Bootstrapping a DQN Replay Memory with Synthetic Experiences

Wenzel Baron Pilar Von Pilchau[1][a], Anthony Stein[2] and Jörg Hähner[1]
[1]*Organic Computing Group, University of Augsburg, Eichleitnerstr. 30, Augsburg, Germany*
[2]*Artificial Intelligence in Agricultural Engineering, University of Hohenheim, Garbenstraße 9, Hohnheim, Germany*

Keywords: Experience Replay, Deep Q-Network, Deep Reinforcement Learning, Interpolation, Machine Learning.

Abstract: An important component of many Deep Reinforcement Learning algorithms is the Experience Replay that serves as a storage mechanism or memory of experienced transitions. These experiences are used for training and help the agent to stably find the perfect trajectory through the problem space. The classic Experience Replay however makes only use of the experiences it actually made, but the stored transitions bear great potential in form of knowledge about the problem that can be extracted. The gathered knowledge contains state-transitions and received rewards that can be utilized to approximate a model of the environment. We present an algorithm that creates synthetic experiences in a nondeterministic discrete environment to assist the learner with augmented training data. The Interpolated Experience Replay is evaluated on the FrozenLake environment and we show that it can achieve a 17% increased mean reward compared to the classic version.

1 INTRODUCTION

The concept known as Experience Replay (ER) started as an extension to Q-Learning and AHC-Learning (Lin, 1992) and developed to a norm in many Deep Reinforcement Learning (RL) algorithms (Schaul et al., 2015; Mnih et al., 2015; Andrychowicz et al., 2017). One major advantage is its ability to increase sample efficiency. Another important aspect is, that algorithms like Deep Q-Network (DQN) are even not able to learn in a stable manner without this extension (Tsitsiklis and Van Roy, 1997). This effect is caused by correlations in the observation sequence and the fact that small updates may significantly change the policy and in turn alternate the distribution of the data. By uniformly sampling over the stored transitions, ER is able to remove these correlations as well as smoothing over changes in the data distribution (Mnih et al., 2015).

Most versions of ER store the real, actually made, experiences. For instance the authors of (Mnih et al., 2015) used vanilla ER for their combination with DQN, and also (Schaul et al., 2015) who extended vanilla ER to their Prioritized Experience Replay, that is able to favour experiences from which the learner can benefit most. But there are also approaches that are filling their replay memory with some kind of synthetic experiences to support the learning process. One example is the Hindsight Experience Replay from (Andrychowicz et al., 2017) that takes a trajectory of states and actions aligned with a goal and replaces the goal with the last state of the trajectory to create a synthetic experience. Both, the actual experienced trajectory, as well as the synthetic one are then stored in the ER. This approach helps the learner to understand how it is able to reach different goals. This approach was implemented in a multi-objective problem space and after reaching some *synthetic* goals the agent is able to learn how to reach the intended one.

Our contribution is an algorithm that is targeted to improve (Deep) RL algorithms that make use of an ER, like e.g. DQN, DDPG or classic Q-Learning (Zhang and Sutton, 2017), in nondeterministic and discrete environments by means of creating synthetic experiences utilizing stored real transitions. We can increase sample efficiency as transitions are further used to generate more and even better experiences. The algorithm therefore computes an average value of the received rewards in a situation and combines this value with observed follow-up states to create so called *interpolated experiences* that assists the learner in its exploration phase.

The evaluation is performed on the *FrozenLake* environment from the OpenAI Gym (Brockman et al., 2016).

This approach investigates only discrete and non-deterministic environments and the averaging is a rather simple method as well, but the intention is

[a] https://orcid.org/0000-0001-9307-855X

von Pilchau, W., Stein, A. and Hähner, J.
Bootstrapping a DQN Replay Memory with Synthetic Experiences.
DOI: 10.5220/0010107904040411
In *Proceedings of the 12th International Joint Conference on Computational Intelligence (IJCCI 2020)*, pages 404-411
ISBN: 978-989-758-475-6

to gain first insights in this highly interesting field. We can reveal promising potentials utilizing this very simple technique and this work serves as a basis to build up further research on.

The paper is structured as follows: We start with a brief introduction of the ER and Deep Q-Learning in section 2 and follow up with some related work in section 3. In section 4 we introduce our algorithm along with a problem description and the Interpolation Component that was used as an underlying architecture. The evaluation and corresponding discussion as well as interpretation of the results is presented in section 5. Section 6 is the conclusion and presents ideas for future work.

2 BACKGROUND

In this section, we introduce some background knowledge.

2.1 Experience Replay

The ER is a biological inspired mechanism (McClelland et al., 1995; O'Neill et al., 2010; Lin, 1992; Lin, 1993) to store experiences and reuse them for training later on.

An experience is defined as: $e_t = (s_t, a_t, r_t, s_{t+1})$ where a_t denotes the start state, a_t the performed action, r_t the corresponding received reward and s_{t+1} the following state. To perform experience replay, at each time step t the agent stores its recent experience in a data set $D_t = \{e_1, \ldots, e_t\}$.

This procedure is repeated over many episodes, where the end of an episode is defined by a terminal state. The stored transitions can then be utilized for training either online or in a specific training phase. It is very easy to implement ER in its basic form and the cost of using it is mainly determined by the storage space needed.

2.2 Deep Q-Learning

The DQN algorithm is the combination of the classic Q-Learning (Sutton and Barto, 2018) with neural networks and was introduced in (Mnih et al., 2015). The authors showed that their algorithm is able to play Atari 2600 games on a professional human level utilizing the same architecture, algorithm and hyperparameters for every single game. As DQN is a derivative of classical Q-Learning it approximates the optimal action-value function:

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi\right] \tag{1}$$

However DQN employs a neural network instead of a table. Equation (1) displays the maximum sum of rewards r_t discounted by γ at each time-step t, that is achievable by a behaviour policy $\pi = P(a|s)$, after making an observation s and taking an action a. DQN performs an Q-Learning update at every time step that uses the temporal-difference error defined as follows:

$$\delta_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \tag{2}$$

(Tsitsiklis and Van Roy, 1997) showed that a nonlinear function approximator used in combination with temporal-difference learning, such as Q-Learning, can lead to unstable learning or even divergence of the Q-Function.

As a neural network is a nonlinear function approximator, there arise several problems:
1. the correlations present in the sequence of observations,
2. the fact that small updates to Q may significantly change the policy and therefore impact the data distribution, and
3. the correlations between the action-values $Q(s_t, a_t)$ and the target values $r + \gamma \max_{a'} Q(s_{t+1}, a')$ present in the td-error shown in (2).

The last point is crucial, because an update to Q will change the values of both, the action-values as well as the target values, that could lead to oscillations or even divergence of the policy. To counteract these issues, two concrete actions have been proposed:
1. The use of an ER solves, as stated above, the two first points. Training is performed each step on minibatches of experiences $(s, a, r, s') \sim U(D)$, that are drawn uniformly at random from the ER.
2. To remove the correlations between the action-values and the target values a second neural network is introduced that is basically a copy of the network used to predict the action-values, but it is freezed for a certain interval C before it is updated again. This network is called target network and is used for the computation of the target action-values. (Mnih et al., 2015)

We use the target network as it was presented above and extend the classic ER with a component to create synthetic experiences.

3 RELATED WORK

The classical ER, introduced in section 2.1, has been improved in many further publications. One prominent improvement is the so called Prioritized Experience Replay (Schaul et al., 2015) that replaces the uniform sampling with a weighted sampling in favour of experience samples that might influence the learning process most. This modification of the distribution in the replay induces bias and to account for this, importance-sampling has to be used. The authors show that a prioritized sampling leads to great success. This extension of the ER also changes the default distribution, but uses real transitions and therefore has a different focus.

Another publication (De Bruin et al., 2015) investigates the composition of experience samples in the ER. They discovered that for some tasks it is important, that transitions, made in an early phase, when exploration is high, are important to prevent overfitting. Therefore they split the ER in two parts, one with samples from the beginning and one with actual samples. They also show that the composition of the data in an ER is vital for the stability of the learning process and at all times diverse samples should be included. Following this results we try to achieve a broad distribution over the state space utilizing synthetic experiences.

(Stein et al., 2017; Stein et al., 2018) use interpolation in combination with XCS Classifier System to speed up learning in single-step problems by means of using previous experiences as sampling points for interpolation. The used component for interpolation is part of this work and discussed in more detail in section 4.3.

4 INTERPOLATED EXPERIENCE REPLAY

In this Section we present the FrozenLake problem and introduce our algorithm to solve it. We also introduce the Interpolation Component (IC) that serves as architectural concept.

4.1 Problem Description

"FrozenLake" is one example of a nondeterministic world in which an action $a_t \in A$ realised in a state $s_t \in S$ may not lead consistently to the same following state $s_{t+1} \in S$. FrozenLake is basically a grid world consisting of an initial state I, a final state G and frozen, as well as unfrozen tiles. The unfrozen tiles equal holes H in the lake and if the agent falls into one of such, he receives a reward of -1 and has to start from the initial state again. If the agent reaches G he receives a reward of 1. The set of possible actions A consists of the four cardinal directions $A = \{N, E, S, W\}$. The environment is nondeterministic, because the agent might slide on the frozen tiles which is implemented through a certain chance of executing a different action instead of the intended one. The environment is discrete, because there is a discrete number of states the agent can reach. The environment used for evaluation is the "FrozenLake8x8-v0" environment from OpenAI Gym (Brockman et al., 2016).

If an action is chosen that leads the agent in the direction of the goal, but because of the slippery factor it is falling into a hole, it also receives a negative reward and creates the following experience: $e_t = (s_t, a_t, -1, s_{t+1})$. If this experience is used for a Q update it misleadingly shifts the state-action value away from a positive value. We denote the slippery factor for executing a neighbouring action as c_{slip}, the resulting rewards for executing the two neighbouring actions as r_t^{right} and r_t^{left} and the reward for executing the intended action as r_t^{int} and can then define the true expected reward for executing a_t in s_t as follows:

$$r_t^{\text{exp}} = \frac{c_{\text{slip}}}{2} \cdot r_t^{\text{right}} + \frac{c_{\text{slip}}}{2} \cdot r_t^{\text{left}} + (1 - c_{\text{slip}}) \cdot r_t^{\text{int}} \quad (3)$$

Following (3) we can define the experience that takes the state-transition function into account and that not confuses the learner as follows:

$$e_t^{\text{exp}} = (s_t, a_t, r_t^{\text{exp}}, s_{t+1}) \quad (4)$$

The learner will converge its state-action value $Q_\pi(s_t, a_t)$ after seeing a lot of experiences to:

$$Q_\pi(s_t, a_t) = Q^*(s_t, a_t) = r_t^{\text{exp}} + \gamma \max_{a'} Q^*(s_{t+1}, a') \quad (5)$$

We define the set of all rewards that belong to the experiences that start in the same state s_t and execute the same action a_t as:

$$R_t := \left\{ r_n \in \{ r | (s, a, r, s') \in D_t \land a = a_t \land s = s_t \} \right\} \quad (6)$$

In our work we utilize stored transitions from the replay memory to create synthetic experiences with an averaged reward r_t^{avg} that is as close as possible to r_t^{exp}. Following (6) we can define these *interpolated* experiences as:

$$r_t^{\text{avg}} = \frac{\sum_{r \in R_t} r}{|R_t|} \quad (7)$$

$$e_t^{\text{avg}} = (s_t, a_t, r_t^{\text{avg}}, s_{t+1}) \quad (8)$$

with

$$e_t^{\text{avg}} \approx e_t^{\text{exp}} \quad (9)$$

The accuracy of this interpolation correlates with the amount of transitions stored in the ER, starting in s_t and executing a_t. As a current limitation so far, because we need a legal follow-up state s_{t+1}, it is crucial for the environment to be discrete (in a continuous world we would have infinite states s_t). Otherwise we had to interpolate or predict this following state or else the state-transition function as well and this could harm the accuracy of the interpolated experience.

4.2 Algorithm

Our algorithm triggers an interpolation after every step the agent takes. A query point $x_q \sim U(S)$ is drawn at random from the state space and all matching experiences:

$$D_{\text{match}} := \{e_t \in D_t | s_t = x_q\} \tag{10}$$

for that holds that their starting point s_t is equal to the query point x_q, are collected from the ER. Then for every action $a \in A$ all experiences that satisfy $a_t = a$ are selected from D_{match} in:

$$D^a_{\text{match}} := \{e_t | e_t \in D_{\text{match}} \wedge a_t = a\} \tag{11}$$

The resulting transitions are used to compute an average reward value r^{avg}_t and a synthetic experience e^{avg}_t for every distinct next state:

$$s_{t+1} \in \{s' | (s_t, a_t, r_t, s') \in D^a_{\text{match}}\} \tag{12}$$

is created. This results in a minimum of 0 and a maximum of 3 synthetic experiences per action and sums up to a maximum of 12 synthetic transitions per interpolation depending on the amount of stored transitions in the ER. As with the amount of stored real transitions, that can be seen as the combined knowledge about the model, the quality of the interpolated experiences may get better, a parameter $c_{\text{s\_int}}$ is introduced, that determines the minimum amount of stored experiences before the first interpolation.

4.3 Interpolation Component

Stein et al. introduce their IC (Stein et al., 2017) that this work uses as underlying basic structure for its interpolation tasks.

This IC serves as an abstract pattern and consists of a Machine Learning Interface (MLI), an Interpolant, an Adjustment Component, an Evaluation Component and the Sampling Points (SP). If the MLI receives a sample it is handed to the Adjustment Component, there, following a decision function, it is added to or removed from SP. If an interpolation is required, the Interpolation Component fetches required sampling points from SP and computes, depending on an interpolation technique, an output. The Evaluation Component provides a metric to track a so-called trust-level as a metric of interpolation accuracy.

We replaced the SP with the ER. It is realized by a FiFo queue with a maximum length. This queue represents the classic ER and is filled only with real experiences. To store the synthetic transitions another queue, a so-called *ShrinkingMemory*, is introduced. This second storage is characterized by a decreasing size. Starting at a predefined maximum it gets smaller depending on the length of the real experience queue. The *Interpolated Experience Replay* (IER) has a total size, comprising the sum of the lengths of both queues as can be seen in fig. 1. If this size is reached, the length of the ShrinkingMemory is decreased and the oldest items are removed, as long as either the real valued queue reaches its maximum length and there is some space left for interpolated experiences or the IER fills up with real experiences. This decision was made because it is expected, that near convergence the learner benefits more from actual experiences than from synthetic transitions spread over the state space. This approach includes a minimum size for the interpolated storage, but this was not further investigated in this work and is left for future work.

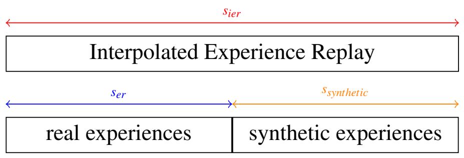


Figure 1: Intuition of Interpolated Experience Replay memory.

The IER algorithm as described in section 4.2 is located in the Interpolant, and, as stated above, executed in every step.

For the IER, we need to be able to find all experiences e in D that matches a randomly chosen first state. To make this efficient we use a dictionary that maps state-action pairs to their associated average rewards and distinct next states of all seen transitions. The dictionary is updated after every transition the agent makes.

To evaluate the quality of computed interpolations in future work, a metric could be designed to be used in the Evaluation part of the IC.

5 EVALUATION

5.1 Experimental Setup

For evaluation purposes, a linear regression in form of a neural network is used. This decision was felt because we use one input node for each state, that gives an overall amount of 64 input nodes. Neural networks have the ability to generalize over neighbouring areas, but using the architecture described above, this seems to have no effect because every state has its own input node. We therefore decided to reduce complexity by

Table 1: Overview of hyperparameters applied for the FrozenLake8x8-v0 experiment.

Parameter	Value
Learning rate α	0.0005
Discount factor γ	0.95
Epsilon start	1
Epsilon min	0.05
Update target net interval τ	300
Size of Experience Replay s_{er}	100k
Size of IER s_{ier}	100k
Start Learning at size of IER	300
Minibatch size	32

not using hidden layers. The observed results can be transferred to a more complex neural network (DQN), as a first step we sticked to the presented approach. One output node for every possible action was used, that results in 4 output nodes. Vanilla ER was selected as a baseline and compared with the IER approach presented in section 4. Preliminary experiments revealed the hyperparameters given in table 1, that are shared for all experiments. Furthermore, different capacities for storing synthetic experiences $s_{syntehtic}$ in combination with different warm-up phases, i.e., values for c_{s_int}, are investigated. As exploration technique a linearly decaying ε-greedy was used, and different durations t_{expl} tried. The different constellations of the individual experiments are shown in table 2. We measure the average return over the last 100 episodes to obtain a moving average that indicates how often the agent is able to reach to goal in this time. Each experiment was repeated for 20 times and the results are reported as the mean values and the observed standard deviations ($\pm$1SD) over the repetitions.

Each configuration was tested against the baseline and the differences have been assessed for statistical significance. Therefore, we first conducted *Shapiro-Wilk* tests in conjunction with visual inspection of *QQ-plots* to determine whether a normal distribution can be assumed. Since this criterion could not be confirmed for any of the experiments, the *Mann-Whitney-U test* has been chosen. All measured statistics, comprising the corresponding p-values for the hypothesis tests are reported in table 4.

5.2 Experimental Results

Fig. 2 depicts the results of the best three IER configurations as given in table 4. Experiment 1 and 2 were run for 1000 episodes. Experiment 3 for 1300 episodes, because of the longer exploration phase compared to the previous experiments. It can be observed, that the baseline approach (DQN using vanilla ER) stays below the IER approach (green line) and the latter one is converging on a higher value alongside a steeper increase, that indicates faster learning. This effect is even more distinct in the experiments with shorter exploration phases (experiments 1 and 2).

Table 2: Overview of the individually conducted experiment constellations.

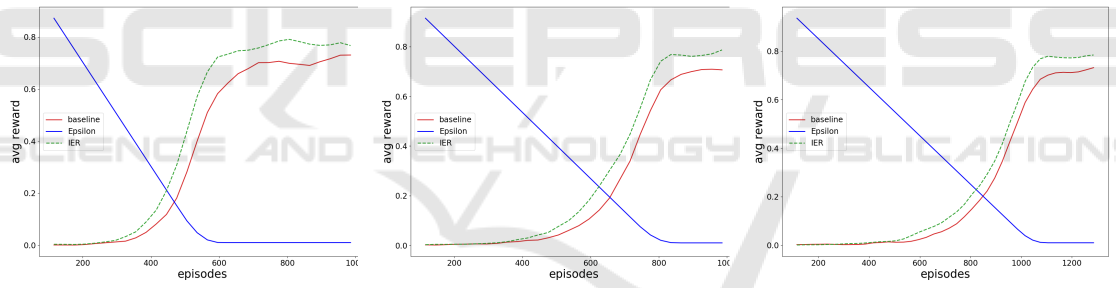
experiment	t_{expl}	s_{syn}	c_{s_inter}
1	500 episodes	20k	250
			500
			1,000
		100k	250
			500
			1,000
2	750 episodes	20k	250
			500
			1,000
		100k	250
			500
			1,000
3	1,000 episodes	20k	250
			500
			1,000
		100k	250
			500
			1,000

Figure 3 reports the results of all experiments. The plots reveal, similar to fig. 2, that the IER approach outperforms the baseline. All the tested IER configurations perform similarly well, with only marginal deviations. It turns out that the choice of s_{syn} and c_{s_int} only has little to no effect. Because the IER algorithm performs better than the baseline, and this effect is even bigger in the scenarios with shorter exploration phases, it can be used to decrease the time needed for exploration.

In fig. 4 the size of the IER can be seen. As the choice of c_{s_int} does not have a huge effect on the amount of interpolated experiences compared to the maximum size we plotted only the graphs for the configurations of the best results. The crossed curves represent the amount of stored interpolated experiences and the dotted curves the amount of stored real experiences. The red curve depicts the baseline and the amount of real transitions is slightly above the IER variants in all three experiments. Taking into account that, first, an episode ends after the agent has, either reached the final state, fell into a hole or reached the maximum time limit, and, second, the IER agents performed better, it seems that the baseline agent learned to avoid falling into a hole, but does not reach the final state as often as the other agents. This explains the higher amount of experiences. Fig. 4a shows that

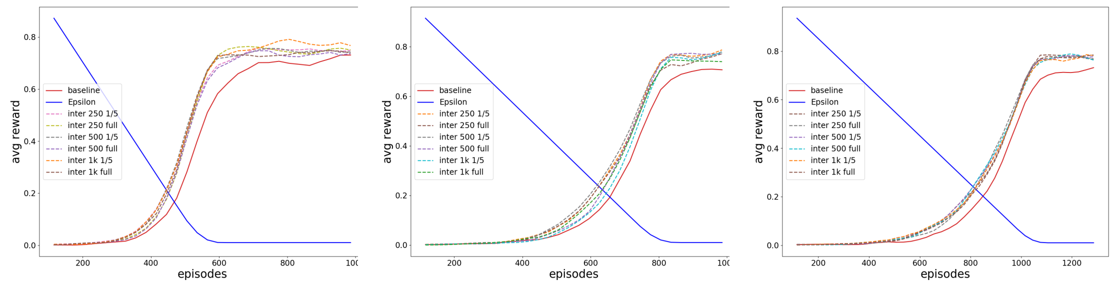
Table 3: Summary of results.

experiment	s_{syn}	c_{s_int}	Mean	±1SD	p-value Shapiro-Wilk	p-value Mann-Whitney-U
1	0	0	0.3772	±0.3121	1.9085e-33	
	20,000	250	**0.43**	±0.3277	2.0967e-34	1.2702e-19
		500	**0.4324**	±0.3297	6.5603e-35	6.6317e-22
		1,000	**0.4416**	±0.3396	2.8203e-34	2.4728e-23
	100,000	250	**0.4287**	±0.3364	3.611e-35	3.2159e-22
		500	**0.4168**	±0.3266	9.4551e-35	1.4136e-13
		1,000	**0.4261**	±0.3282	3.0105e-35	9.3601e-21
2	0	0	0.2385	±0.2807	8.7954e-35	
	20,000	250	**0.2877**	±0.3126	1.9496e-33	1.6066e-06
		500	**0.2911**	±0.3105	5.4785e-33	1.9508e-06
		1,000	**0.2653**	±0.309	9.4381e-35	1.3255e-02
	100,000	250	**0.2785**	±0.3018	5.5895e-33	1.7829e-04
		500	**0.2782**	±0.3155	1.555e-34	2.7468e-03
		1,000	**0.2734**	±0.3026	7.4416e-34	1.0413e-03
3	0	0	0.0885	±0.1347	5.9146e-38	
	20,000	250	**0.1194**	±0.1618	3.9763e-35	3.1267e-09
		500	**0.1236**	±0.1642	1.1407e-34	6.4606e-09
		1,000	**0.1215**	±0.161	8.5940e-35	1.0439e-09
	100,000	250	**0.1198**	±0.1716	1.901e-36	2.5456e-03
		500	**0.1229**	±0.1666	3.8836e-35	3.5305e-03
		1,000	**0.116**	±0.1602	1.5933e-35	6.5812e-04



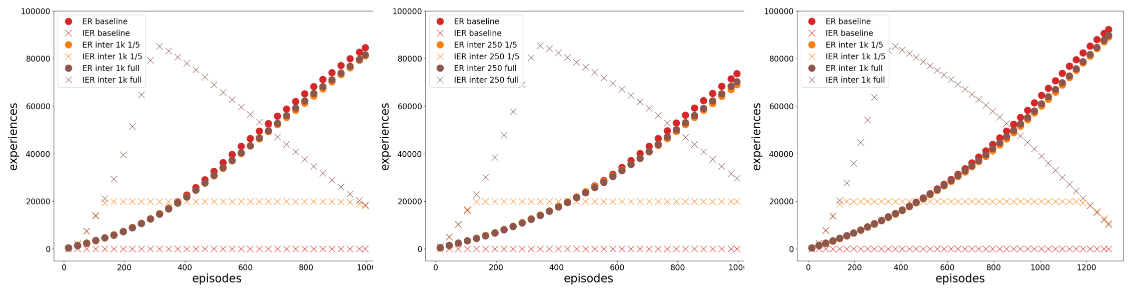
(a) Experiment 1. (b) Experiment 2. (c) Experiment 3.

Figure 2: The best results among all conducted experiments. The solid red line represents the classical ER serving as baseline to compare with. The dashed green line shows the average reward of the IER approach. The blue line depicts the decaying epsilon. The lines for IER and the baseline represent the repetition averages.



(a) Experiment 1. (b) Experiment 2. (c) Experiment 3.

Figure 3: All experiments with all perturbations of t_{expl}, s_{syn} and c_{s_int}. The dashed lines show the results of the single experiments. The blue line depicts the decaying epsilon. The x-axis represents the episodes and the y-axis the average reward of all 20 repetitions.



(a) Experiment 1. (b) Experiment 2. (c) Experiment 3.

Figure 4: The size of the IER represented by the amount of real and synthetic experiences. c_{s_int} was chosen from table 4. The crosses represent the amount of synthetic and the dots the amount of real experiences. Brown show the size of the IER with $s_{syntehtic} = 100,000$ and orange with $s_{syn} = 20,000$. The red curves represent the baseline. The x-axis marks the episode length and the y-axis the amount of stored experiences.

Table 4: Best IER configurations found during the parameter study.

experiment	s_{syn}	c_{s_int}
1	20,000	1,000
2	20,000	250
3	100,000	1,000

the ratio of experiences at the end of the exploration phase is in favor of the synthetic ones in the case of $s_{syn} = 100,000$. Fig. 4b shows that the ratio changed but is still in favor of the synthetic transitions and fig. 4c shows that at this time the ratio is in favor of the real experiences. If we look at the graphs for $s_{syn} = 20,000$, then all ratios are in favor of the real examples, but also not that far away from a ratio of 50/50 in experiment 1 and 2. This seems to be a good choice as the best results were achieved with a choice of s_{syn} that is close to an equal distribution of interpolated and real transitions. This should be investigated further.

6 CONCLUSION AND FUTURE WORK

We presented an extension for the classic ER used in Deep RL that includes synthetic experiences to speedup and improve learning in nondeterministic and discrete environments. The proposed algorithm uses stored, actually seen transitions to utilize the experience of the model that serve as basis for the calculation of synthetic (s,a,r,s) tuples by means of interpolation. The synthetic experiences comprise a more accurate estimate of the expected long-term return a state-action pair promises, than a real transition does. So far the employed interpolation technique is a simple equally weighted averaging that serves as an initial approach. More complex methods in more complex problem spaces have to be investigated in the future. The IER approach was compared to the default ER in the FrozenLake8x8-v0 environment from the OpenAI Gym and showed an increased performance in terms of a 17% increased overall mean reward. Several configurations for the maximum size of the stored synthetic experiences, different warm-up times for the interpolation, as well as different exploration phases were examined, but revealed no remarkable effect. Nevertheless, a ratio of 50/50 for real and synthetic experiences in the IER seems promising and needs further research.

As the algorithm creates a synthetic experience for every action and every follow-up state there is a huge amount of transitions created that could be decreased in a way that takes further knowledge into account. An example would be that only those actions are considered, that the actual policy would propose in the given situation. Or only for that follow-up state that has the most (promising) stored experiences in the storage. Also, further investigation of the composition regarding the IER seems interesting, since, as stated above, the ratio of the stored transitions might have an effect. As the evaluation was limited to the FrozenLake environments provided by OpenAI Gym, the proposed algorithm could be tested on more complex versions that differ in size and difficulty. Also a continuous version with a greatly increased state and action space is required for deeper analysis.

As of yet, the proposed approach is limited to discrete and nondeterministic environments. We plan to develop the IER further to solve more complex problems (increased state and action space) as well. To achieve this, a solution for the unknown follow-up state is needed, that could also be interpolated or even predicted by a state-transition function that is learned in parallel. Here the work from (Jiang et al., 2019) could serve as a possible approach to begin with. A yet simple, but nevertheless more complex problem, because of its continuity, that is beyond the domain of grid worlds is the MountainCar problem. Other, more complex interpolation techniques have to be examined to adapt our IER approach in this environment. And at last, the impact of interpolated experiences on more sophisticated experience replay mechanisms such as Hindsight ER and Prioritized ER have to be investigated as well.

REFERENCES

Andrychowicz et al. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc.

Brockman, G. et al. (2016). Openai gym.

De Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2015). The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, NIPS*.

Jiang, W., Hwang, K., and Lin, J. (2019). An experience replay method based on tree structure for reinforcement learning. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3):293–321.

Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Mnih et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

O'Neill, J., Pleydell-Bouverie, B., Dupret, D., and Csicsvari, J. (2010). Play it again: reactivation of waking experience and memory. *Trends in Neurosciences*, 33(5):220 – 229.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized Experience Replay. *arXiv e-prints*, page arXiv:1511.05952.

Stein, A., Menssen, S., and Hähner, J. (2018). What about Interpolation? A Radial Basis Function Approach to Classifier Prediction Modeling in XCSF. In *Proc. of the GECCO*, GECCO '18, page 537–544, New York, NY, USA. Association for Computing Machinery.

Stein, A., Rauh, D., Tomforde, S., and Hähner, J. (2017). Interpolation in the extended classifier system: An architectural perspective. *Journal of Systems Architecture*, 75:79–94.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

Zhang, S. and Sutton, R. S. (2017). A deeper look at experience replay. *CoRR*, abs/1712.01275.