To simulate above two settings, we make a simple modification to MuJoCo tasks respectively: 1) deliver d-step accumulated reward every d timesteps;2) delay the immediate reward of each step by d steps.With the same experimental setups in Section 6.1, we evaluate the algorithms on HalfCheetah-v1 and Walker2d-v1 under different delayed reward settings, with a delay step d from 16 to 128. Table 1 plots the results under the first delayed reward setting and similar results are also observed in the second delayed reward setting.

As the increase of delay step d, all algorithms gradually degenerate in comparison with Figure 2 (i.e., $d = 0$). DDSR can hardly learn effective policies under such delayed reward settings due to the failure of its immediate reward model even with a relatively small delay step (e.g., d = 16). VD-DDPG consistently outperforms others under all settings, in both learning speed and final performance (2x to 4x than DDPG). Besides, VD-DDPG shows good robustness with delay step $d \leq 64$. As mentioned in Section 4, we suggest that the reason can be two-fold: 1) with VDFP, it can always learn the dynamics effectively from states and actions, no matter how rewards are delayed; 2) the trajectory return model is robust with delayed reward since it approximates the overall utilities instead of one-step rewards.

6.3 Ablation

We analyze the contribution of each component of VDFP: 1) CNN (v.s. LSTM) for trajectory representation model (Representation); 2) conditional VAE (v.s. MLP) for predictive dynamics model (Architecture); 3) element-wise product (v.s. concatenation) for conditional encoding process (Operator); and 4) model choices (linear v.s. convex) for trajectory return function (Return). We interpolate the slope α for negative input of a single fully-connected layer with Leaky-ReLU activation from 1 (linear) to 0 (ReLU-activated). Specially, we also consider two more advanced convex models, i.e., Input Convex Neural Network (ICNN) (Amos, Xu, and Kolter 2017) and Negation-Extened ICNN (NE-ICNN) (Chen, Shi, and Zhang 2019). We use the same experimental setups in Section 6.1. The results are presented in Table 2 and complete curves are in Supplementary Material E.

First, we can observe that CNN achieves better performance than LSTM. Additionally, CNN also shows lower

Representation		Architecture		Operator					
CNN	LSTM	VAE	MLP	Elem.-Wise Prod.	Concat.	Return Model	KL Weight (β)	Clip Value (c)	Results
✓		✓		✓		Linear	1000	0.2	**5818.60 ± 336.25**
	✓	✓		✓		Linear	1000	0.2	5197.03 ± 156.52
✓			✓	–	–	Linear	–	–	2029.00 ± 486.11
✓		✓			✓	Linear	1000	0.2	4541.71 ± 104.22
✓		✓		✓		Convex (LReLu w/ α=0.2)	1000	0.2	4781.18 ± 440.50
✓		✓		✓		Convex (LReLu w/ α=0.5)	1000	0.2	4985.66 ± 348.08
✓		✓		✓		Convex (ICNN)	1000	0.2	5646.12 ± 328.61
✓		✓		✓		Convex (NE-ICNN)	1000	0.2	5310.23 ± 198.35
✓		✓		✓		Linear	100	0.2	4794.96 ± 370.02
✓		✓		✓		Linear	10	0.2	3933.33 ± 361.82
✓		✓		✓		Linear	1000	∞	4752.84 ± 328.75
✓		✓		✓		Linear	1000	0.0	5712.55 ± 233.74

Table 2: Ablation of VDFP across each contribution in HalfCheetah-v1. Results are max Average Episode Reward over 5 trials of 1 million timesteps. ± corresponds to half a standard deviation. LReLU/Linear denotes a single fully-connected layer with Leaky-ReLU/None activation. Note that Operator, KL Weight and Clip Value are not applicable ('–') for the MLP architecture.

training losses and takes less practical training time (almost 8x faster) in our experiments. indicating CNN is able to extract effective features. Second, the significance of conditional VAE is demonstrated by its superior performance over MLP. This supports our analysis in Section 5.3. Conditional VAE can well capture the trajectory distribution via the instance-to-instance reconstruction and then obtain the expected representation during the generation process. Third, element-wise product shows an apparent improvement over concatenation. We hypothesize that the explicit relation between the condition and representation imposed by element-wise product, forces the conditional VAE to learn more effective hidden features. Lastly, adopting linear layer for trajectory return model slightly outperforms the case of using convex models. This is due to the equality between the composite function approximation and the Q-function (Lemma 1) that ensured by the linear layer. ICNN and NE-ICNN are comparable with the linear case and better than Leaky-ReLU layers due to their stronger representation ability.

Moreover, we analyse the influence of weight β for KL loss term (Equation 10). The results are consistent to the studies of β-VAE (Burgess et al. 2018; Higgins et al. 2017): larger β applies stronger emphasis on VAE to discover disentangled latent factors, resulting in better inference performance. For clip value c in prediction process (Equation 11), clipping achieves superior performance than not clipping ($c = \infty$) since this narrows down the discrepancy between prediction instance and expected representation of future trajectory. Besides, using the mean values of latent variables ($c = 0.0$) does not necessarily ensure the generation of the expected representation, since the decoder network is non-linear. Considering the non-linearity and imperfect approximation of neural networks, setting c to a small value ($c = 0.2$) empirically performs slightly better.

7 Discussion and Conclusion

Discussion. In VDFP, we train a variational predictive dynamics model (conditional VAE) from sampled trajectories. To some extent, it can be viewed as a Monte Carlo (MC) learning at the perspective of trajectory representation. MC is known in traditional RL studies to suffer from high variance and is almost dominated by TD in deep RL. A recent work (Amiranashvili et al. 2018) provides an evidence that finite MC can be an effective alternative of TD in deep RL problems. In our paper, the idea of decomposed MC estimation reflected in VDFP may direct a possible way to address the variance by separately modeling dynamics and returns. Another thing is that the conditional VAE is trained in an off-policy fashion for VD-DDPG. It is supposed to be flawed since trajectories collected by old policies may not able to represent the future under current policy. However, we do not observe apparent adverse effects of using off-policy training in our experiments, and explicitly introducing several off-policy correction approaches shows no apparent benefits. Similar results are also found in DFP (Dosovitskiy and Koltun 2017) and D4PG (Barth-Maron et al. 2018). It may also be explained as in AWR (Peng et al. 2019) which optimizes policy with regard to advantages over an averaged baseline value function that trained from the off-policy data collected by different policies. Analogically, an off-policy trained VAE can be considered to model the average distribution of trajectory representations under recent policies.

VDFP allows flexible uses of decomposed models. We believe it can be further integrated with such as pre-training either part from offline trajectory data, reusing from and transfer to different tasks. Besides, recent advances in Representation Learning (Zhang et al. 2020; Srinivas, Laskin, and Abbeel 2020; Schwarzer et al. 2020a) can also be incorporated. See Supplementary Material C.4 for further discussions. We expect to further investigate the problems discussed above in the future.

Conclusion. we present an explicit two-step understanding of value estimation from the perspective of trajectory-based latent future prediction. We propose VDFP which allows value estimation to be conducted effectively and flexibly in different problems. Our experiments demonstrate the effectiveness of VDFP for both off-policy and on-policy RL especially in delayed reward settings. For future work, it is worthwhile to investigate the problems discussed above.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (Grant Nos.: 61702362, U1836214, U1813204), Special Program of Artificial Intelligence and Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (No.: 56917ZXRGGX00150), Tianjin Natural Science Fund (No.: 19JCYBJC16300), Research on Data Platform Technology Based on Automotive Electronic Identification System.

References

Abdolmaleki, A.; Springenberg, J. T.; Tassa, Y.; Munos, R.; Heess, N.; and Riedmiller, M. A. 2018. Maximum a Posteriori Policy Optimisation. In *ICLR*.

Amiranashvili, A.; Dosovitskiy, A.; Koltun, V.; and Brox, T. 2018. TD or not TD: Analyzing the Role of Temporal Differencing in Deep Reinforcement Learning. In *ICLR*.

Amos, B.; Xu, L.; and Kolter, J. Z. 2017. Input Convex Neural Networks. In *ICML*, 146–155.

Anschel, O.; Baram, N.; and Shimkin, N. 2017. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning. In *ICML*, volume 70, 176–185.

Atance, C. M.; and O'Neill, D. K. 2001. Episodic future thinking. *Trends in Cognitive Sciences* 5: 533–539.

Atkeson, C. G.; and Schaal, S. 1997. Robot Learning From Demonstration. In *ICML*, 12–20.

Barreto, A.; Borsa, D.; Quan, J.; Schaul, T.; Silver, D.; Hessel, M.; Mankowitz, D. J.; Zídek, A.; and Munos, R. 2018. Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. In *ICML*, 510–519.

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; Silver, D.; and v. Hasselt, H. 2017. Successor Features for Transfer in Reinforcement Learning. In *NIPS*, 4055–4065.

Barth-Maron, G.; Hoffman, M. W.; Budden, D.; Dabney, W.; Horgan, D.; TB, D.; Muldal, A.; Heess, N.; and Lillicrap, T. P. 2018. Distributed Distributional Deterministic Policy Gradients. In *ICLR*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *CoRR* abs/1606.01540.

Burgess, C. P.; Higgins, I.; Pal, A.; Matthey, L.; Watters, N.; Desjardins, G.; and Lerchner, A. 2018. Understanding disentangling in β-VAE. *CoRR* abs/1804.03599.

Chen, Y.; Shi, Y.; and Zhang, B. 2019. Optimal Control Via Neural Networks: A Convex Approach. In *ICLR*.

Dayan, P. 1993. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation* 5: 613–624.

Dosovitskiy, A.; and Koltun, V. 2017. Learning to Act by Predicting the Future. In *ICLR*.

Fu, H.; Tang, H.; Hao, J.; Chen, C.; Feng, X.; Li, D.; and Liu, W. 2020. Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning. *CoRR* abs/2009.13891.

Fujimoto, S.; Meger, D.; and Precup, D. 2018. Off-Policy Deep Reinforcement Learning without Exploration. *CoRR* abs/1812.02900.

Fujimoto, S.; v. Hoof, H.; and Meger, D. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML*.

Hafner, D.; Lillicrap, T. P.; Ba, J.; and Norouzi, M. 2020. Dream to Control: Learning Behaviors by Latent Imagination. In *ICLR*.

Hafner, D.; Lillicrap, T. P.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2018. Learning Latent Dynamics for Planning from Pixels. *CoRR* abs/1811.04551.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8): 1735–1780.

Kaiser, L.; Babaeizadeh, M.; Milos, P.; Osinski, B.; Campbell, R. H.; Czechowski, K.; Erhan, D.; Finn, C.; Kozakowski, P.; Levine, S.; Sepassi, R.; Tucker, G.; and Michalewski, H. 2019. Model-Based Reinforcement Learning for Atari. *CoRR* abs/1903.00374.

Ke, N. R.; Singh, A.; Touati, A.; Goyal, A.; Bengio, Y.; Parikh, D.; and Batra, D. 2019. Learning Dynamics Model in Reinforcement Learning by Incorporating the Long Term Future. In *ICLR*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, 1746–1751.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep Successor Reinforcement Learning. *CoRR* abs/1606.02396.

Lan, Q.; Pan, Y.; Fyshe, A.; and White, M. 2020. Maxmin Q-learning: Controlling the Estimation Bias of Q-learning. In *ICLR*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. In *ICLR*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.;

Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Peng, X.; Kumar, A.; Zhang, G.; and Levine, S. 2019. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *CoRR* abs/1910.00177.

Schacter, D. L.; and Addis, D. R. 2007. The cognitive neuroscience of constructive memory: remembering the past and imagining the future. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences* 362 1481: 773–86.

Schacter, D. L.; Addis, D. R.; and Buckner, R. L. 2007. Remembering the past to imagine the future: the prospective brain. *Nature Reviews Neuroscience* 8: 657–661.

Schreck, J. S.; Coley, C. W.; and Bishop, K. J. 2019. Learning retrosynthetic planning through simulated experience. *ACS central science* 5(6): 970–981.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M. I.; and Moritz, P. 2015. Trust Region Policy Optimization. In *ICML*, 1889–1897.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347.

Schwarzer, M.; Anand, A.; Goel, R.; Hjelm, R. D.; Courville, A. C.; and Bachman, P. 2020a. Data-Efficient Reinforcement Learning with Momentum Predictive Representations. *CoRR* abs/2007.05929.

Schwarzer, M.; Anand, A.; Goel, R.; Hjelm, R. D.; Courville, A. C.; and Bachman, P. 2020b. Data-Efficient Reinforcement Learning with Momentum Predictive Representations. *CoRR* abs/2007.05929.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587): 484–489.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. A. 2014. Deterministic Policy Gradient Algorithms. In *ICML*, 387–395.

Srinivas, A.; Laskin, M.; and Abbeel, P. 2020. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. *CoRR* abs/2004.04136.

Srivastava, R.; Shyam, P.; Mutz, F.; Jaskowski, W.; and Schmidhuber, J. 2019. Training Agents using Upside-Down Reinforcement Learning. *CoRR* abs/1912.02877.

Sutton, R. S. 1991. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *SIGART Bulletin* 2(4): 160–163.

Sutton, R. S.; and Barto, A. G. 1988. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks* 16: 285–286.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* 5026–5033.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.; Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575(7782): 350–354.

Zhang, A.; McAllister, R.; Calandra, R.; Gal, Y.; and Levine, S. 2020. Learning Invariant Representations for Reinforcement Learning without Reconstruction. *CoRR* abs/2006.10742.