2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Improved Deep Reinforcement Learning for Robotics Through Distribution-based Experience Retention

Tim de Bruin[1] Jens Kober[1] Karl Tuyls[2,1] Robert Babuška[1]

***Abstract*— Recent years have seen a growing interest in the use of deep neural networks as function approximators in reinforcement learning. In this paper, an experience replay method is proposed that ensures that the distribution of the experiences used for training is between that of the policy and a uniform distribution. Through experiments on a magnetic manipulation task it is shown that the method reduces the need for sustained exhaustive exploration during learning. This makes it attractive in scenarios where sustained exploration is in-feasible or undesirable, such as for physical systems like robots and for life long learning. The method is also shown to improve the generalization performance of the trained policy, which can make it attractive for transfer learning. Finally, for small experience databases the method performs favorably when compared to the recently proposed alternative of using the temporal difference error to determine the experience sample distribution, which makes it an attractive option for robots with limited memory capacity.**

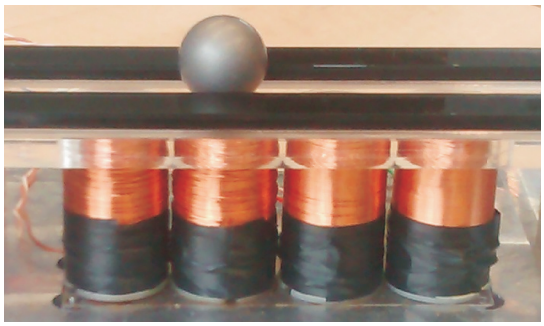


Fig. 1: Magnetic manipulation setup. The horizontal position of the ball needs to be controlled via the four coils. The results in this paper are from a simulation model of this setup.

I. Introduction

Modern day robots are increasingly required to adapt to changing circumstances and to learn how to behave in new and complex environments. Reinforcement Learning (RL) provides a powerful framework that enables them to do this with minimal prior knowledge about their environment or their own dynamics [1]. When applying RL to problems with medium to large state and action dimensions, function approximators are needed to keep the process tractable. Deep neural networks have recently had great successes as function approximators in RL both for robotics [2] and beyond [3], [4].

When RL is used to learn directly from trial and error, the amount of interaction required for the robot to learn good behavior policies can be prohibitively large. Experience Replay (ER) [5] is a technique that can help to overcome this problem by allowing interaction experiences to be re-used. This can make RL more sample efficient, and has proven to be important to make RL with deep neural networks as function approximators work in practice [6], [7].

As reported previously [8], RL with deep neural network function approximators can fail when the experiences that are used to train the neural networks are not diverse enough. When learning online, or when using experience replay in the standard manner, in which experiences are added in a First In First Out (FIFO) manner and sampled uniformly, this effectively translates to a requirement to always keep on exploring. This can be problematic when using RL on physical systems such as robots, where continued thorough exploration leads to increased wear or even damage of the system. Additionally, this can lead to bad task-performance of the robot while learning.

In this paper a method is proposed in which two experience replay databases are deployed. One is filled with experiences in the standard FIFO manner, while in the other one the experiences are overwritten with new experiences in order to get an approximately uniform distribution over the state-action space. By sampling experiences from both databases when training the deep neural networks, the detrimental effects of reduced exploration can be limited. This method is tested on a simulated magnetic manipulation task (Figure 1).

This work is closely related to [9] where an experience replay strategy is proposed in which all experiences are saved, but the sampling procedure is based on the temporal difference error. We show that when a small database is used, the temporal difference error does not yield good results. In [3] a model free RL method with deep neural network function approximation was proposed in which no experience replay was used. However, this method requires several different exploration policies to be followed simultaneously, which seems implausible outside of simulation.

The remainder of this paper[1] is organized as follows: Section II explains the used deep reinforcement learning

[1]All authors are with the Delft Center for Systems and Control, Delft University of Technology. {t.d.debruin, j.kober, r.babuska}@tudelft.nl

[2]Karl Tuyls is with the Department of Computer Science, University of Liverpool. K.Tuyls@liverpool.ac.uk

This work is part of the research programme Deep Learning for Robust Robot Control (DL-Force) with project number 656.000.003, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO).

[1]Also see accompanying video.

method, some preliminaries about experience replay and our proposed extension. In Section III the magnetic manipulator problem on which our method is tested is discussed. Then in Section IV we examine the properties and the performance of our method on the magnetic manipulator problem. We also compare the method to alternatives.

II. Method

The experience replay method proposed in this paper is used in combination with the Deep Deterministic Policy Gradient (DDPG) algorithm presented in [6]. The results are however expected to apply similarly to other deep reinforcement learning methods that make use of experience replay.

A. Deep Deterministic Policy Gradient (DDPG)

The DDPG method is an off-policy actor-critic reinforcement learning algorithm. Actor-critic algorithms are interesting for robot control, as they allow for continuous action spaces. This means that smooth actuator control signals can be learned.

The algorithm uses an actor and a critic. The actor π attempts to determine the real-valued control action $a^\pi \in \mathbb{R}^n$ that will maximize the expected sum of future rewards r based on the current state of the system $s \in \mathbb{R}^m$; $a^\pi = \pi(s)$. The critic Q predicts the expected discounted sum of future rewards when taking action $a(k)$ in state $s(k)$ at time k and the policy π is followed for all future time steps:

$$Q^\pi(s,a) = \mathbb{E}\Bigg[\Big(r\Big(s(k),a,s(k+1)\Big) + \sum_{j=k+1}^{\infty} \gamma^{j-k} r\big(s(j),\pi(s(j)),s(j+1)\big)\Big| s(k)=s\Bigg] \tag{1}$$

with $0 \leq \gamma < 1$ the discount factor, which is used to ensure this sum is finite.

The actor and critic functions are approximated by neural networks with parameter vectors ζ and ξ respectively. The critic network weights ξ are updated to minimize the squared temporal difference error:

$$\mathcal{L}(\xi) = \Big(\big[r + \gamma Q\left(s', \pi(s'|\zeta^-)\middle|\,\xi^-\right)\big] - Q(s,a|\xi)\Big)^2 \tag{2}$$

Where $s = s(k)$, $s' = s(k+1)$ and $r = r(s,a,s')$ for brevity. The parameter vectors ζ^- and ξ^- are copies of ζ and ξ that are updated with a low-pass filter to slowly track ζ and ξ:

$$\xi^- \leftarrow \tau\xi + (1-\tau)\xi^- \tag{3}$$

$$\zeta^- \leftarrow \tau\zeta + (1-\tau)\zeta^-. \tag{4}$$

This improves the stability of the learning algorithm [6]. The parameter τ determines how quickly the ζ^- and ξ^- track ζ and ξ. Values of τ close to one result in fast yet unstable learning, whereas small values of τ result in slow yet stable learning. Here $\tau = 10^{-2}$ is used.

The actor network is updated in the direction that will maximize the expected reward according to the critic:

$$\Delta\zeta \sim \nabla_a Q(s,a|\xi)|_{s=s(k),a=\pi(s(k)|\zeta)} \nabla_\zeta \pi(s|\zeta)|_{s=s(k)} \tag{5}$$

B. Experience Replay

The use of an off-policy algorithm is very relevant for robotics as it allows for experience replay [5] to be used. When using experience replay, the experience tuples $\langle s, a, s', r\rangle$ from the interaction with the system are stored in a database. During the training of the neural networks, the experiences are sampled from this database, allowing them to be used multiple times. The addition of experience replay aids the learning in several ways. The first benefit is the increased sample efficiency by allowing samples to be reused. Additionally, in the context of neural networks, experience replay allows for mini-batch updates which improves the computational efficiency, especially when the training is performed on a GPU.

On top of the efficiency gains that experience replay brings, it also improves the stability of RL algorithms that make use of neural network function approximators such as DQN [7] and DDPG [6]. One way in which the database helps stabilize the learning process is that it is used to break the temporal correlations of the neural network learning updates. Without an experience database, the updates of (2), (5) would be based on subsequent experience samples from the system. These samples are highly correlated since the state of the system does not change much between consecutive time-steps. For real-time control, this effect is even more pronounced with high sampling frequencies. The problem this poses to the learning process is that most mini-batch optimization algorithms are based on the assumption of independent and identically distributed (i.i.d.) data [6]. Learning from subsequent samples would violate this i.i.d. assumption and cause the updates to the network parameters to have a high variance, leading to slower and potentially less stable learning [10]. By saving the experiences over a period of time and updating the neural networks with mini-batches of experiences that are sampled uniformly random from the database this problem is alleviated.

1) Effects of the Experience Sample Distributions: In this paper we use a deterministic policy in combination with an actor-critic algorithm that uses Q-learning updates. This means that in theory, no importance sampling is needed to compensate for the fact that we are sampling our experiences off-policy [11].

However, we are using deep neural networks as global function approximators for the actor and the critic. After every episode E the critic network is updated to minimize an estimate of the loss function (2), the empirical loss:

$$\mathcal{E}(\xi) = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}} \Big(r_i + \gamma Q\left(s_i', \pi(s_i'|\zeta^-)\middle|\,\xi^-\right) - Q(s_i,a_i|\xi)\Big)^2 \tag{6}$$

Here i are the samples in the experience replay database $\mathcal{D}$ after episode E. The distribution of the samples over the state-action space clearly determines the contribution of the approximation accuracy in these regions to the empirical loss. An approximation of Q that is very precise in a part of the state-action space that has many samples but imprecise in a region with few experience samples might result in a low

empirical loss. Meanwhile, an approximation that is more or less correct everywhere might result in a higher empirical loss. From (5) it can be seen that when the critic is not accurate for a certain region of the state action space, the updates to the actor will also likely be wrong.

Additionally, even if a neural network has previously learned to do a task well, it can forget this knowledge completely when learning a new task, even when the new task is related to the old one [12]. In the case of the DDPG algorithm, even if the critic can accurately predict the expected future sum of rewards for parts of the state-action space, this ability can disappear when it is no longer trained on data from this part of the state-action space, as the same parameters apply to the other parts of the state-action space as well and might be changed to reduce the temporal difference error there.

We earlier observed [8] that when the experiences are sampled by exclusively following a deterministic policy without exploration, even a good one, the DDPG method fails. Since sufficient exploration prevents this problem, this seems to imply that having a value function and policy that generalize to the whole state-action space to at least some extent is important. Therefore, we would like to have at least some sample density over the whole state-action space. We are however mostly interested in those areas of the state-action space that would actually be encountered when performing the task. We therefore want most of our experiences to be in this region. For optimal performance we therefore need to find a trade-off between both criteria. Furthermore, these properties of the database distribution should ideally hold after all episodes E.

In Section IV experiments are shown that investigate the influence of the experience sample distribution over the state-action space. These experiments indeed show that the ideal distribution is likely to be somewhere between the distribution that results from simply following the most recent policy with some exploration and a uniform distribution over the state-action space.

2) Distribution Based Experience Retention: The most common experience replay method, which is used in the DQN [7] and DDPG [6] papers, is to use a database $\mathcal{D}$ that is overwritten in a First In First Out (FIFO) fashion. The experiences are then sampled from this database uniformly random. This will however in general not yield a desirable distribution of the experiences in the sampled batches over the state-action space. In fact, as will be shown in Section IV, when at some point during the training the amount of exploration is reduced too far, the performance of the controller policy will decrease.

Maintaining high levels of exploration might place infeasible demands on physical systems such as robots. On these systems, continued extensive exploration might cause increased wear or damage or be simply impossible because the robot is required to perform a task adequately while learning is under way.

In this paper, a method is proposed to maintain a desirable distribution over the state-action space of the experiences in the batches used to update the neural networks.

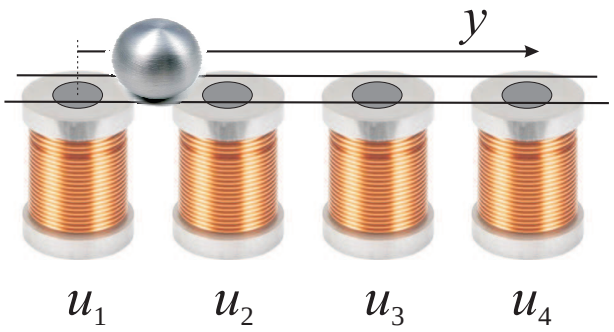


Fig. 2: Experimental setup schematic.

The proposed method is based on having two experience databases of limited size. The first database $\mathcal{D}_\pi$ is overwritten in the standard FIFO manner. The distribution of the experience samples in this database will therefore correspond approximately to the current policy.

For the second database $\mathcal{D}_U$, the experiences in the database are overwritten by the new experiences in such a way that we approximate a uniform distribution over the state-action space. To do this, after this experience database has been filled to capacity, each new experience will overwrite the experience i already in this database that is most likely under the distribution induced by the experiences j already contained in $\mathcal{D}_U$. The following distance metric, employing kernel density estimation [13] on the experiences in the database $\mathcal{D}_U$, is used to determine which experience will be overwritten:

$$i_{\text{overwrite}} = \underset{i \in \mathcal{D}}{\operatorname{argmax}} \frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} e^{-\sum_{d=1}^{D_N} (i_d - j_d)^2 / C_d} \qquad (7)$$

where d are the dimensions in the state-action space, D_N is the total dimensionality of the state-action space and C_d is a dimension dependent scaling constant. Here, C_d is chosen as $|d|/C$ with d the size of the considered part of that state-action dimension. C is a constant that is dependent on the size of the database and the properties of the distribution. It is chosen manually based on the approximation quality of the sample distribution.

When training the neural networks, experiences are drawn uniformly random from $\mathcal{D}_U$ with probability β and uniformly random from $\mathcal{D}_\pi$ with probability $(1-\beta)$. The constant β represents a trade-off between generalization performance and task performance. Additionally, the value of β could be increased when the amount of exploration is reduced to prevent loss of performance.

III. Experiments

To test the proposed method, control policies are learned for a simulated magnetic manipulation task.

A. MAGMAN

Magnetic manipulation is contactless, which opens up new possibilities for actuation on a micro scale and in

environments where it is not possible to use traditional actuators. An example of this type of application are medical micro and nano robots [14].

Our magnetic manipulation setup (Figures 1 and 2) has four electromagnets in a line. The current through the electromagnet coils is controlled to dynamically shape the magnetic field above the electromagnets and so to position a steel ball accurately and quickly to a desired set point. The ball position is measured by a laser sensor.

The horizontal acceleration of the ball is given by:

$$\ddot{y} = -\frac{b}{m}\dot{y} + \frac{1}{m}\sum_{i=1}^{4} g(y,i)\, u_i \tag{8}$$

with

$$g(y,i) = \frac{-c_1\ (y - 0.025i)}{\left((y-0.025i)^2 + c_2\right)^3}. \tag{9}$$

Here, y denotes the position of the ball, $\dot{y}$ its velocity and $\ddot{y}$ the acceleration. With u_i the current through coil $i = 1, 2, 3, 4$, $g(y, i)$ is the nonlinear magnetic force equation, m [kg] the ball mass, and b $[\frac{Ns}{m}]$ the viscous friction of the ball on the rail. The model parameters are listed in Table I.

TABLE I: Magnetic manipulation system parameters

Model parameter	Symbol	Value	Unit
Ball mass	m	$3.200 \cdot 10^{-2}$	kg
Viscous damping	b	$1.613 \cdot 10^{-2}$	Nms
Empirical parameter	c_1	$5.520 \cdot 10^{-10}$	$\mathrm{Nm^5A^{-1}}$
Empirical parameter	c_2	$1.750 \cdot 10^{-4}$	$\mathrm{m^2}$
Sampling period	T_s	0.02	s

The reinforcement learning state s is given by the the position and velocity of the ball. The action a is defined as the vector of currents $u_1 \ldots u_n \in [0, 0.6]$ to the coils. The reward function is defined as:

$$r(s) = -\left(100\,|y - y_r| + 5\,|\dot{y}|\right) \tag{10}$$

where the reference position y_r is set to $y_r = 0.035m$.

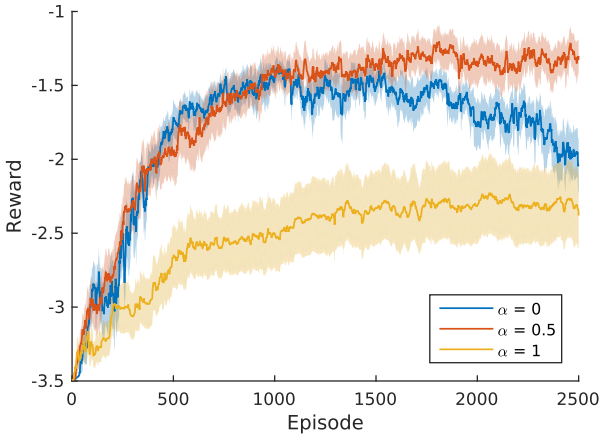
For the theoretical experiments, simulations have been performed with 3 coils. In these experiments the ball always starts with the position and velocity equal to zero. We measure the performance of the controller in two ways:

- The *task* performance: the average reward for an episode when using the same initial conditions as were used during training.
- The *generalization* performance: the average reward for an episode when starting from several different initial positions and velocities.

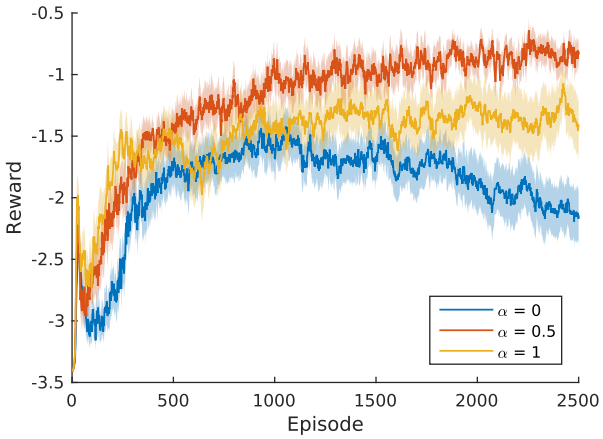
On physical systems such as robots, continued thorough exploration is not always desirable or even feasible. To reflect this fact, the amount of exploration in the experiments is decayed exponentially per episode for all experiments.

IV. Results

To investigate the merits of the method proposed in Section II, several experiments are conducted on the magnetic manipulation problem described in Section III.



(a) Performance on the training task



(b) Performance on the generalization task

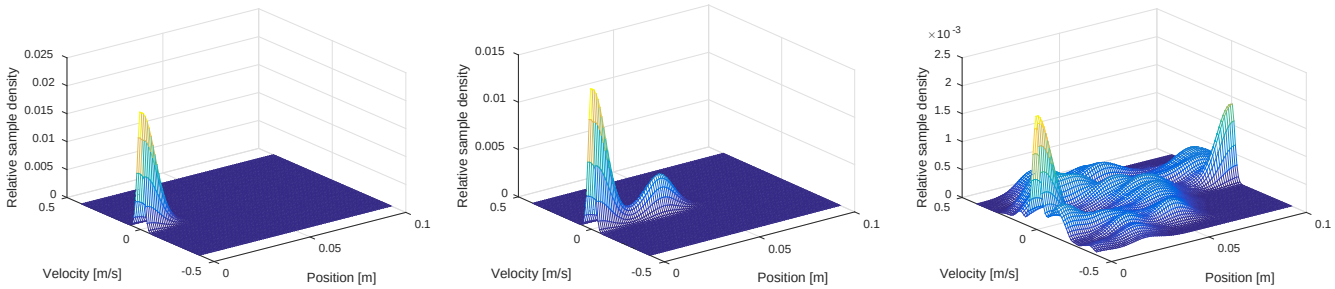
Fig. 3: Influence of the database distribution on the learning performance. Means and 90% confidence bounds shown for 30 trials.

A. Distribution Effects

In Section II-B.1 we theorized that the ideal distribution of the experiences in the mini-batches over the state-action space would be somewhere between a uniform distribution and the distribution resulting from the policy. We now test this hypothesis experimentally.

Trials are conducted with an experience replay database that is overwritten in the standard FIFO manner. However, with a probability α the experience that results from interacting with the system is replaced with a *hypothetical experience* before being written to the database. The hypothetical experience is synthesized by choosing the state and action uniformly random from the state and action spaces. The next state and the reward are known since a simulation is used. In general this is not the case, but here it serves to test the desirability of the theoretical database distribution.

The average results of 30 repetitions of this experiment are shown in Figure 3, for different values of α. For $\alpha = 0$ we get the standard FIFO method. Here, training is based on the experiences from the 10 most recent episodes. The



(a) FIFO overwriting (b) TDE overwriting (c) Distribution overwriting

Fig. 4: Experience distributions over the state-space after 2000 episodes for different database overwrite policies. The reinforcement learning performance when training with databases like these can be seen in Figure 5.

policy being followed is the current trained policy plus exploration. It can be seen that from episode 1500 the decaying exploration causes the diversity of the experiences in the database to drop to a point where the performance of the method, even on the training task, starts to decrease.

For $\alpha = 1$ the method uses only hypothetical experiences that are sampled uniformly from the state-action space. It can be seen from Figure 3b that this results in a somewhat better performance on the generalization task, but in a significantly worse performance on the training task. This makes sense as for the training task we would like the accuracy of the Q function approximation and the policy to be greatest in those regions of the state action space that are actually visited by the policy.

For $\alpha = 0.5$ the distribution of the experiences that the networks are trained with is an even mix of those from the policy with exploration and the uniform distribution. It can be seen from Figure 3b that this improves the generalization performance, while Figure 3a shows that it does not compromise the performance on the training task. In fact, when the exploration decays, the uniform samples help prevent the loss of performance observed for $\alpha = 0$. The eventual performance on both the training and the generalization task is significantly better when training on the mixed distribution than when training on either the policy distribution or the uniform distribution.

B. Overwriting Policy Effects

In general, the transition and reward functions are not available. Therefore simply adding the uniform hypothetical experiences to an existing database is not an option. To still get an approximation of a uniform distribution the database overwrite method of Section II-B.2 was proposed. This method takes the stream of experiences that result from interacting with the system and overwrites the database such that its contents are maximally spread out over the state-action space.

To test the effectiveness of this method, an experiment is performed in which the stream of experiences resulting from interacting with the system is fed to three separate databases. The first database simply overwrites the experiences in a FIFO manner. The second uses the temporal difference errors of (2) to decide which experiences to replace in the database. As suggested by [9] the experiences with the lowest temporal difference error are considered the least informative. Therefore these are replaced by new ones. The last database uses our overwrite policy based on the locations in the state-action space of the experiences already in the database.

In Figure 4 the distributions of the experiences in the three databases over the state space are shown after 2000 episodes. Note that the visualizations are only based on the state space and not on the action space as the full state-action space is too high dimensional to visualize. The density estimations are computed in a similar fashion to (7).

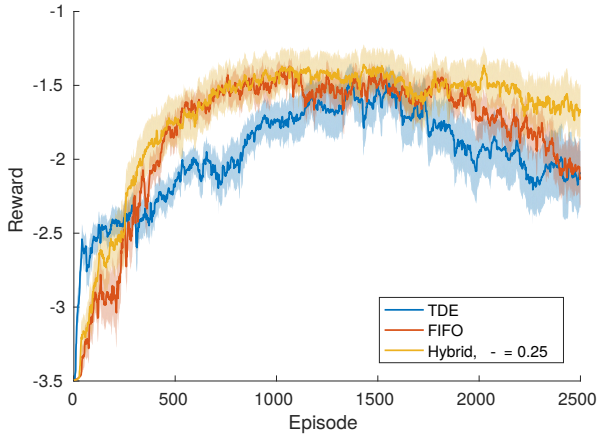
Figure 4a shows the danger of only remembering recent episodes. At this stage the exploration had decayed and the policy failed to reach the reference position. This meant that the FIFO database only contained samples very close to the initial position even though at an earlier stage much of the state space had been explored.

Figure 4b shows that by basing the retention on the temporal difference error, more diverse experiences are retained. However, also here much of the state space is not represented in the database. This is likely because at some point the temporal difference error at those locations was low, which results in the experiences being forgotten. When later the approximation of the Q function becomes worse in this region, the experiences can not be recovered. This issue is prevented in [9] by remembering all experiences and using the temporal difference error to decide how to sample from the database. This might however not be possible when high dimensional observations are used in combination with the lower dimensional states and actions, such as in [2], as it would require too much memory.

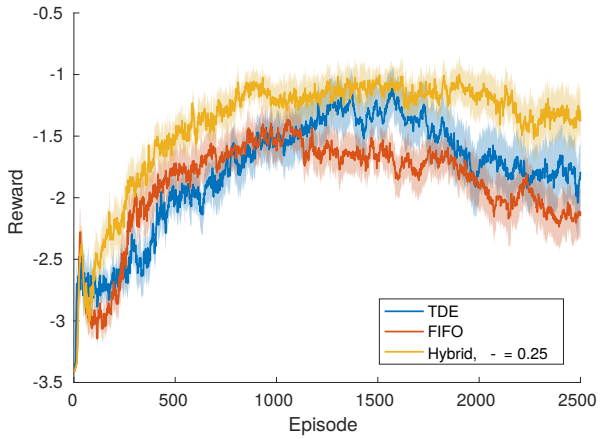
Our method is shown in Figure 4c. Although the distribution of the experiences over the state space is not uniform, the experiences cover a much larger fraction of the state space than either of the other methods.

C. Multiple Databases

In Section IV-A it was shown that mixing in experiences that are sampled uniformly from the state-action space with those resulting from interacting with the system was beneficial to the learning process. In Section IV-B we showed



(a) Performance on the training task



(b) Performance on the generalization task

Fig. 5: Performance of the proposed method compared to FIFO and TDE overwrite policies. Means and 90% confidence bounds shown for 30 trials.

that our proposed database overwrite policy resulted in a reasonable coverage of the state-space. Here it will be investigated if the beneficial effects of Section IV-A still hold when the experiences from the database $\mathcal{D}_U$ are used as an approximation of uniform experiences.

The method proposed in Section II-B.2 is compared to using the standard FIFO method and the TDE method in Figure 5. In these tests, the size of the FIFO and TDE databases is equal to the sum of the sizes of the two databases used in our method. It can be seen that for a reasonably chosen value of the mixing rate β, the method indeed works. Experiments were conducted for $\beta = 0.25, \beta = 0.5$ and $\beta = 0.75$. The best task and generalization performance was observed for $\beta = 0.25$. For this value of β, the performance on both the training and the generalization tasks is improved with respect to the FIFO method. When the amount of exploration decays, mixing in the older experiences from the rest of the state-action space helps limit the performance loss of the method. Interestingly the temporal difference based method performed worse than even the FIFO method in these experiments. This is likely due to the fact that only a limited number of experiences is kept in memory.

V. Conclusions

In this paper a method was introduced that uses two experience replay databases with different overwrite policies. One databases overwrites the experiences in its memory in a First In First Out manner. The second database overwrites the experiences in such a way as to ensure that the resulting distribution of the experiences over the state-action space is as close as possible to uniform. By sampling experiences from both databases, deep neural networks can be used more safely as function approximators for reinforcement learning. This is especially true for reinforcement learning on physical systems such as robots, as the method reduces the need for continued thorough exploration, allows for improved generalization performance, and allows the use of smaller experience replay databases.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," 2015, arXiv:1504.00702 [cs.LG].

[3] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016, arXiv:1602.01783 [cs.LG].

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7585, pp. 484–489, 2016.

[5] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[6] 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," in *International Conference on Learning Representations (ICLR)*, 2016.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, "The importance of experience replay database composition in deep reinforcement learning," 2015, deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS).

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, arXiv:1511.05952 [cs.LG].

[10] G. Montavon, G. B. Orr, and K.-R. Müller, Eds., *Neural Networks: Tricks of the Trade*, 2nd ed., ser. Lecture Notes in Computer Science (LNCS). Springer, 2012, vol. 7700.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning (ICML)*, 2014, pp. 387–395.

[12] I. Goodfellow, M. Mirza, X. Da, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgeting in gradient-based neural networks," 2013, arXiv:1312.6211 [stat.ML].

[13] E. Parzen, "On estimation of a probability density function and mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[14] G. Ciuti, P. Valdastri, A. Menciassi, and P. Dario, "Robotic magnetic steering and locomotion of capsule endoscope for diagnostic and surgical endoluminal procedures," *Robotica*, vol. 28, no. 02, pp. 199–207, 2010.