

Memory-based Deep Reinforcement Learning for POMDPs

Lingheng Meng¹, Rob Gorbet² and Dana Kulic³

Abstract—A promising characteristic of Deep Reinforcement Learning (DRL) is its capability to learn optimal policy in an end-to-end manner without relying on feature engineering. However, most approaches assume a fully observable state space, i.e. fully observable Markov Decision Processes (MDPs). In real-world robotics, this assumption is unpractical, because of issues such as sensor sensitivity limitations and sensor noise, and the lack of knowledge about whether the observation design is complete or not. These scenarios lead to Partially Observable MDPs (POMDPs). In this paper, we propose Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3) by introducing a memory component to TD3, and compare its performance with other DRL algorithms in both MDPs and POMDPs. Our results demonstrate the significant advantages of the memory component in addressing POMDPs, including the ability to handle missing and noisy observation data.

I. INTRODUCTION

Deep Reinforcement Learning (DRL) [1], [2] has been intensively studied in simulated environments, such as games [1] and simulated robots [2], as well as in real-world studies, such as robotics control [3], [4], [5] and human-robot interaction [6], [7], [8]. DRL enables end-to-end policy learning on tasks with high-dimensional state and action spaces, without relying on labour-consuming feature engineering. However, most works focus on developing algorithms [1], [9], [2], [10], [11], [12], [13] for Markov Decision Processes (MDPs) with fully observable state spaces [14], i.e. the observation at each time step fully represents the state of the environment. Few works consider the more complex Partially Observable Markov Decision Process (POMDP) where the observation is just a partial representation of the underlying state. However, POMDPs are ubiquitous in real robotics applications [15], such as robot navigation [16], robotic manipulation [17], autonomous driving [18], [19], and planning under uncertainty [20], [21]. Partial observability may be due to limited sensing capability, or an incomplete system model resulting in uncertainty about full observability.

POMDPs have been tackled with the concept of belief state [22], which represents the agent's current belief about the possible physical states it might be in, given the sequence of actions and observations up to that point. These algorithms

¹Lingheng Meng is with Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada lingheng.meng@uwaterloo.ca

²Rob Gorbet is with Departments of Knowledge Integration and Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada rob.gorbet@uwaterloo.ca

³Dana Kulic is with the Faculty of Engineering, Monash University, 14 Alliance Lane, Melbourne, Victoria, Australia dana.kulic@monash.edu

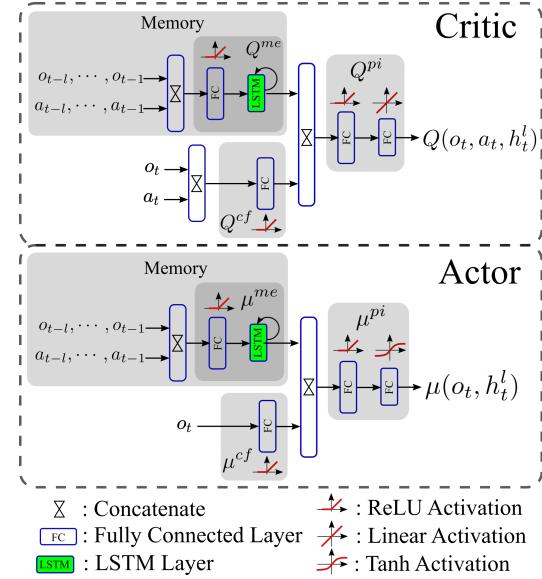


Fig. 1: Recurrent Actor-Critic Framework

are designed to estimate the belief state, then the value function and/or the policy are learned based on the belief state [23]. However, these methods need to know the environment model and the state space and they only work on tasks with small state and action spaces.

POMDPs have also been addressed with DRL, for both discrete [24], [9], [25] and continuous [26], [19] control problems. Recurrent Neural Networks (RNN) have been exploited in DRL to solve POMDPs by considering both the current observation and action, and the history of the past observations and actions [27], [26], [28], [25], [9].

In this paper, we propose a memory-based DRL, called *Long-Short-Term-Memory-based Twin Delayed Deep Deterministic Policy Gradient (LSTM-TD3)*, for continuous robot control. We provide a comparison study with other DRL algorithms where both MDP and POMDP versions of the tasks are investigated to demonstrate how observability properties influence performance on both POMDPs and MDPs. Compared to other DRL algorithms, results show that LSTM-TD3 improves performance significantly on POMDPs where the observation space is disturbed to reduce the observability. We will also provide an ablation study to show the contribution of each design component, and discuss the advantages and disadvantages of the proposed method.

II. RELATED WORK

A number of previous works have investigated how memory can be incorporated into DRL. Deep Recurrent Q-

Learning (DRQN) [24] adds recurrency to the Deep Q-Network (DQN) [29] by replacing the first post-convolutional fully-connected layer with a recurrent Long-Short-Term-Memory (LSTM). The results on Atari 2600 games show DRQN significantly outperforms DQN on POMDPs, which validates the effectiveness of memory extracted by the LSTM for solving POMDPs. [9] investigated a similar idea but augmented the structure with an auxiliary game feature, e.g. presence of enemies, learning in 3D environments in first-person shooter games. The results show the proposed architecture substantially outperforms DRQN. These methods only consider past observations in the history. [25] proposed Action-specific Deep Recurrent Q-Network (ADRQN) to also consider past actions in the memory. However, these works are based on tasks with discrete action spaces, rather than on continuous control tasks.

[27] extended Deterministic Policy Gradient (DPG) [30] to Recurrent DPG (RDPG) by adding LSTM and investigated it on continuous control tasks with partial observations. Dramatic performance improvement was observed with memory. However, even though the observation space was large for some tasks, the action space had relatively few dimensions for the investigated tasks. [26] investigated RDPG on bipedal locomotion tasks with both visual and sensory input, but only one task was examined. Different from directly optimizing RNN, [28] proposed to augment the observations and actions with the continuous memory states and use guided policy search to optimize a linear policy. The method shows better performance than other policy search methods. However, the guided policy search is less powerful and generalizable than non-linear policy.

In our work, we consider continuous control tasks with large observation and action spaces and propose LSTM-TD3 within a recurrent actor-critic framework, which is a further improvement of RDPG by exploiting TD3 to reduce the overestimation problem.

III. BACKGROUND

A. Decision Process

A Markov Decision Process (MDP) is a sequential decision process for a fully observable, stochastic environment with a Markovian transition model and additive rewards [31], [22]. Formally, MDP can be defined as a 4-tuple (S, A, P, R) , where S is the state space, A is the action space, P is the transition probability and R is the reward function. At each discrete time t , an agent selects an action $a_t \in A$ in state $s_t \in S$, transitions to the next state s_{t+1} with probability $P(s_{t+1} | s_t, a_t)$, and receives the immediate reward $R(s_t, a_t, s_{t+1})$. Partially Observable Markov Decision Process (POMDP) [32], [33], [22] is a generalization of a MDP, but does not assume that the state is fully observable, and is defined as a 6-tuple (S, A, P, R, O, Ω) , where S , A , P , and R are the same as that in MDP, with an additional observation space O and observation model Ω . Although the underlying state transition in a POMDP is the same as those in an MDP, the agent cannot observe the underlying state, instead it receives

an observation $o_{t+1} \in O$ when reaching the next state s_{t+1} with the probability $\Omega(o_{t+1} | s_{t+1})$.

The goal for an agent in either MDPs or POMDPs is to choose actions at each time step that maximize its expected future discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where r_t is the immediate reward received at time t and $\gamma \in [0, 1]$ is the discount factor that describes the preference of the agent for current rewards over future rewards.

B. Reinforcement Learning (RL)

RL [34] solves decision problems such as MDPs and POMDPs based on a learning paradigm where an agent learns to act by trial-and-error without knowing the underlying transition and reward model. Specifically, at a discrete time t an agent interacts with the external environment by taking action a_t according to either a stochastic policy $\pi(a_t | s_t)$ or a deterministic policy $a_t = \mu(s_t)$ when observing the current state s_t . By continuously interacting with the environment and receiving new states and rewards, an agent learns an optimal policy to maximize the expected future return. Accompanying the policy π or μ , it is common to learn a state-value function $V(s)$ and/or an action-value function $Q(s, a)$. For MDP, The state-value function $V^\pi(s)$ of a state s under a policy π represents the expected return starting from s and following π thereafter, and the action-value function $Q^\pi(s, a)$ of taking action a in state s under a policy π is the expected return starting from s , taking a , and following π thereafter. For POMDP, since the state s is not observable, the observation o is used in learning these value functions or policy. Traditional RL methods employ either tabular representation or simple function approximation, with hand-crafted feature construction, to represent the value function, while DRL [1] [2] learns the value function and policy in an end-to-end manner without feature engineering by recruiting deep neural networks.

C. Long-Short-Term-Memory (LSTM)

LSTM [35] is a type of Recurrent Neural Network (RNN) [36] that has an outer recurrence from the outputs to the inputs of the hidden layer and also an internal recurrence between LSTM-Cells. Within a LSTM-Cell, a system of gating units control the flow of information, and enable the remembering and forgetting of information given a sequence of inputs.

D. Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 [12] is a variant of Deep Deterministic Policy Gradient (DDPG) [2] designed to address the overestimation problem [37] in Actor-Critic methods. Specifically, TD3 employs two critics Q_1 and Q_2 , and uses the minimum of the predicted optimal future return in state s_{t+1} to bootstrap the Q-value of the current state s_t and action a_t .

IV. PROPOSED APPROACH

In this paper, we propose a memory-based DRL algorithm named LSTM-TD3 within a recurrent actor-critic framework, where both the actor and the critic employ recurrent neural

networks, as illustrated in Fig. 1. In this section, we will first introduce the proposed recurrent actor-critic framework, then present the optimization method for the actor-critic.

In the proposed approach, a mini-batch of N experiences $\{(h_t^l, o_t, a_t, r_t, o_{t+1}, d_t)_i\}_{i=1}^N$ is sampled from the replay buffer D of experiences $(o_t, a_t, r_t, o_{t+1}, d_t)$, where d_t indicates whether the terminal state is reached after observing o_{t+1} and for each sample the past history h_t^l with length l until observation o_t at time t is defined as:

$$h_t^l = \begin{cases} o_{t-l}, a_{t-l}, \dots, o_{t-1}, a_{t-1} & \text{if } l, t \geq 1. \\ o^0, a^0 & \text{Otherwise.} \end{cases} \quad (1)$$

where o^0 and a^0 are the zero-valued dummy observation and action vectors with the same dimensions as those of the normal observation and action. As defined in Eq. 1, if history length $l \geq 1$ and time step $t \geq 1$, the history h_t^l at time t is defined as the past l (observation, action) pairs, otherwise no history is used and zero-valued dummy vectors (observation, action) are used as input to the memory component.

A. Recurrent Actor-Critic Framework

The structure of the proposed recurrent actor-critic framework is illustrated in Fig. 1, where Long-Short-Term-Memory is introduced to extract information beneficial to the actor and critic from past history. The proposed framework can handle history of any length.

Formally, given a mini-batch sample of experiences, the memory-based critic Q , as illustrated in Fig. 1, can be seen as a compound function of the memory extraction Q^{me} , the current feature extraction Q^{cf} , and the perception integration Q^{pi} components, following Eq. 2

$$\begin{aligned} Q(o_t, a_t, h_t^l) &= Q^{me} \circ Q^{cf} \circ Q^{pi} \\ &= Q^{pi}(Q^{me}(h_t^l) \bowtie Q^{cf}(o_t, a_t)) \end{aligned} \quad (2)$$

where \bowtie indicates the concatenation operation, Q^{me} is the extracted memory based on history h_t^l , and Q^{cf} is the extracted current feature based on current observation o_t and action a_t .

Similarly, the memory-based actor μ is also a compound function of the memory extraction μ^{me} , the current feature extraction μ^{cf} , and the perception integration μ^{pi} components, defined as follows:

$$\begin{aligned} \mu(o_t, h_t^l) &= \mu^{me} \circ \mu^{cf} \circ \mu^{pi} \\ &= \mu^{pi}(\mu^{me}(h_t^l) \bowtie \mu^{cf}(o_t)) \end{aligned} \quad (3)$$

where μ^{me} is the extracted memory based on history h_t^l , and μ^{cf} is the extracted current feature based on current observation o_t .

B. Optimization of the Recurrent Actor-Critic

The optimization of the proposed recurrent actor-critic framework follows that of TD3. Specifically, each critic $Q_{j \in \{1, 2\}}$ is optimized to minimize the mean-square-error between the predicted Q_j and the estimated target \hat{Q} with respect to the parameters θ^{Q_j} of the critic Q_j , as follows:

$$\min_{\theta^{Q_j}} \mathbb{E}_{\{(h_t^l, o_t, a_t, r_t, o_{t+1}, d_t)_i\}_{i=1}^N} (Q_j - \hat{Q})^2 \quad (4)$$

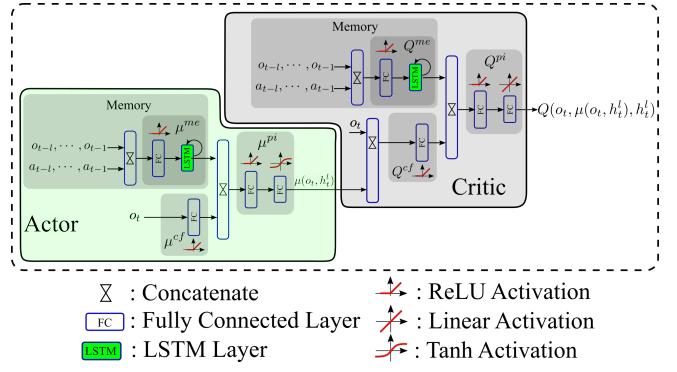


Fig. 2: Actor Optimization, where the parameters θ^{Q_1} of the critic Q_1 is fixed while the parameters θ^μ of the actor is optimized according to Eq. 6.

where given the definition of memory-based critic (Eq. 2) and actor (Eq. 3), the target Q-value \hat{Q} based on the target actor μ^- and critic Q_j^- is defined as Eq. 5

$$\hat{Q} = r_t + \gamma * (1 - d_t) * \min_{j=1,2} Q_j^-(o_{t+1}, a^-, h_{t+1}^l) \quad (5)$$

where $a^- = \mu^-(o_{t+1}, h_{t+1}^l) + \epsilon$ with $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$ and c is the boundary of target action noise, $h_{t+1}^l = (h_t^l - (o_{t-l}, a_{t-l})) \cup (o_t, a_t)$ is the l observation and action pairs before o_{t+1} , and the minimum of the estimated optimal Q-values of the two target critics in (o_{t+1}, h_{t+1}^l) is taken to bootstrap the target Q-value of (o_t, a_t, h_t^l) .

For the actor, its parameters θ^μ are optimized to maximize the approximated Q-value in observation (o_t, h_t^l) and the corresponding estimated optimal action $\mu(o_t, h_t^l)$ with respect to the parameters of the actor, as follows:

$$\max_{\theta^\mu} \mathbb{E}_{\{(h_t^l, o_t)_i\}_{i=1}^N} Q(o_t, \mu(o_t, h_t^l), h_t^l) \quad (6)$$

where the Q could be either of the two critics Q_1 and Q_2 , as in TD3. The pseudo-code for optimizing the recurrent actor-critic can be found in Alg. 1.

V. EXPERIMENT SETTINGS

The tasks (Fig. 3) tested in this work come from PyBullet-Gym¹, an open-source implementation of the OpenAI Gym MuJoCo environment based on BulletPhysics². In this work, an MDP-version and 4 POMDP-versions of each task are investigated, described in Table I. The MDP-version is the original task, as it has a fully observable state-space, while the 4 POMDP-versions simulate different scenarios that potentially cause partial observability in real applications. Specifically, the POMDP-RemoveVelocity (POMDP-RV) is designed to simulate the scenario where the observation space is not well-designed, which is applicable to a novel control task that is not well-understood by researchers and therefore the designed observation may not fully capture the underlying state of the robot. The POMDP-Flickering (POMDP-FLK) models the case where remote sensor data is

¹<https://github.com/benelot/pybullet-gym>

²<https://github.com/bulletphysics/bullet3>

Algorithm 1: Pseudo-code for LSTM-TD3

```

Input: History length  $L$ 
1 Initialize critics  $Q_{\theta Q_1}, Q_{\theta Q_2}$ , and actor  $\mu_{\theta \mu}$  with
   random parameters  $\theta^{Q_1}, \theta^{Q_2}$  and  $\theta^{\mu}$ 
2 Initialize target networks  $\theta^{Q_1^-} \leftarrow \theta^{Q_1}, \theta^{Q_2^-} \leftarrow \theta^{Q_2}$ 
   and  $\theta^{\mu^-} \leftarrow \theta^{\mu}$ 
3 Initialize environment  $o_1 = \text{env.reset}()$ , past history
    $h_1^l \leftarrow \mathbf{0}$ , and replay buffer  $D$ 
4 for  $t = 1$  to  $T$  do
   /* Interacting */
5   Select action with exploration noise
    $a_t \sim \mu_{\theta \mu}(o_t, h_t^l) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$ 
6   Interact and observe new observation, reward,
      and done flag:  $o_{t+1}, r_t, d_t = \text{env.step}(a_t)$ 
7   Store experience tuple  $(o_t, a_t, r_t, o_{t+1}, d_t)$  in  $D$ 
8   if  $d$  then
9     Reset environment  $o_{t+1} = \text{env.reset}()$  and
        history  $h_{t+1}^l \leftarrow \mathbf{0}$ 
10    else
11      /* Update  $h_{t+1}^l$  */
12       $h_{t+1}^l = (h_t^l - (o_{t-l}, a_{t-l})) \cup (o_t, a_t)$ 
13    end
14    /* Learning */
15    Sample mini-batch of  $N$  experiences with their
       corresponding histories
        $\{(h_i^l, o_i, a_i, r_i, o_{i+1}, d_i)\}_{i=1}^N$  from  $D$ 
16    Optimize  $Q_j$  according to Eq. 4
17    Optimize  $\mu$  according to Eq. 6
18    Update target actor-critic
19 end

```

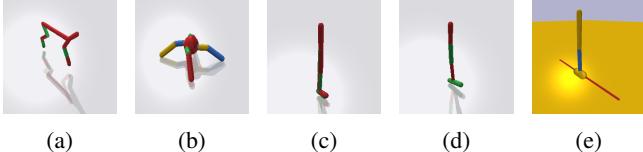


Fig. 3: Example PyBulletGym tasks (a) HalfCheetahPyBulletEnv-v0, (b) AntPyBulletEnv-v0, (c) Walker2DPyBulletEnv-v0, (d) HopperPyBulletEnv-v0, and (e) InvertedDoublePendulumPyBulletEnv-v0.

lost during long-distance data transmission. The case when a subset of the sensors are lost is simulated in POMDP-RandomSensorMissing (POMDP-RSM). Sensor noise is simulated in POMDP-RandomNoise (POMDP-RN).

The baselines used to compare with the proposed LSTM-TD3 are the DDPG [2], SAC [13], TD3 [12], TD3 with Observation-Window (TD3-OW) where the o_t is simply concatenated with the observations within the history window h_t^l to form an observation as input, and TD3 with Observation-Window-AddPastAct (TD3-OW-AddPastAct) where o_t is concatenated with the observations and the actions within the history window h_t^l . The hyperparameters for the baseline algorithms were always the defaults provided in Ope-

TABLE I: MDP- and POMDP-version of Tasks

Name	Description	Hyper-parameter
MDP	Original task	—
POMDP-RV	Remove all velocity-related entries in the observation space.	—
POMDP-FLK	Reset the whole observation to 0 with probability p_{flk} .	p_{flk}
POMDP-RN	Add random noise $\epsilon \sim \mathcal{N}(0, \sigma_{rn})$ to each entry of the observation.	σ_{rn}
POMDP-RSM	Reset an entry of the observation to 0 with probability p_{rsm} .	p_{rsm}

nAISpinningUp³. For the proposed algorithm, hyperparameters were empirically set to that for TD3, and the network structures of the LSTM-TD3 were chosen to have a similar number of parameters to the networks in TD3. All reported results are averaged over 10 evaluation episodes based on 4 different random seeds. The code used for this work can be found in <https://github.com/LinghengMeng/LSTM-TD3>. All hyperparameter testing and additional results (e.g. LSTM-TD3 in POMDPs with lower observability and larger history length than that reported here) are reported in the Supplementary Material (available at <http://arxiv.org/abs/2102.12344>).

VI. RESULTS

A. Performance Comparison

The rows of Fig. 4 show the learning curves of the three sampled tasks, where the first column shows the performance on MDP, while the following 4 columns show results on POMDPs. The results on MDP show that the proposed method has competitive performance to the baselines. The results on the POMDPs highlight the advantage of having memory when solving partially observable tasks. On all types of POMDP, LSTM-TD3 outperforms all baselines, except on POMDP-RV of HalfCheetahPyBulletEnv-v0, where LSTM-TD3(5) shows slightly worse performance than TD3. Although TD3-OW shows better performance than DDPG, TD3, and SAC on POMDPs for most tasks, it still fails for some POMDPs, such as the POMDP-FLK version of most tasks. This reveals that simply concatenating observations is not a good choice, compared to having a LSTM-based memory extraction component as that in LSTM-TD3. LSTM-TD3(0) seems sensitive to random seeds (1st panel of the 3rd row of Fig. 4) as it achieves lower performance compared to that of TD3 and LSTM-TD3 with history length larger than 0. To explain this, even though we set the history for it to zero, it may still predict nonzero for the Q^{me} (introduced in Eq. 2), because the gradients with respect to the randomly initialized weights of Q^{me} may be nonzero and back-propagated, which could influence the agent during learning.

Particularly, a significant performance gap can be observed on POMDP-FLK for all tasks (the 3rd column in Fig. 4),

³<https://spinningup.openai.com>

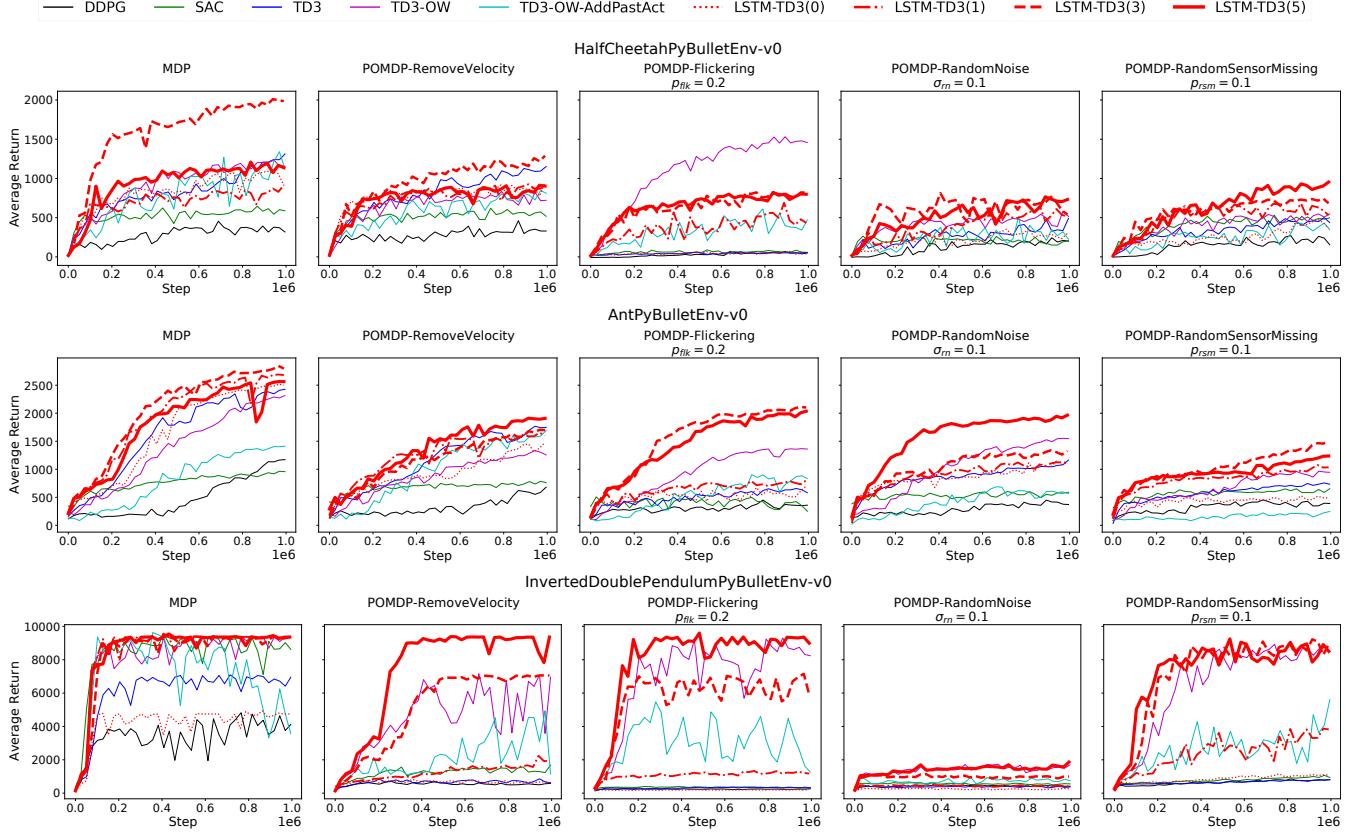


Fig. 4: Learning curves for PyBulletGym tasks, where to ease the comparison only average values are plotted. In the legend, the value in the bracket of LSTM-TD3 indicate the length of the history, e.g. LSTM-TD3(5) uses history length 5.

where the baselines basically fail while LSTM-TD3 achieves comparative performance to that on MDP. This is especially promising for tasks where whole sensor data may be lost, either caused by hardware failure or by temporary occlusion, etc. Similar, dramatic performance improvement can be seen on POMDP-RN and POMDP-RSM.

Surprisingly, comparing LSTM-TD3 and TD3, memory does not always help for POMDP-RV (the 2nd column in Fig. 4) of HalfCheetahPyBulletEnv-v0 and AntPyBulletEnv-v0. Intuitively, if the velocity is important to learn a task, there is no way to infer such information without past observations i.e. memory. However, if previous observations are available, the velocity can be inferred by differences in position between consecutive steps. This intuition can be clearly observed on the POMDP-RV of HopperPyBulletEnv-v0 and InvertedDoublePendulumPyBulletEnv-v0, where the performance of LSTM-TD3 is significantly better than that of TD3. For the HalfCheetahPyBulletEnv-v0 and AntPyBulletEnv-v0, if we compare the performance on MDP and POMDP-RV, we can still see a noticeable gap, which means velocity does contribute to learn a good policy. We hypothesize that LSTM-TD3(5) does not outperform TD3 on the POMDP-RV version of HalfCheetahPyBulletEnv-v0 and AntPyBulletEnv-v0 due to the fact that within the history window all speeds are very similar and velocity cannot be accurately inferred, which may be caused by relatively high sampling rate.

Interestingly, by comparing the results of TD3-OW and TD3-OW-AddPastAct, we found that adding past actions consistently harms the performance compared to TD3-OW, which does not have past actions in its observation window. Even though TD3-OW-AddPastAct still outperforms TD3 on POMDP, it performs worse than TD3 on MDP, which is undesirable if we have no prior knowledge of whether the current design of the observation space is partially or fully observable. Ideally, even if the past action-related information does not provide anything new beyond the past observation, it can be safely ignored and should not harm the performance. We think this is related to the simple construction method where actions are concatenated with observations to form a single observation that includes history information. In this way, the observation dimension is expanded, which makes the learning harder. In addition, this simple construction method treats all observations equally instead of prioritizing the most recent observation, which is normally more valuable in decision-making than earlier observations. This observation based on TD3-OW and TD3-OW-AddPastAct in fact supports our idea to structurally separate the memory extraction and current feature extraction in the recurrent actor-critic framework (Fig. 1) designed for LSTM-TD3, then combine them together to further learn a presentation of the critic and the actor. In section VII-C, we

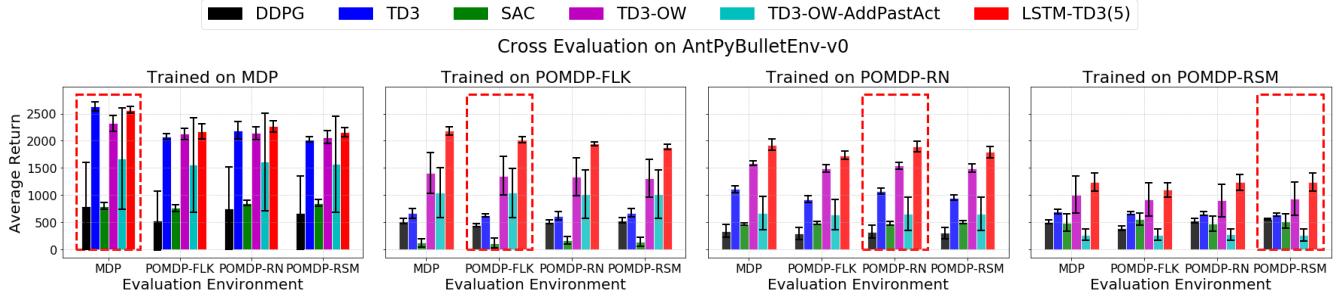


Fig. 5: Cross Evaluation. In each panel, the x-axis indicates the evaluation environment (proposed in Table I) and the y-axis is the average return, where the bars highlighted with red dashed box correspond to performances evaluated on the same environment where the policies are trained, and errorbar on the bar tips indicates the standard deviation of the performance.

will further investigate if adding past actions is beneficial for LSTM-TD3.

B. Policy Generalization

To better understand the generalization of the learned policy using LSTM-TD3, we evaluated the learned policy on a different version of a given task, e.g. if the policy is learned on the MDP-version of a task, and evaluated on the POMDP-versions of the task. This is valuable for real applications where the environment may be non-stationary. Fig. 5 shows the cross evaluation results on AntPyBulletEnv-v0. POMDP-RV is not included as it has a different observation dimension which corresponds to a different input shape for the neural networks. From the first panel (i.e. policies trained on MDP), TD3, TD3-OW, and LSTM-TD3 significantly outperform DDPG, SAC, and TD3-OW-AddPastPact. When evaluating on POMDPs, there is always a decrease for TD3, TD3-OW, and LSTM-TD3, but LSTM-TD3 is the most robust and achieves better performance on these evaluation environments than TD3 and TD3-OW. As for the last three panels, even though LSTM-TD3 still outperforms TD3-OW significantly when evaluated on a different environment, for each algorithm there is not much change in performance. Actually, when trained on POMDP-FLK and evaluated on MDP, LSTM-TD3 achieves a better performance than evaluated on POMDP-FLK.

VII. ABLATION STUDY

To further understand the effect of each component of the proposed LSTM-TD3, in this section we perform an ablation study. Specifically, we examine the effects of the following components: (1) double critics (DC), (2) target policy smoothing (TPS), (3) current feature extraction (CFE), and (4) including past actions (PA) in the history. Fig. 7 shows the learning curves of ablated versions of LSTM-TD3, each removing a different component.

A. Effect of Double Critics and Target Policy Smoothing

As shown in Fig. 7, Full-DC shows a significant decrease in performance compared to Full on all MDPs and most POMDPs, whereas TPS seems less important to the best performance of Full. When simultaneously removing DC and TPS, the performance significantly decreases. Note that

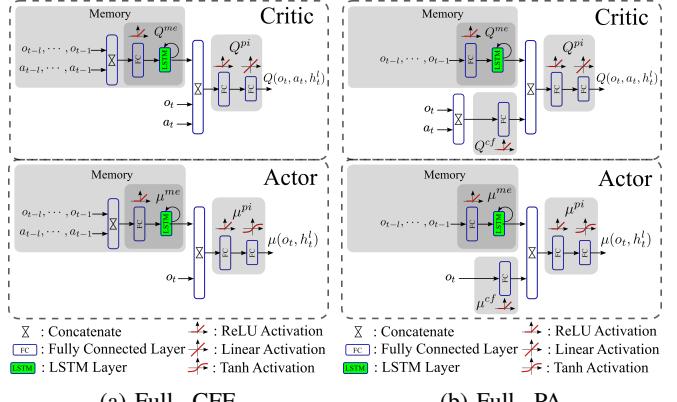


Fig. 6: Diagram of Full-CFE and Full-PA, where for the Full-CFE the extracted memory is directly concatenated with the current observation for the actor and with the current observation and action for the critic; and for the Full-PA past actions are excluded from the history.

without DC and the TPS, the LSTM-TD3 is in fact reduced to LSTM-DDPG, similar to RDPG proposed in [27]. To ease the comparison, we summarized the results of DDPG and LSTM-DDPG, i.e. Full-DC-TPS, in Table II. When compared on the same version of a task, LSTM-DDPG always outperforms DDPG, which can be observed by comparing the results in each row. Remarkably, LSTM-DDPG even achieves significantly better performance on POMDPs than DDPG on MDP.

B. Effect of Current Feature Extraction

In this paper, we intentionally separate the memory extraction and the current feature extraction, then combine them together (Fig. 1), in order to differentiate the current and the past and to reduce the interference from useless information in the memory. Alternatively, we can directly combine the current observation for the action (or the current observation and action for the critic) with the extracted memory, i.e. removing the CFE (Full-CFE) (Fig. 6a). As shown in Fig. 7, Full-CFE performs much worse, compared to Full, especially for MDP-version tasks. Recall that one scenario for devising the LSTM-TD3 is the situation where

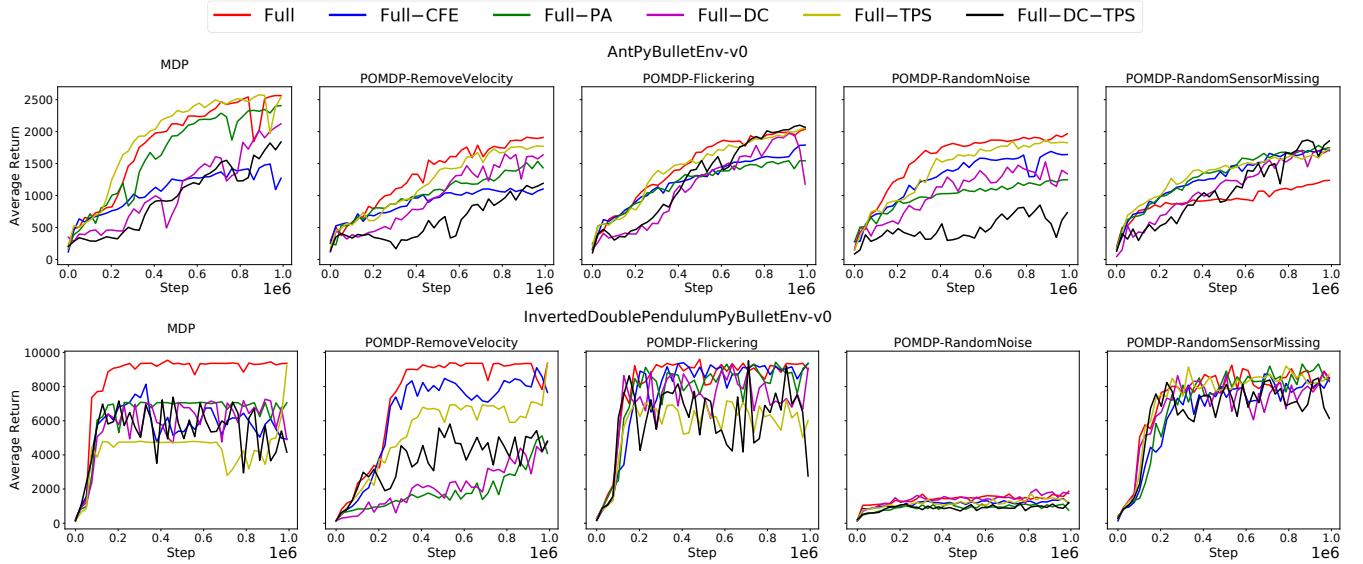


Fig. 7: Learning curves of ablation study, where to ease the comparison only average values over 10 evaluation episodes based on 4 different random seeds are plotted. In the legend, Full, Full–CFE, Full–PA, Full–DC, Full–TPS, and Full–DC–TPS correspond to LSTM-TD3 with full components, removing current feature extraction, excluding past action, not using double critics, not using target policy smoothing, and simultaneously not using double critics and target policy smoothing.

TABLE II: Comparing DDPG and LSTM-DDPG in terms of Maximum Average Return, where \pm indicates a single standard deviation. The bolded value of LSTM-DDPG indicates the performance of LSTM-DDPT on a specific version of a task is better than the performance of DDPG on MDP-version of the task.

Task		Algorithms	
Name	Version	DDPG	LSTM-DDPG
HalfChePB	MDP	487.6 ± 6.1	517.4 ± 102.0
	POMDP-RV	508.4 ± 23.9	552.0 ± 1.4
	POMDP-FLK	84.8 ± 20.4	690.8 ± 0.0
	POMDP-RN	268.7 ± 70.2	731.1 ± 330.9
	POMDP-RSM	283.7 ± 27.0	606.3 ± 63.0
AntPB	MDP	1210.8 ± 226.1	1855.8 ± 494.2
	POMDP-RV	683.5 ± 101.4	1068.6 ± 363.0
	POMDP-FLK	449.0 ± 93.3	2145.1 ± 107.2
	POMDP-RN	449.6 ± 18.5	879.3 ± 446.9
	POMDP-RSM	465.2 ± 51.0	1831.7 ± 33.9

$$p_{flk} = 0.2, \sigma_{rn}=0.1, p_{rsm} = 0.1.$$

engineers are not sure if the design of the observation space is appropriate to capture the state of the agent, if the designed observation space properly captures the state of the agent and there is no CFE, poor performance will be achieved. Therefore, CFE is important for such scenarios.

C. Including Past Action Sequence in Memory

Fig. 6b illustrates Full–PA, where past actions are excluded from the history. As shown in Fig. 7, removing PA causes a decrease in performance, where a remarkable decrease can be observed on the POMDP-RV version of InvertedDoublePendulumPyBulletEnv-v0 (the 2nd panel in the last row in Fig. 7), which is contrary to the observation

in Section VI-A that TD3-OW-AddPastAct performs significantly worse than TD3-OW by adding past actions in the history. This means LSTM-TD3 is more robust than OW-TD3. This is desirable especially when designers have no prior about whether observation of past actions is needed to infer the current state of an agent for an unknown task.

VIII. DISCUSSION AND CONCLUSION

In this paper, we proposed a memory-based DRL algorithm called LSTM-TD3 by combining a recurrent actor-critic framework with TD3. The proposed LSTM-TD3 was compared to standard DRL algorithms on both the MDP- and POMDP-versions of continuous control tasks. Our results show that LSTM-TD3 not only achieves significantly better performance on POMDPs than the baselines, but also retains the state-of-art performance on MDP. Our ablation study shows that all components are essential to the success of the LSTM-TD3 where DC and TPS help in stabilizing learning, CFE is especially important to retain the good performance in MDP, and PA is beneficial for tasks where past actions provide information about the current state of the agent.

The proposed approach is particularly useful when engineers do not have enough knowledge about the environment model and the appropriate design of the observation space to capture the underlying state. Memory can be useful in such a scenario to help infer the underlying state. However, the interpretation of the extracted memory is a challenge. If there is a way to properly interpret the extracted memory, such information, e.g. if the current task is a POMDP or a MDP, can be exploited to improve the observation space design and advance the understanding of the task. Unfortunately, without adding specific constraint terms in the cost functions Eq. 4 and 6 to facilitate the interpretation, there is no way

to properly interpret the extracted memory. Future research should give attention to this direction.

In this paper, for each run of LSTM-TD3 we treat the history length l as a hyper-parameter and fixed it for each run. While LSTM-TD3 with a history length $l = 5$ achieves good performance on the devised POMDPs, this may not be achieved for other tasks where the underlying state depends on less recent memory. However, a long history length increases computation resources and time, during both the training and the inferring, i.e. decision making, phases. In the future, an approach for dynamic adaptation of history length l that achieves the best performance while minimising training and decision making time should be investigated. In addition, more sophisticated POMDP tasks relying on long past history should be examined. SAC with LSTM is also worth to investigate in the future. With the insight of the importance of CFE of LSTM-TD3, TD3-OW with a separate CFE should also be studied and compared to LSTM-TD3.

ACKNOWLEDGMENT

This research was enabled in part by support provided by Compute Canada (www.computecanada.ca). This work is supported by a SSHRC Partnership Grant in collaboration with Philip Beesley Studio, Inc.

REFERENCES

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [3] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates,” in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.
- [4] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, “Towards vision-based deep reinforcement learning for robotic motion control,” *arXiv preprint arXiv:1511.03791*, 2015.
- [5] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.
- [6] A. H. Qureshi, Y. Nakamura, Y. Yoshikawa, and H. Ishiguro, “Robot gains social intelligence through multimodal deep reinforcement learning,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 745–751.
- [7] Y. F. Chen, M. Everett, M. Liu, and J. P. How, “Socially aware motion planning with deep reinforcement learning,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.
- [8] L. Meng, D. Lin, A. Francey, R. Gorbet, P. Beesley, and D. Kulic, “Learning to engage with interactive systems: A field study on deep reinforcement learning in a public museum,” *J. Hum.-Robot Interact.*, vol. 10, no. 1, Oct. 2020. [Online]. Available: <https://doi.org/10.1145/3408876>
- [9] G. Lample and D. S. Chapat, “Playing fps games with deep reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.
- [10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
- [11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [12] S. Fujimoto, H. Van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” *arXiv preprint arXiv:1802.09477*, 2018.
- [13] 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*. PMLR, 2018, pp. 1861–1870.
- [14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, “Benchmarking deep reinforcement learning for continuous control,” in *International conference on machine learning*. PMLR, 2016, pp. 1329–1338.
- [15] K. P. Murphy, “A survey of pomdp solution techniques,” *environment*, vol. 2, p. X3, 2000.
- [16] S. Candido and S. Hutchinson, “Minimum uncertainty robot navigation using information-guided pomdp planning,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 6102–6108.
- [17] J. Pajarinen and V. Kyrki, “Robotic manipulation of multiple objects as a pomdp,” *Artificial Intelligence*, vol. 247, pp. 213–228, 2017.
- [18] V. Talpaert, I. Sobh, B. R. Kiran, P. Mannion, S. Yogamani, A. El-Sallab, and P. Perez, “Exploring applications of deep reinforcement learning for real-world autonomous driving systems,” *arXiv preprint arXiv:1901.01536*, 2019.
- [19] C. Wang, J. Wang, Y. Shen, and X. Zhang, “Autonomous navigation of uavs in large-scale complex environments: A deep reinforcement learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.
- [20] T. Wang, A. Taghvaei, and P. G. Mehta, “Q-learning for pomdp: An application to learning locomotion gaits,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2758–2763.
- [21] R. Cheng, A. Agarwal, and K. Fragkiadaki, “Reinforcement learning of active vision for manipulating objects under occlusions,” in *Conference on Robot Learning*. PMLR, 2018, pp. 422–431.
- [22] S. Russell and P. Norvig, “Artificial intelligence: a modern approach,” 2002.
- [23] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based pomdp solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.
- [24] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” *arXiv preprint arXiv:1507.06527*, 2015.
- [25] P. Zhu, X. Li, P. Poupart, and G. Miao, “On improving deep reinforcement learning for pomdps,” *arXiv preprint arXiv:1704.07978*, 2017.
- [26] D. R. Song, C. Yang, C. McGreavy, and Z. Li, “Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge,” in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 311–318.
- [27] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, “Memory-based control with recurrent neural networks,” *arXiv preprint arXiv:1512.04455*, 2015.
- [28] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, “Learning deep neural network policies with continuous memory states,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 520–527.
- [29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [30] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International conference on machine learning*. PMLR, 2014, pp. 387–395.
- [31] R. Bellman, “A markovian decision process,” *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.
- [32] K. J. Åström, “Optimal control of markov processes with incomplete state information,” *Journal of Mathematical Analysis and Applications*, vol. 10, no. 1, pp. 174–205, 1965.
- [33] A. N. Burnetas and M. N. Katehakis, “Optimal adaptive policies for markov decision processes,” *Mathematics of Operations Research*, vol. 22, no. 1, pp. 222–255, 1997.
- [34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [35] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [36] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.
- [37] S. Thrun and A. Schwartz, “Issues in using function approximation for reinforcement learning,” in *Proceedings of the Fourth Connectionist Models Summer School*. Hillsdale, NJ, 1993, pp. 255–263.

Memory-based Deep Reinforcement Learning for POMDPs: Supplementary Material

Lingheng Meng¹, Rob Gorbet² and Dana Kulic³

Abstract—This manuscript is the supplementary material for the paper "Memory-based Deep Reinforcement Learning for POMDP" by Lingheng Meng, Rob Gorbet and Dana Kulic. It provides additional implementation details, and complementary results.

I. Algorithms Implementation

The implementation of the algorithms is based on OpenAI Spinningup¹. The code used for this work can be found in <https://github.com/LinghengMeng/LSTM-TD3>. Table I details the hyperparameters used in this work, where – indicates the parameter does not apply to the corresponding algorithm. For the actor and critic neural network structure of LSTM-TD3, the first row corresponds to the structure of the memory component, the second row corresponds to the structure of the current feature extraction, and the third row corresponds to the structure of perception integration after combining the extracted memory and the extracted current feature.

II. Supplementary Results

A. Performance Comparison

Table II summarizes the maximum average return of each algorithm on different tasks, where the POMDP-FLK, POMDP-RN, and POMDP-RSM are examined with $p_{flk} = 0.2$, $\sigma_{rn} = 0.1$, and $p_{rsm} = 0.1$. From Table II, we can see that LSTM-TD3 either outperforms or achieves comparative performance to other baselines on MDP, and significantly outperforms other baselines on POMDP-versions of each task. These results provide evidence of the promising advantages of memory based DRL on both MDP and POMDP. On one hand, LSTM-TD3 can be used out-of-box without caring too much of the design of the observation as memory component can partially compensate the lost information. On the other hand, by comparing the performance difference of adding and removing the memory component, LSTM-TD3 provides a way to detect if the current design of the observation space is improvable or not in terms of capturing the underlying state.

¹Lingheng Meng is with Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada lingheng.meng@uwaterloo.ca

²Rob Gorbet is with Departments of Knowledge Integration and Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON, Canada rob.gorbet@uwaterloo.ca

³Dana Kulic is with the Faculty of Engineering, Monash University, 14 Alliance Lane, Melbourne, Victoria, Australia dana.kulic@monash.edu

¹<https://spinningup.openai.com>

TABLE I: Hyperparameters for Algorithms

Hyperparameter	Algorithms			
	DDPG	TD3	SAC	LSTM-TD3
discount factor: γ			0.99	
batch size: N_{batch}			100	
replay buffer size: $ D $				10^6
random start step: N_{start_step}			10000	
update after N_{update_after}			1000	
target NN update rate τ			0.005	
optimizer			Adam [1]	
actor learning rate lr_{actor}			10^{-3}	
critic learning rate lr_{critic}			10^{-3}	
actor NN structure:	[256, 256]		[128] + [128] [128] [128, 128]	
critic NN structure:	[256, 256]		[128] + [128] [128] [128, 128]	
actor exploration noise σ_{act}	0.1		-	0.1
target actor noise σ_{targ_act}	-	0.2	-	0.2
target actor noise clip boundary c_{targ_act}	-	0.5	-	0.5
policy update delay	-	2	-	2
entropy regulation coefficient α	-	-	0.2	-
history length l	-	-	-	{0, 1, 3, 5}

B. Robustness to Partial Observability

Fig. 1 compares the proposed LSTM-TD3 with the baselines in terms of the robustness to different partial observabilities, where the higher the σ_{rn} and the p_{rsm} , the lower the observability. In general, LSTM-TD3 has better robustness than TD3-OW which has better robustness than TD3. A small reduction of observability can be handled by LSTM-TD3, but for POMDP with severe reduction of observability LSTM-TD3 performance also degrades.

C. Effect of History Length

The history length l is a hyperparameter of LSTM-TD3, and it determines the maximum history length in the observation window. As illustrated in Fig. 4 and Table II, LSTM-TD3 with a relatively short history length $l = 5$ produces significantly better performance than other baselines without a memory component. Fig. 2 shows the performance of LSTM-TD3 with different history lengths on POMDP-FLK AntPyBulletEnt-v0 with various flickering probabilities

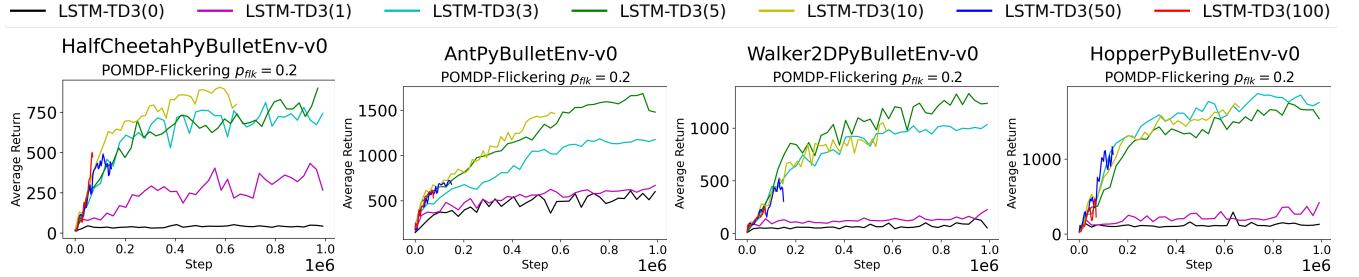


Fig. 3: Performance of LSTM-TD3 with Different History Lengths, where LSTM-TD3 with history length 10, 50 and 100 are not fully run up to 1 million steps due to the extra computation cost caused by long history.

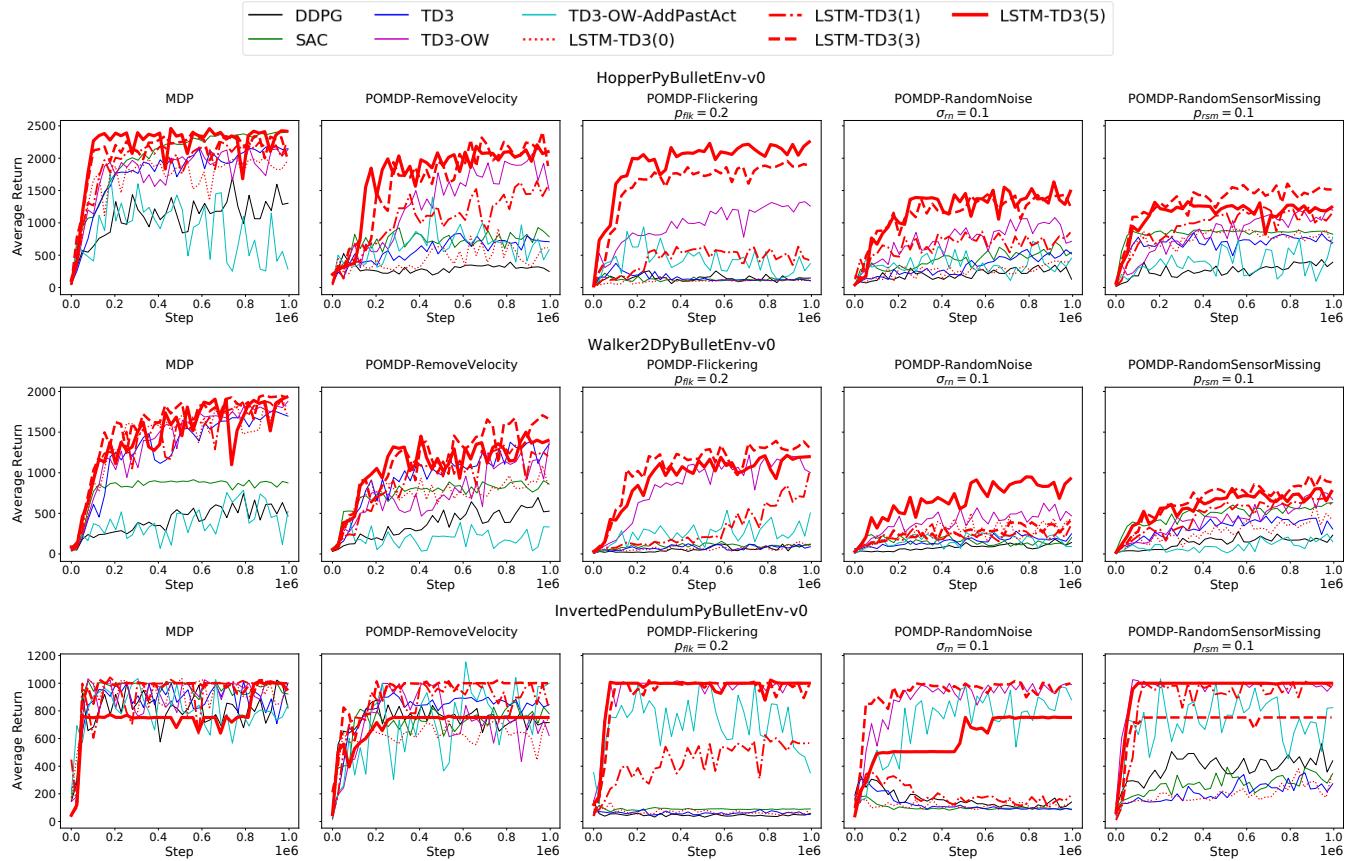


Fig. 4: Learning curves for PyBulletGym tasks, where to ease the comparison only average values are plotted. In the legend, the value in the bracket of LSTM-TD3 indicate the length of the history, e.g. LSTM-TD3(5) uses the history length 5.

improves too. However, the long history length causes much extra computation consumption.

D. Policy Generalization

The second extension is to evaluate the learned policy with different history length from that used during training. If we use l_{train} and l_{eval} to represent the history length used during training and evaluation respectively, the generalization capability for both $l_{train} > l_{eval}$ and $l_{train} < l_{eval}$ can be useful in different scenarios. On one hand, for $l_{train} > l_{eval}$, if l_{train} and l_{eval} can achieve the same performance, using a shorter history length during evaluation can reduce the inferring time of an action, and this is valuable for tasks where

real-time decision making is important and training can be run in parallel and is not time-sensitive. On the other hand, for $l_{train} < l_{eval}$, if l_{eval} can achieve better performance than l_{train} by increasing the history length, which means how to extract useful memory information can be learned with a shorter history length and longer history length during evaluation is only to extract more useful information, then using a shorter history length during training can speed up the training and reduce resource consumption, at the same time without compromising the performance. As for other DRL algorithms, the training is on a mini-batch while the evaluating is only on a single data, so if the l_{train} is large, the training will take more time and computation resources

than evaluation. Therefore, it is a good choice to achieve the same performance by reducing the cost during training and increasing the cost during evaluation. Normally, training cost is unvalued by researchers, especially some research [2], [3] proposed to use large distributed computer cluster. However, there are cases, where the robot cannot communicate fleetly with the remote computation center and onboard computation resources are limited, that the effort in saving computation is still required. This practical consideration inspires the second evaluation extension.

1) Evaluation with Different History Lengths: Fig. 5 illustrates the evaluation results of LSTM-TD3 with various history lengths that may be different from the history length used for training. From this figure, it can be seen that when trained with a specific history length l_{train} but evaluated with a different history length $l_{eval} > l_{train}$, the performance remains at the same level. However, when evaluated with a history $l_{eval} < l_{train}$, the performance cannot be guaranteed, as for some cases a shorter evaluation history length will cause dramatic decrease in performance, e.g. LSTM-TD3(3) on POMDP-RN (the blue line in the 4th panel of Fig. 5), while for others a shorter evaluation history length can still achieve similar performance, e.g. LSTM-TD(5) on POMDP-RN and on POMDP-RSM (the red line in the 4th and 5th panels of Fig. 5). Based on this observation, it seems the performance can be generalized to a longer evaluation history length rather than a shorter one. It is worth to note that the longest training history length investigate here is only 5, so more valuable insights may be found with more results with longer training history length.

The results that when evaluating with history length longer than 0, the performance does not change (the cyan lines in Fig. 5) of LSTM-TD3(0), are very interesting, because in the training phase LSTM-TD3(0) takes zero-valued dummy observation and action as history as defined in Eq. ?? and this dummy history cannot provide any useful information about how to extracting memory that is useful to current task. Therefore, when replacing this dummy history with real history in the replay buffer, the extracted memory can be anything and will disturb the decision making, which makes us to expect that when evaluating the learned policy with history length longer than 0, the performance will be decreased. However, the results is surprisingly remained at the same level as that for LSTM-TD3(0). This reminds us that LSTM-TD3(0) may have learned a policy that intentionally ignores the history. To validate this conjecture, we plotted the average extracted memory of the actor in Fig. 6c. As shown in Fig. 6c, the average extracted memory of the actor of LSTM-TD3(0) initially starts with a non-zero value, but after a few thousands steps its value remains at a value around 0.0034 ± 0.0024 , which is very close to 0, whereas for other LSTM-TD3s with longer history length the average extracted memories are relatively far away from 0 and have relatively large standard deviation. And this observation is consistent for both MDP and POMDPs. Even though we cannot claim the 0 in the extracted memory can be interpreted as neglect of past history, but at least we can say the history

is uniformly mapped to a roughly fixed value rather than a random value for each history. In this way, even replacing the zero-value dummy history with a real history of experiences, the performance will not bad than that for a dummy history.

E. A Glance of The Relationship Among the Return, the Predicted Q-value, and the Extracted Memory of the Actor-Critic

The proposed LSTM-TD3 has been experimentally proved to be useful according to the results presented in Section II-A, but the understanding of the LSTM-TD3, especially the interpretation of the extracted memory, is still a big challenge. In Fig. 6, we presented the average test return, the average predicted Q-value, the average extracted memory of actor and critic to have a glance of the relationship among them.

By comparing Fig. 6a and 6b, we found that the average test return and the average predicted Q-value match approximately perfect, which means there is no overestimation problem as studied in [4], [5], [6] and is desirable.

Our special interest is in the relationship between the test return and the extracted memory of the actor-critic. It is worth to note that both the actor and the critic have a memory component and there is no sharing of the memory component, which means they may learn different coding of the memory that is helpful for learning a Q-value function and a policy, respectively. As shown in Fig. 6c and 6d, no matter if we comparing these plots horizontally or vertically, there is no consistent trend can be found. Horizontally, for both the actor and the critic, the average extracted memories for different versions, i.e. the MDP and the various POMDPs, of the task are showing different trends, where an exception is the extracted memory of the actor of LSTM-TD3(0) as discussed in the previous section II-D.1. Vertically, for different versions of the task, the extracted memories of the actor and the critic show different trends too. Especially, the average extracted memory of the critic of the LSTM-TD3(0) is not similar with that of the actor whose value is consistently close to 0. Contrarily, the average extracted memory of the critic of LSTM-TD3(0) is even further from 0 than that of LSTM-TD3(5). This may indicate the different roles of the memory component playing in the actor and the critic. Again, the interpretation of the extracted memory is not so straight-forward, and special constraints may be forced to improve the interpretability of the memory component, which is out of the scope of this paper and will be left for the future study.

F. Supplementary results for the Ablation Study

Fig. 7 shows the learning curves of various ablated LSTM-TD3 by removing different components, namely (1) using double critics (DC), (2) using target policy smoothing (TPS), (3) having current feature extraction (CFE) component, and (4) including past actions (PA) in the history. Table III reports the maximum average return of the investigated ablated algorithms.

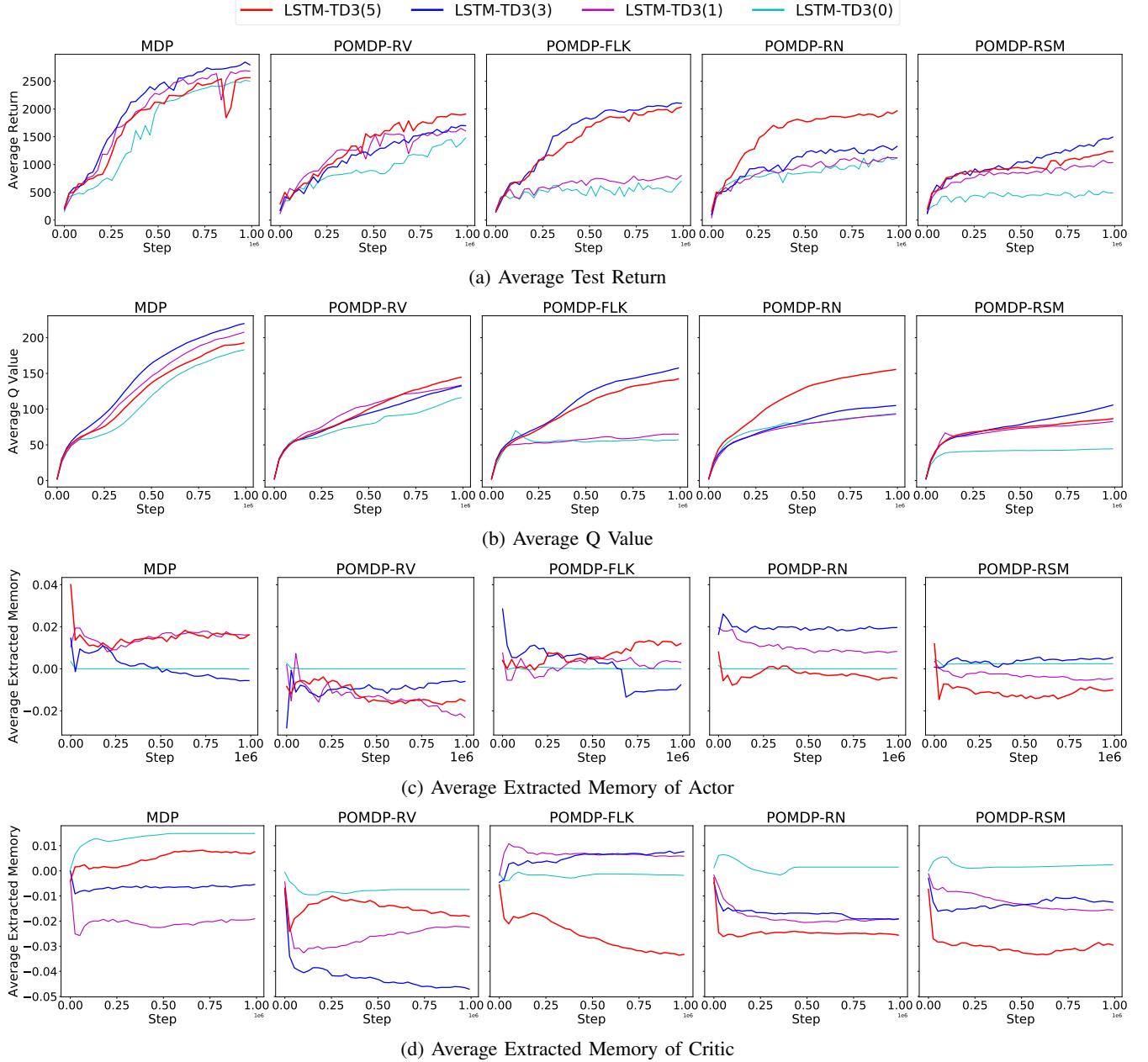


Fig. 6: Relationship Among the Return, Predicted Q-value, Extracted Memory of Actor-Critic, where (a), (b), (c), and (d) shows the average test return, the average predicted Q-value, the average extracted memory of actor, and the average extracted memory of critic. The average extracted memory of actor and critic is an average over all output neurons of the memory component.

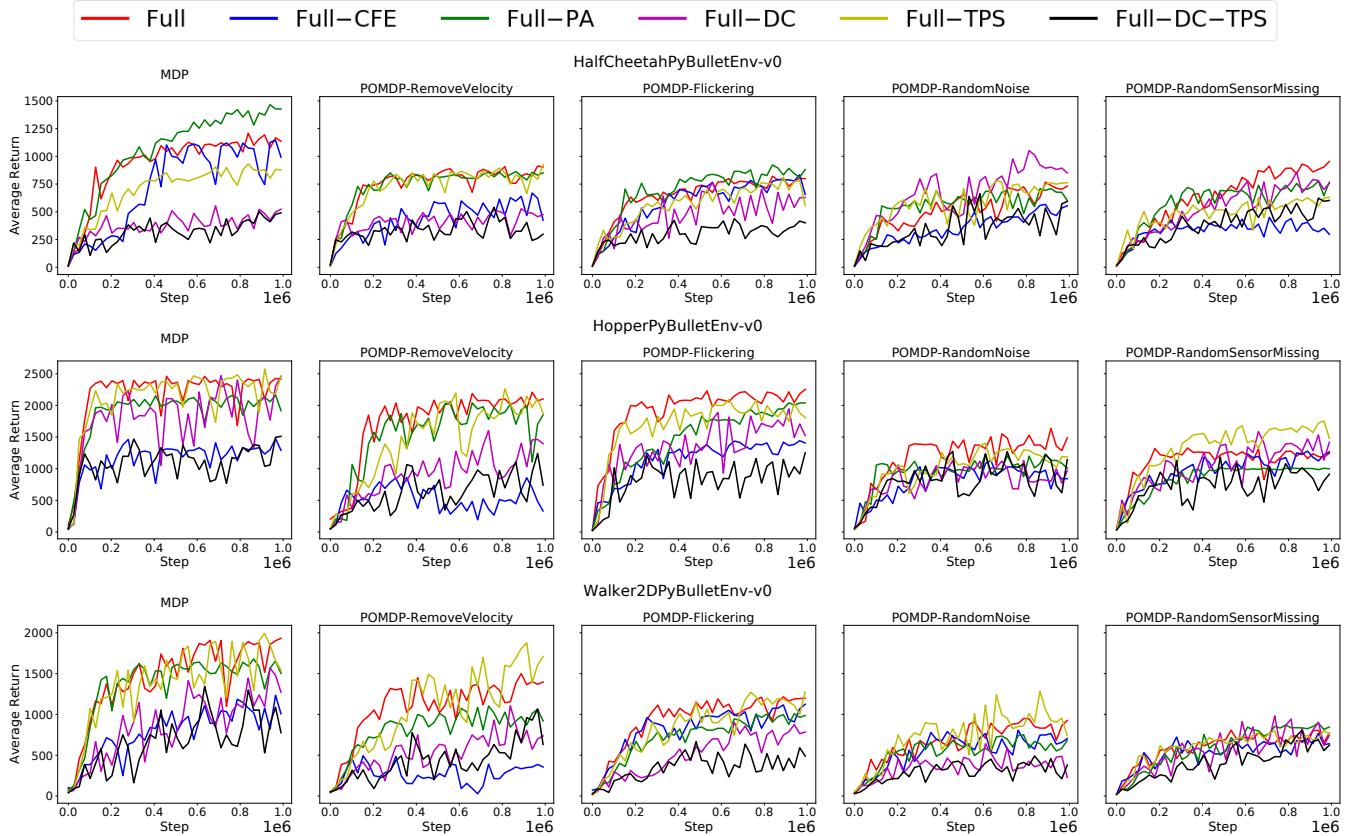


Fig. 7: Learning curves of ablation study, where to ease the comparison only average values are plotted. In the legend, Full, Full–CFE, Full–PA, Full–DC, Full–TPS, and Full–DC–TPS correspond to LSTM-TD3 with full components, removing current feature extraction, excluding past action, not using double critics, not using target policy smoothing, and simultaneously not using double critics and target policy smoothing.