In: Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), pages: 139–144, Rio de Janeiro, Brazil. IEEE, July, 2018.

Accelerating Deep Continuous Reinforcement Learning through Task Simplification

Matthias Kerzel[1], Hadi Beik Mohammadi[1], Mohammad Ali Zamani[1], and Stefan Wermter[1]

***Abstract*— Robotic motor policies can, in theory, be learned via deep continuous reinforcement learning. In practice, however, collecting the enormous amount of required training samples in realistic time, surpasses the possibilities of many robotic platforms. To address this problem, we propose a novel method for accelerating the learning process by task simplification inspired by the Goldilocks effect known from developmental psychology. We present results on a reach-for-grasp task that is learned with the Deep Deterministic Policy Gradients (DDPG) algorithm. Task simplification is realized by initially training the system with "larger-than-life" training objects that adapt their reachability dynamically during training. We achieve a significant acceleration compared to the unaltered training setup. We describe modifications to the DDPG algorithm with regard to the replay buffer to prevent artifacts during the learning process from the simplified learning instances while maintaining the speed of learning. With this result, we contribute towards the realistic application of deep reinforcement learning on robotic platforms.**

I. INTRODUCTION

Deep reinforcement learning can be used to learn discrete or continuous motor policies based on interaction with the environment. Therefore, this class of learning algorithms is a perfect fit for research in developmental robotics [1], as it enables an embodied neuro-cognitive system to learn complex sensorimotor abilities through trial and error. These approaches have been successfully applied in virtual environments [2]. However, it is challenging to adapt these results to real-world applications, as deep reinforcement learning relies on a vast amount of training samples. Considering the number of operation hours [3], [4], which an agent needs to learn a policy, using an actual robot is often not practicable. Therefore, the scientific challenge arises to accelerate the learning process, with regard to the number of samples that are necessary to learn a motor policy (sample efficiency). To this end, we present a novel approach that follows the developmental robotics paradigm, where more and more sophisticated cognitive and sensorimotor abilities are developed through interacting with increasingly challenging environments [1].

We focus on a reach-for-grasp task, one of the most general and fundamental tasks in robotics, which is in the focus of research [2]–[6]. In our approach, a simulated robot arm learns to reach for an object, as shown in Figure 1. While traditional approaches for grasping use separate models for vision, grasp planning and inverse kinematics [5], neural learning paradigms offer the possibility to learn all of these subtasks in an end-to-end approach without even knowing the robot model [7].

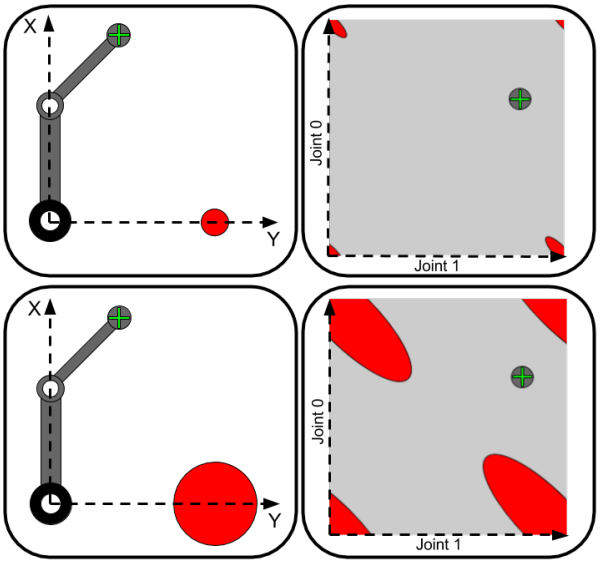

Fig. 1. Task simplification: The joint space shows possible solutions (red) for grasping the red target, which is represented twice in this space due to redundant solutions. Reaching a small target requires a very precise joint configuration that is difficult to find by random exploration. By simplifying the task (lower row) through larger training targets, the requirements for the joint configuration are relaxed; this is represented by larger red areas. This can accelerate the initial phase of deep reinforcement learning.

To address the problem of sample efficiency, the task is simplified by presenting the system with "larger-than-life" training targets which are easier to reach, as depicted in Figure 1. Motor policies that develop from this simplified task instance are then used to learn to reach for increasingly difficult, e.g., smaller, targets.

We evaluate the proposed learning strategy using the well-established continuous deep reinforcement learning algorithm DDPG (Deep Deterministic Policy Gradients) [2]. Results show a 50% average acceleration for the overall learning time by dynamically adapting the reachability of the target according to the learning progress. We also document necessary modifications to the algorithm with regards to how episodes are stored in the replay buffer and utilized for updating the neural networks of DDPG to prevent learning suboptimal policies from simplified learning instances.

*The authors gratefully acknowledge partial support from the German Research Foundation DFG under project CML (TRR 169), the European Union under project SECURE (No 642667), and the Hamburg Landesforschungsförderungsprojekt CROSS.

[1]Knowledge Technology, Department of Informatics, University of Hamburg, Germany. kerzel / 6beik / zamani / wermter @informatik.uni-hamburg.de

II. BACKGROUND

A. Continuous Deep Reinforcement Learning

First, we describe deep reinforcement learning with DDPG, to motivate both the idea of task simplification and the necessary modifications to DDPG. Though task simplification can be applied to most variants of (continuous) reinforcement learning algorithms like the continuous actor-critic learning algorithm (CACLA) [8] and continuous deep q-learning with model-based acceleration [9] we focus on the widely-used and well-established DDPG algorithm.

Q-learning is based on trial-and-error learning by an agent in an interactive environment. The agent learns the mapping from a discrete set of states s_t and actions a_t to an expected reward (q-value), which allows selecting the best action in a given state. Mnih et al. [10] introduced Deep Q-Network learning, which uses complex, high dimensional input data, e.g., unprocessed pixel data, but it is still limited to discrete actions. To address this challenge, DDPG was introduced by Lilicrap et al. [2] to extend deep reinforcement learning to continuous actions.

Deep Q-Networks are based on neural function approximates for q-values, i.e., the neural network is updated with the Bellman equation, where the sum of direct rewards and expected (discounted) future rewards from the environment are learned for each action-state combination. In deep Q-networks, an action is selected by letting a neural network approximate the expected reward in a given state for a fixed set of actions. The action with the highest expected reward is then chosen.

For a continuous control problem, like robot grasping, even a coarse discretization of actions would lead to an exponentially high number of actions with increasing degrees of freedom. DDPG solves this challenge by training a separate parameterized actor function realized as a neural network. It follows the actor-critic architecture for deterministic policy gradient algorithms [11].

The critic and actor-networks are not trained directly with a sequence of performed actions, but performed actions are committed to the replay memory (replay buffer) and drawn randomly for updates. This decorrelates the samples used for a single update to increase stability. The actor network is updated by gradient transfer from the critic network with respect to the actor parameters. Given a state, the actor network outputs the action with the highest expected reward.

To increase stability, "soft" target updates are used for both the critic and actor networks by creating copies of the respective networks that are only slowly adjusted towards the learned networks. This further slows down learning, requiring hundreds of thousands of samples for a policy [2].

B. Speeding up Deep Reinforcement Learning

Various approaches have been suggested to accelerate deep reinforcement learning. They modify different aspects of the learning procedure; here we give an exemplary overview:

Mnih et al. [10] suggest utilizing an optimal selection of samples from the replay memory to update the neural network. They propose to use prioritized sweeping [12] to select those samples that have the best effect on learning. This was successfully realized by Schaul et al. [13].

Other approaches focus on the exploitation behavior: Hafez et al. [14] use a curiosity-driven approach in a reach-for-grasp task to reward the exploration of yet unknown parts of the state space to speed up learning time as a repeated exploration of known parts of the action space is prevented.

Otte et al. [15] employ a curriculum training method for a neural network realization of inverse kinematics. The training is split into several episodes, with incrementally relaxed joint limits, thus increasing the size of the action space gradually. This can be seen as a form of task simplification by constraining the possible actions. None of the summarized approaches, however, alter the learning task itself.

A different approach to increase sample efficiency is to transform the reinforcement learning task into supervised learning. Levine et al. [16] and Kerzel and Wermter [7] utilize fully annotated training samples of successful trials to train a neural architecture, thus fully avoiding the trial-and-error learning phase. Levine et al. compute the forward kinematics of a robot that is generating samples, while Kerzel and Wermter exploit the fact that the action of grasping an object can be inverted into the much simpler task of placing an object at a random position. Both approaches rely on having additional information available to the system (forward kinematics) or being able to devise a suitable learning scenario (task inversion) - both options are not always available.

C. Developmental Background

According to Libertus et al. [17] there is biological evidence that infants who are less experienced in grasping show a preference to visually and manually explore larger objects. Once the reach-for-grasp ability of the infants improves, it shifts towards smaller objects. Though Newman at al. [18] attribute this shift in preference to a development in visual processing streams that enable object-related actions, it can also be interpreted as an instance of the Goldilocks effect [19], that describes the tendency of infants to prefer stimuli that have *just the right* complexity to enable learning novel abilities without being overburdened.

To address the research question if and how deep continuous reinforcement learning can also benefit from the Goldilocks effect, i.e., learner appropriate task difficulty, we develop an experimental reach-for-grasp setup and evaluate the effect of task simplification.

III. METHODOLOGY: TASK SIMPLIFICATION BY REACHABILITY ADAPTATION

To enable DDPG to learn from task-simplification, we introduce two changes to the algorithm: first, we reorganize learning into learning *stories*. Stories are learning units, where the target has a fixed position but dynamically adapts its size and therefore reachability according to the learning success. Secondly, for each story, we construct a local memory that is separated from the main replay memory to

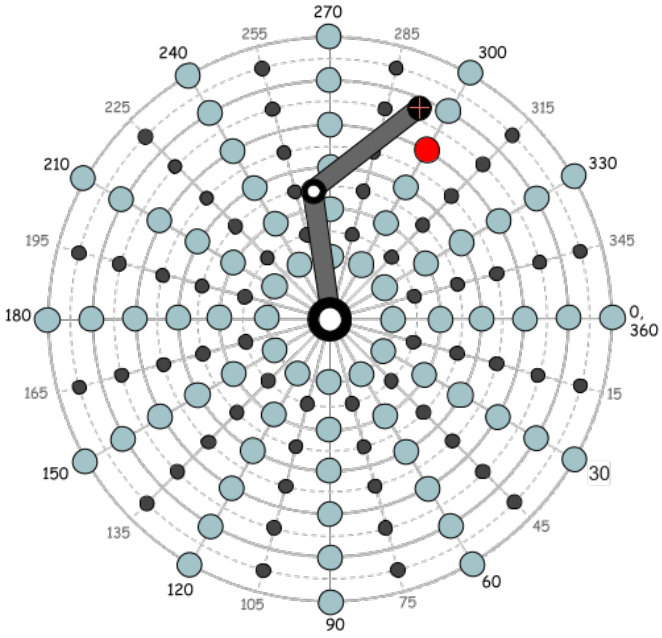

Fig. 2. Simulated environment with train and test targets. Blue circles indicate the training targets and black circles indicate the test targets.

prevent the algorithm from learning artifacts from simplified task instances. Our experimental setup is a two-dimensional reach-for-grasp task with a simulated robot arm with two degrees of freedom depicted in Figure 2.

A. Training Phases

Learning is organized into learning *stories*. During each *story*, the agent will try to repeatedly reach a target at the same position until it achieves a reliable success rate. A *story* consists of episodes. At the beginning of each episode, the joint configuration is randomly initialized before the learning steps begin. Finally, during each step, when the actor network is fed the current joint configuration and target location, it determines an action (modification to the joint values) with the highest expected reward. This action is then executed in the simulator and a reward is assessed: if the gripper reaches the target a positive reward of 10 is given, and the episode terminates. The episode also terminates if the target was not reached within 800 steps ($MaxStepNumber = 800$). In every other step a small negative reward of -0.001 is given. The expected reward is calculated by adding the reward for the current step to the discounted future reward according to the Bellman equation with a discount factor γ of 0.99.

A story is finished when the agent manages to reach the non-simplified target reliably. In our experimental setup, this is defined as reaching the target 20 consecutive times (found empirically). Figure 3 shows the three nested loops of our modified DDPG algorithm: the outer loop controls the position of the target. The next loop, called episode loop, controls the size of the target. This loop is repeated depending on learning success; it also performs the fine tune procedure. Finally, the core loop controls the configuration of the robot during single steps.

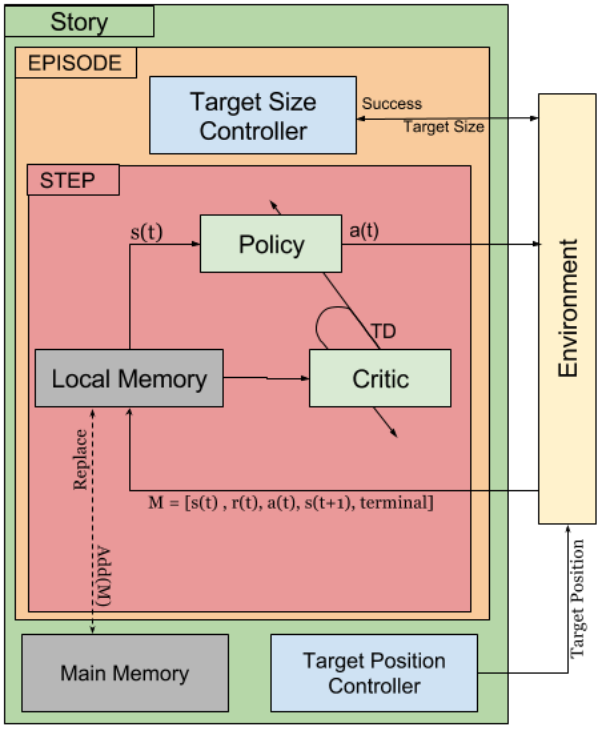

Fig. 3. Overview of our modified DDPG algorithm.

Fig. 4. Exemplary target size adaptation: during the 20 first steps, the size is not changed as the current reaching ability of the learner is evaluated. Usually the target is not reliably reached which results in the target growing to its upper size limit. Once the target is reliably reached, it begins to shrink gradually. Finally, when the agent reliably reaches the target, fine tuning on the non-simplified target prevents learning artifacts from simplified instances.

B. Task-simplification through Adaptive Reachability

To adapt to the learner's progress, the size of the target is dynamically adjusted according to the learning success. In the initial learning phase the target will gain an increased size and is, therefore, easier to find by random exploration. Once the target is reached, its size, and therefore its reachability, is dynamically adjusted by the *target size controller* according to Figure 4. Different schemes for adapting the target size were tested and the best results were gained when the starting size of the target is non-simplified to see if the actor can already reach the target reliably from past learning stories. If not, the target size is increased quickly. Once the target reached reliably the size is again decreased at a slower pace. The *target size controller* monitors if episodes are terminated successfully by reaching the target or unsuccessfully after 800 steps of not reaching the target. After each episode, the

target size is adjusted in the following way:

$$MAS(\beta) = \frac{1}{\beta} \sum_{E'_N = E_N - \beta}^{E_N} S_{E'_N} \quad S_{E_N} = \begin{cases} 1 & if\ success \\ 0 & if\ fail \end{cases} \tag{1}$$

$$T_S = \begin{cases} S_{min} & E_N < E_{thr} \\ min(T_S + \delta^+, S_{max}) & MAS(20) < P_{thr}\ \&\ E_N \geq E_{thr} \\ max(T_S - \delta^-, S_{min}) & MAS(20) \geq P_{thr}\ \&\ E_N \geq E_{thr} \end{cases} \tag{2}$$

MAS represents the sum of moving average over the success queue. Target size (T_S) is defined by three different equations for each stage of the algorithm: initialization, growing, and shrinking. These stages are formalized by three conditions (equation 2) which indicate how the target size has been modified (Figure 4). P_{thr} is the performance threshold of 0.9 and E_{thr} is the episode threshold of 20 (empirically optimized). δ^- and δ^+ are the rate of shrink and expand, respectively. Both size operations are limited to the minimum (S_{min}) and maximum (S_{max}) size of the target. For initialization and fine-tuning, target minimum size S_{min} is assigned. This ensures that the size of the target and therefore the difficulty of the task adjusts to the learning success of the agent in a stable way.

C. Consolidation Phase and Local Memory

A necessary modification to DDPG is the introduction of a *local memory* in addition to the usual replay memory, renamed *main memory*. The *local memory* is used to separate experiences from main memory to ensure that task-simplification does not lead to learning artifacts. At the beginning of a *story* a new local memory is initialized by creating a copy of the main memory. After each step, the transition *(state(t), action(t), reward(t), state(t+1), terminal)* is stored as a 5-tuple in the *local memory*. If the episode took place while the target had already reached its minimum size, then the transitions of all steps within that episode are also committed to the main memory. This separation ensures that the *local memory* can take advantage of the simplified task instances while the main memory does not contain artifacts from these instances.

During an update, samples from the local memory are used; this leverages the acceleration gained from task simplification, but prevents catastrophic forgetting [20], as each local memory is initialized as a copy of the main memory.

Once a story ends because the target has been reliably reached during the last 20 episodes, the learning is consolidated in a *fine tuning phase*, for another 50 episodes ($Success^{max}_{story} = 50$), the learning is continued to learn transitions of the non-simplified task instance.

Algorithm 1 DDPG Algorithm [2] with task simplification

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ with Glorot uniform
Initialize target network Q' and μ': $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize main and local replay buffer R_M, R_L
for $story = 0 : TargetPoints$ **do**
  Assign a new position to target
  Replace the local memory by the main memory
  Initialize $FineTune_n = 0$, Episode number $E_N = 0$, $Success_{story} = 0$, $FineTunePhase = False$
  while $FineTune_n < FineTune_{max}$ **do**
    if $E_N < E_{thr}$ OR $FineTunePhase$ **then**
      $T_s = S_{min}$
    else if $E_N > E_{thr}$ **then**
      if $MAS(20) < P_{thr}$ **then**
        T_S = $min(T_S$ + δ^+, $S_{max})$
      else if $MAS(20) \geq P_{thr}$ **then**
        T_S = $max(T_S$ - δ^-, $S_{min})$
      end if
    end if
    Initialize a random process N for action exploration
    Receive initial observation state s_0
    for t = 0, $MaxStepNumber$ **do**
      Select action $a_t = \mu(s_t|\theta^\mu) + N_t$ according to the current policy and exploration noise
      Execute action a_t and observe reward r_t and observe new state s_{t+1}
      if $T_S > S_{min}$ **then**
        $R_L \overset{Store}{\leftarrow} (s_t, a_t, r_t, s_{t+1}, ter_t)$
      else
        $R_L, R_M \overset{Store}{\leftarrow} (s_t, a_t, r_t, s_{t+1}, ter_t)$
      end if
      Sample a random mini-batch from R_L
      Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update critic: $\min_{\theta^Q} \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
      Update actor network:
      $\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|^{a=\mu(s_i)}_{s=s_i} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$
      Update the target networks:
      $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
      $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
    end for
    if $T_S == S_{min}$ **and** $S_{E_N} == 1$ **then**
      $Success_{Story} += 1$
      if $Success_{story} \geq Success^{max}_{story}$ **then**
        $FineTunePhase = True$
      end if
    end if
    if $FineTunePhase$ **then**
      $FineTune_n += 1$
    end if
  end while
end for

IV. IMPLEMENTATION AND RESULTS

We evaluated the effect of task simplification by dynamically adapting the target size in the above-described task.

A. Experimental Setup

We have developed a fully graphical simulator for this purpose where the control frequency is almost 1 kHz. Though this setup cannot be compared to real robotic platforms, it enables us to evaluate the (sample) efficiency of different learning setups.

Similar to Lillicrap et al. [2] we excluded vision processing and directly fed target position and joint angles into the DDPG network that outputs changes to the joint configuration. The rationale behind this is that joint angles are known to a learning robotic system, and the visual localization of the target can happen via pre-trained components.

B. Implementation

Our modified DDPG algorithm was implemented in Keras [21] with Tensorflow backend [22]. Critic-network and actor-network have 400 hidden units each. The input is defined as a low-level feature vector regarding the position of the target in polar space and the current joint configuration of the arm in joint space.

$$Input(t) = [sin(\theta_0), cos(\theta_0), sin(\theta_1), cos(\theta_1), sin(\theta_{target}), cos(\theta_{target}), radius_{target}] \quad (3)$$

The Adam optimizer has been used to train new policies with a learning rate of 10^{-4} for the actor and 2×10^{-4} for the critic. The activation function for all layers was Rectified Linear Unit (ReLU). The activation function of the last layer of the actor-network which produces the joint angles was a hyperbolic tangent function ($\tanh$). The neural network architecture and hyper-parameters were optimized empirically; smaller or more shallow networks had problems learning larger numbers of targets.

The actor-network produces one value between -1 and 1 per joint; at each time step each joint moves at most 1 degree in either direction. In case bigger steps are desired, the output of the actor can be scaled up by a proportional factor.

C. Evaluation

In both the static *baseline* and the dynamic *task simplification* condition the algorithm was trained with 72 stories. To ensure comparability between conditions, the positioning of targets both for training and evaluation followed the fixed scheme depicted in Figure 2 and used the same network architecture and hyper-parameters. Each learning target was presented once in random order. Both conditions have been repeated six times to avoid biases through random initialization of the neural networks. Figure 5 shows the success rate during training for known targets and Figure 6 for unseen test targets to demonstrate how the network generalizes. Results indicate that adapting the target size to the learning success leads to a better reach-for-grasp accuracy.

More importantly, adapting the target size to the learning progress significantly shortened the overall number of training steps. Figure 7 shows the accumulated number of learning steps in each story. In the dynamic *task simplification* condition, the number of steps accumulates slower than in the static baseline condition, which means the agent has learned to generalize well to new targets.

D. Discussion

These results can be interpreted in the following way: as any reinforcement learning algorithm, DDPG relies on gaining reward through successful actions. Only these rewards

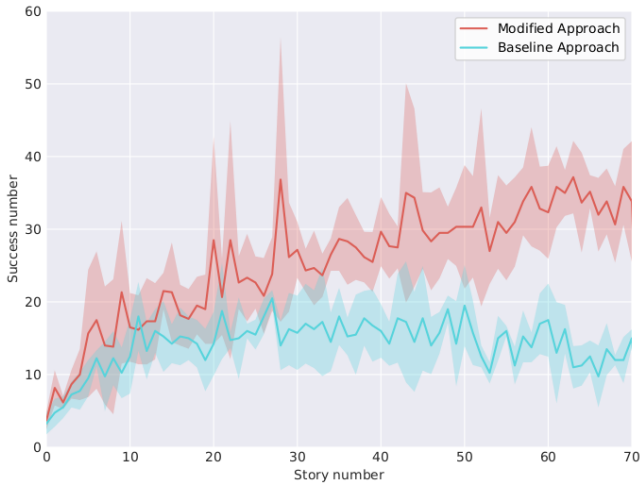

Fig. 5. The average number of successes to reach the *training* targets. The vertical axis indicates the average success number and the horizontal axis indicates the story number.

Fig. 6. The average number of successes to reach the *test* targets. The vertical axis indicates the average success number and the horizontal axis indicates the story number.

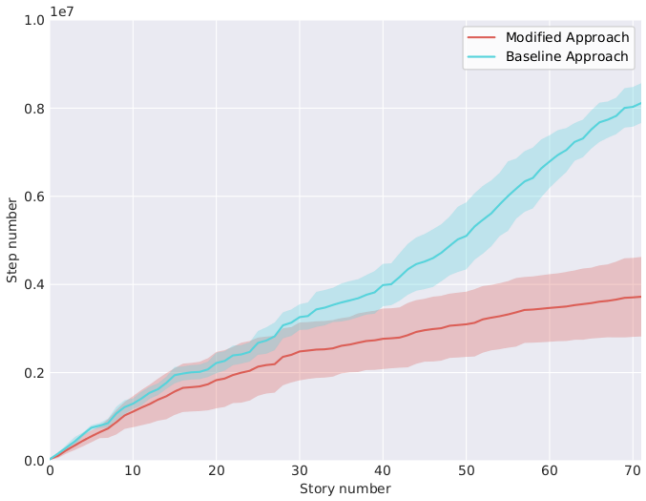

Fig. 7. The cumulative number of learning steps. Our modified DDPG algorithm learns about twice as fast as the baseline algorithm.

train the critic and indirectly the actor-network. The updated actor then directs the exploitation towards more rewarding areas of the state space while a random term ($[-0.3, 0.3]$ linearly scaling down to $[-0.05, 0.05]$ over 300 epochs) added to the output of the actor ensures ongoing exploration, e.g., finding shorter paths to the target.

However, the networks for the critic and the actor are randomly initialized. Therefore, the actor generates random actions, and the target can only be found by a low chance. If the range of possible joint configurations that do reach the target are marginal compared to the configurations that do not, it becomes unlikely to reach the target; extending the initial phase unnecessarily. Providing a larger target, on the other hand, simplifies the task and increases the likelihood to gain initial rewards quickly, thus accelerating learning.

The observed acceleration can also be explained by analyzing what transitions are stored in the main memory. During the long, initial random exploration at the beginning of new stories, transitions with little learning value are committed to the memory. This is shown in Figure 5, where forgetting of already learned policies [20] can be observed.

In the dynamic condition, however, transitions are only committed to the main memory when the task has reached its non-simplified difficulty; this, in turn, happens when the learner reliably reaches the target. Therefore, the dynamic condition realizes a hard prioritization of samples with a high learning value which could also be interpreted as a pushing the learning scenario towards supervised learning [7], [16].

V. CONCLUSION

We have presented a novel approach to accelerate deep continuous reinforcement learning with DDPG in terms of sample efficiency by utilizing task-simplification inspired by the Goldilocks effect from developmental psychology. We introduced modifications to DDPG to prevent overfitting to simplified task instances. Results show that adapting the difficulty of a task according to the learner's progress, e.g., by presenting an easier target in a reach-for-grasp task, accelerates the learning compared to a baseline condition and furthermore creates a more stable learning result. For a 2D reach-for-grasp task with a two-jointed arm, we achieve a 50% percent average acceleration compared to the DDPG baseline algorithm. We assume that this relative increase will also apply to more complex tasks like controlling a robot arm with six degrees of freedom.

We contribute a novel technique to accelerate reinforcement learning by modifying the learning task itself. This method and the modifications to the learning algorithm can be adapted to related approaches and also be combined with other methods to accelerate the learning. To what extent this is successful will be evaluated in future work. Likewise, the approach can be generalized to other typical reinforcement learning tasks by providing simplified problem instances to accelerate initial learning like a more stable pole to swing up, a shorter path to find or a more straight road to follow. As this contribution is aimed to enable deep reinforcement learning on real robotic systems, in future work, we will employ our approach on a humanoid robot.

REFERENCES

[1] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Cambridge, MA: MIT Press, 2015.

[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. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, p. 0278364917710318, 2016.

[4] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3406–3413.

[5] J. Leitner, S. Harding, A. Förster, and P. Corke, "a modular software framework for eye–hand coordination in humanoid robots," *Frontiers in Robotics and AI*, vol. 3, p. 26, 2016.

[6] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[7] M. Kerzel and S. Wermter, "Neural end-to-end self-learning of visuomotor skills by environment interaction," in *International Conference on Artificial Neural Networks (ICANN)*, 2017 accepted.

[8] H. Van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*. IEEE, 2007, pp. 272–279.

[9] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.

[10] 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.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 387–395.

[12] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.

[13] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[14] B. Hafez, C. Weber, and S. Wermter, "Curiosity-driven exploration enhances motor skills of continuous actor-critic learner," in *Proceedings of the 7th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL-EpiRob)*, Sep 2017, pp. 39–46.

[15] S. Otte, A. Zwiener, R. Hanten, and A. Zell, "Inverse recurrent models–an application scenario for many-joint robot arm control," in *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 149–157.

[16] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," 2015.

[17] K. Libertus, J. Gibson, N. Z. Hidayatallah, J. Hirtle, R. A. Adcock, and A. Needham, "Size matters: How age and reaching experiences shape infants preferences for different sized objects," *Infant Behavior and Development*, vol. 36, no. 2, pp. 189–198, 2013.

[18] C. Newman, J. Atkinson, and O. Braddick, "The development of reaching and looking preferences in infants to objects of different sizes." *Developmental Psychology*, vol. 37, no. 4, p. 561, 2001.

[19] C. Kidd, S. T. Piantadosi, and R. N. Aslin, "The goldilocks effect in infant auditory attention," *Child development*, vol. 85, no. 5, pp. 1795–1804, 2014.

[20] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.

[21] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.

[22] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.