Experience Replay for Deep Reinforcement Learning: Towards an Organic Sampling Technique

Marc-Alexander Frey
35313329
m.frey@lancs.ac.uk

Abstract

The advances in deep reinforcement learning have not only made the underlying function approximation by deep neural networks more stable. Out of the many approaches to accomplish better convergence, especially the well established method of experience replay has shown efficient. But whilst it has been proposed as a plain random sampling technique in it's very beginning, various algorithms have since been proposed. However, they are quite inorganic, since none of these has yet considered the temporal difference and the correlating tendency to forget older impressions. This paper proposes a new replay logic that can be mounted on top of existing experience replay forms and is considering this. This variant A comparison with its plain pendant and the widely used Prioritized Experience Replay is carried out. The replay variants are tested in regards to their learning performance and it is shown, that the new technique hast the potential to perform better than the uniform sampling technique.

1 Introduction

Since the development of Deep Q Learning during the last decade, reinforcement learning experienced a huge boost in performance, beating the best players in different disciplines of games, such as Go or even Chess. This was mainly possible by combining different techniques to make nonlinear function approximation more stable. One of the most important publications on the matter was done by Google's Deep Mind Team. The 2015 paper, as first of it's kind, proposed two extensions to Deep Q-Learning, which were **experience replay** (ER) and the use of **two different networks** for Q value estimation, known as double deep Q learning, its architecture consisting of two networks, named double deep Q network (DDQN). While in DDQNs the first network namely ”Q” or ”action” network, is fitted after each action taken and used to predict the Q values to obtain the next action, the other network, named target network, is only updated every k steps with the weights of the online network. This is to ensure, that the target against which the online network is fitted are not volatile but fixed for a certain period, since the loss optimization is more stable optimizing towards a fixed target.

Experience Replay on the other side, is a buffer of past experiences. It is filled after each action taken by the agent and used for sampling experiences to fit the online network with. The buffer hence mimics the human memory and helps to make more use of already faced transitions and removes existing correlation between consecutive experiences, otherwise biasing the agent. And whilst the synchronization of online and target network can either be done as hard or soft update, experience replay offers more potential for improvement. Since its proposal by Mnih et al. (2015) using a simple uniform random sampling approach, many other sampling methods have been proposed, underpinning the potentials it bears.

This report explores and compares the uniform sampling technique, as well as Prioritized Experience Replay as one of the most famous approaches. Further than that, a new and easy approach named ”Short-TErM Memory Experience Replay” (STEMMER) is proposed, which pays higher attention to more recent experiences just like humans tend to more likely remember experiences they stored in their short term memory more recently, than older information. The new approach is thereafter described and experimentally analysed.

2 Related Work

Out of the proposed experience replay optimizations, Prioritized Experience Replay (PER) by Schaul et al. (2015) represents the most famous one. It works by assigning higher sampling priority to experiences, with a higher deviation between estimated and target Q-value, known as temporal difference error δ_t.

Other more recent approaches were proposed by (Andrychowicz et al., 2017) or (Brittain et al., 2019). Especially (Novati and Koumoutsakos, 2018) tackles the problem often seen with uniform experience replay, which impedes learning success. The issue occurs when the optimal policy diverges from past experiences, which can be the case, when an agent makes new, unseen experiences, e.g. due to changes in the game dynamics. Novati and Koumoutsakos (2018) propose a solution named "remember and forget", in which some experiences that are too different from the current policy are selectively omitted.

The in this paper followed idea of training two different estimators has been initially applied to Q-learning by Hasselt (2010) and was later adapted to non-linear function approximation in deep Q-learning by Hasselt et al. (2015) and Mnih et al. (2015). It solves the problem of the "moving target" that occurs with only a single model by having a target value fixed for a certain period. It has shown, that this increases stability and convergence.

Another important work to mention is by Hessel et al. (2017), proposing the so called rainbow technique, combining several improvements in Deep Q-Learning - especially to mention PER - to create a new benchmark technique.

Even though ER has been explored widely, yielding numerous adaptions with different focuses, none of the works has yet explored ER that considers the time differences between experiences stored in the replay buffer.

3 Objective

Considering temporal difference in experience making, mimics the memory of human beings more adequately, since we are more likely to recall recent experiences, while we tend to forget such that are far in the past. The existing ER approaches contrarily focus mostly on determining the importance of experiences.

This work aims to propose an adaption of ER, that is exactly that: a more natural, organic representation of the human memory by considering time differences in experience making.

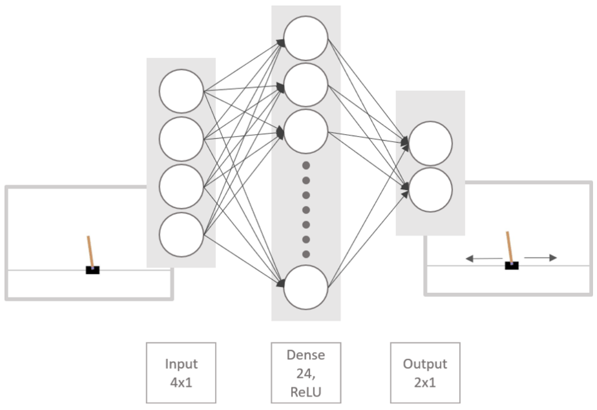


Figure 1: Architecture of the used Network

4 Methodology

4.1 Environment

In this paper, the gym implementation of Open AI is used as engine. It resembles several games as Markov-Decision problem (MDP) and provides a unified API for the input and response to and from the game. The environment used in the following is "CartPole" a well known and simple problem, with the target to balance a pole mounted on a cart moving either left or right, while preventing it to fall down or running out of the visible window. The reason of choosing such a simple environment is, that whilst experimenting with different, more complex environments, such as Space Invaders with pixel data response, have proven to train extraordinarily long and don't converge in an appropriate training time.

CartPole contrarily is quicker to train, due to its popularity easily comparable and can be seen partially representative for more complex problems such as space invaders, hence appears suitable for the given Proof of Concept.

For training, a double deep Q learning agent is used with different experience replay methods, of which more details will follow in the subsequent section.

The neural network architecture used for function approximation can be seen in Fig. 1. It is a simple one , given the environment and consists of a 4 signal input layer, a 24 node hidden layer and a 2 positional output.

4.2 Short-Term Memory Experience Replay

In the following, the new Short-Term Memory Experience Replay approach is explained. The cognitive behaviour of forgetting experiences over time is only unsatisfactorily mimicked by uniform random sampling from the experience buffer. A more organic way of memory representation is given by sampling more recent experiences with a higher probability than these that have occurred further in the past. To accomplish this, instead of sampling each object in the memory with equal probability, the entries are assigned a probability in order of appearance, meaning recent entries are sampled with a higher probability, than older records.

Since the probabilities are changing over time due to the volatility of the memory, the assignment of probabilities is executed during each experience replay. Therefore, within this paper, the following simple formula is used

$$f(x) = a^{(m \cdot x)}$$

, where a and m tuneable parameters, changing the shape of the distribution and hence influences sampling probability. The following value ranges have shown reasonable

$$0.5 < a \leq 1$$

$$0.2 < m < 1$$

where m determines the probability span between most recent and oldest samples. The underlying formula can be further changed and adjusted. It is noteworthy, that it is required to subtract the product of m and i from 1, to comply with the queue object indexing the oldest elements with the lowest index. The complete algorithm embedded into Double Q-learning (Hasselt et al., 2015) can be found in Algorithm 1.

According to Algorithm 1[1], for each x, the function value is calculated and assigned to the Vector V. To obtain a probability, the resulting values are normalized by division with the sum of all entries in V, resulting in V_N, whose elements sum to 1. In the memory queue, the oldest entry has the lowest index. To ensure, that older indices receive a lower probability, the indices of V_N are flipped. Based on these probabilities, values are sampled from the memory and used for training the online network.

[1]Note, that the target network update is performed as hard update after every iteration, different from Hasselt et al. (2015)

Algorithm 1: Organic Experience Replay with STEMMER

```
Initialize replay buffer $D$ with size n, priority vector $V$ with size n,
  batch size $b$, hyperparameters $a$ and $m$
foreach iteration do
    foreach environment step do
        Observe state $s_t$ and select $a_t \sim \pi(a_t, s_t)$
        Execute $a_t$ and observe next state $s_{t+1}$ and reward
          $r_t = R(s_t, a_t)$
        Store $(s_t, a_t, r_t, s_{t+1})$ in replay buffer $D$
    end foreach
    foreach update step do
        – STEMMER –
        foreach x in memory do
            $V(x) \leftarrow a^{m \cdot x}$
        end foreach
        normalize by sum of V
        $V_N(x) \leftarrow \frac{V(x)}{\sum_{x=0}^{n} V(x)}$
        resulting in discrete probability distribution
          $P(x) \geq 0 \forall x \epsilon V; \sum P(x) = 1$
        Flip indices of $V_N$
    end foreach
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim P(i)$
    Compute target $Q$ value: $Q^*(s_t, a_t) \approx$
      $r_t + \gamma Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1}, a'))$
    Perform gradient descent step on
      $(Q^*(s_t, a_t) - Q_\theta(s_t, a_t))^2$
end foreach
Update target network parameters: $\theta' \leftarrow \theta$
```

In this work, the two hyperparameters are set as follows examplarily:

$$a = 0.9$$

$$m = 1$$

STEMMER not only has the potential to solve the problem discussed in Novati and Koumoutsakos (2018) which occurs when the policy diverges from past experiences, by paying more attention to recent experiences, that can be assumed to be closer aligned with the current policy, since the corresponding actions are derived from it. Further than that, it can seamlessly be mounted on existing ER approaches, such as prioritized experience replay, by additionally storing the order of occurrence in the tree when adding new records.

4.3 Comparison

After the new algorithm has been defined, an analysis is carried out in the next section. For this purpose, as a baseline, the new approach is compared against two other, well established experience replay methods. The first one to mention is uniform random sampling. Since within this work, STEMMER is applied atop of it, it can be seen as the baseline. Therefore it is of particular interest, how STEMMER performs in comparison. Additionally prioritized experience replay has shown to perform well in deep Q learning and is widely used, thus also taken into consideration.

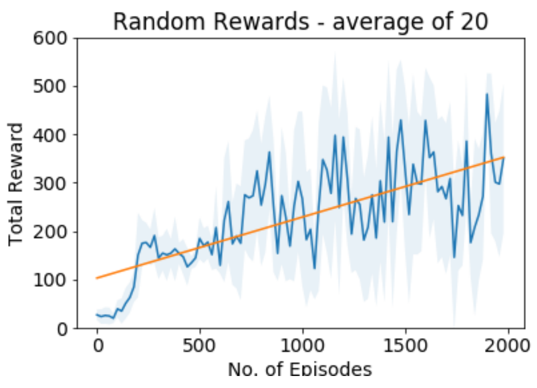


Figure 2: Random Sampling Learning Trend

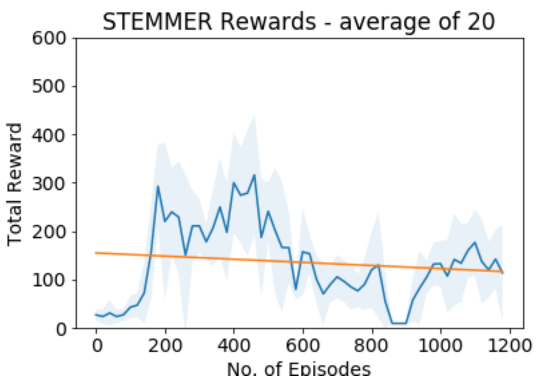


Figure 4: STEMMER Sampling Learning Trend[2]

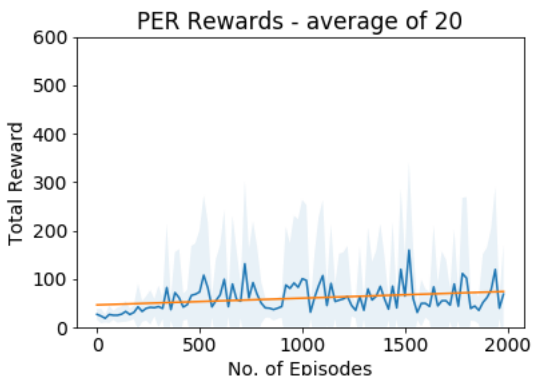


Figure 3: PER Sampling Learning Trend

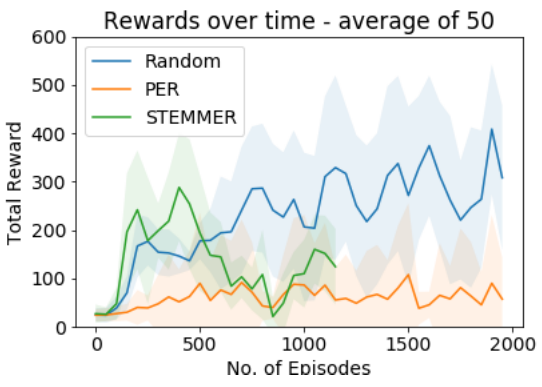


Figure 5: Rewards over time

The evaluation metric used for these comparisons is the reward the agent scored during each iteration, where each non terminal action (moving left or right) yields a reward of 1 and loosing the game (dropping the pole or leaving the window borders) is penalized with a reward of -10. To ensure comparability, all approaches are executed with the same hyper parameters such as learning rate, epsilon decay etc. The environment as well as the used libraries are seeded were required. Based on the so gathered data, a thorough statistical analysis is carried out, showing similarities and differences of the approaches.

5 Analysis & Results

From the plot in Fig. 5 and the attached graphs in the appendix, it can be seen, that uniform random sampling performs consistently best.

PER unexpectedly enables some learning, but remains below STEMMER and Random during the whole period of 2000 episodes.

It is interesting, that from the beginning on, STEMMER seems to learn much quicker than Random and PER and peaks at around 350 episodes. However after peaking, the gained rewards start falling, surpassing Random at 500 iterations and approaching PER after 750 episodes.

Further investigating the period until STEMMER peaks up until 400 periods, the regression slopes underpin the good performance of Random (slope $m_{Random} \approx 10$) and STEMMER (slope $m_{STEMMER} \approx 13$) with a small leap of the latter. PER only gains 2 additional reward points per period played, but whatsoever learns.

However it has to be mentioned, that due to the high standard deviation, STEMMERS supremacy over Random for a low number of episodes cannot be fully supported.

The statistical analysis indicates, that STEMMER enables the agent to train as well as – or even slightly better – than it does with uniform random sampling for the first 500 periods. Yet it decreases thereafter, and oscillates between Random and PER. A possible reason for this oscillation to occur is, the behaviour of the STEMMER function. Due to its exponential nature and therefore strong favouring of recent experiences during sampling, recently made experiences, may they be rewarding or not, are forcing the agent.

This may also explain the steep learning curve in the very beginning, since probably the agent has made a number of valuable experiences by random, they were sampled with high likeliness and hence let the policy adjust accordingly. The same applies for experiences with low rewards: if a critical number of these occur, the policy is adjusted to minimize loss for these and hence expanded into a wrong direction. This back and forth may have caused the volatile behaviour during testing.

The computational performance has shown to be about 40% lower than without exponential sampling. This is also the reason, why this data is only available for the first half of periods.

Also, CartPole is not prone to enforce the development of policies, which are strongly deviating from the memory content, since the game concept is easy and does not change over time, other than e.g. Space invaders with destructible game environment, moving and dying enemies etc. This is, STEMMER must be tested in such an environment in order to determine whether it can help to reduce the issue of diverging policies, as described by (Novati and Koumoutsakos, 2018).

6 Discussion

The results of this study can only be seen partially representative. While STEMMER has shown to converge quicker in the current set up, than the uniform sampling, three pillars for improvement can be pointed out that are slightly lowering its weight

- The algorithm has only been tested with an underlying uniform random sampling, not with others such as prioritized replay.
- Only a single, very simple environment was tested.
- The test configurations such as seeds and parameters were fixed during all trails presented.

One way to increase the comparably weak computational performance is by adjusting the used sampling distribution, instead of sampling from the whole memory, a cutoff could be done, so that e.g. only the first n samples from the memory are taken into consideration and sampling probabilities have to be calculated.

7 Conclusion and Future Work

In this work, a quick overview over the most common approaches for experience replay was given. The lack of timely considerations for experience replay were underpinned and a new approach named STEMMER was deducted and formulated. The analysis has shown that the new technique has the potential to improve existing replay methods, but more and wider analysis is required to verify this reliably.

Especially given the points stated in ”Discussion”, to verify the findings and increase their generalizability, trails with several more complex environments should be conducted, expanding the application of STEMMER to existing replay approaches. Prioritized experience replay could be easily enabled, by also storing an iterator in the tree leafs, giving the order of when an entry was stored and considering this priority for crafting a distribution on top of the TD-error based priority. Also different parameter settings and seeds need to be incorporated could be tested as subsequent steps to this paper, because the results are strongly depending on these pre-sets and in order to improve the quality of its results. Should these trails show positive, further improvements of the computational performance could be carried out.

Even tough further work is required to gain more representative results, the approach proposed in this report can be seen as a new and disruptive tool to further tune and improve existing replay procedures to learn quick and organic. Also the work can be seen as a starting point, to further explore how reinforcement learning can be moved closer to how we as humans think, learn and evolve.

8 Acknowledgements

The author would like to thank Leandro Soriano Marcolino, University of Lancaster and Olusanmi Adekunle Hundogan, University of Utrecht, for providing helpful advice throughout this work.

The Code for Prioritized Experience Replay was adapted from: https://github.com/rlcode/per

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay.

Marc Brittain, Josh Bertram, Xuxi Yang, and Peng Wei. 2019. Prioritized sequence experience replay.

Hado V. Hasselt. 2010. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc.

Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press.

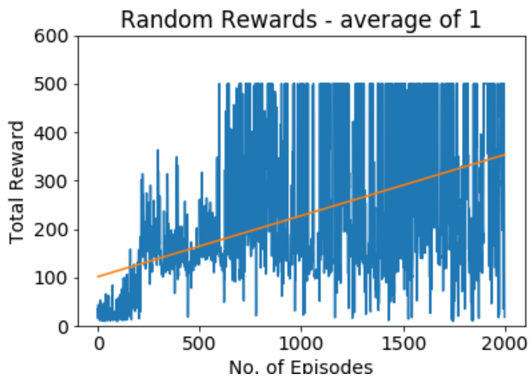
Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2017. Rainbow: Combining improvements in deep reinforcement learning.

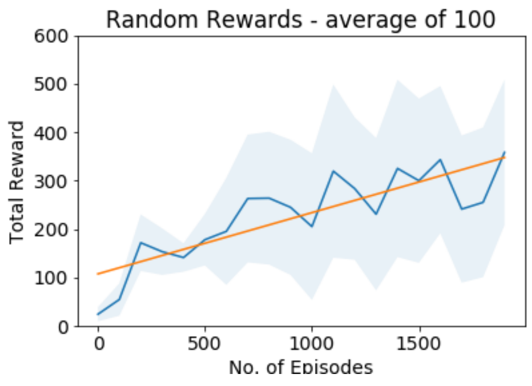
Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

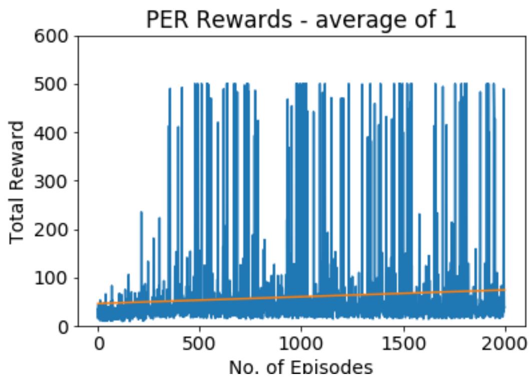
Guido Novati and Petros Koumoutsakos. 2018. Remember and forget for experience replay.

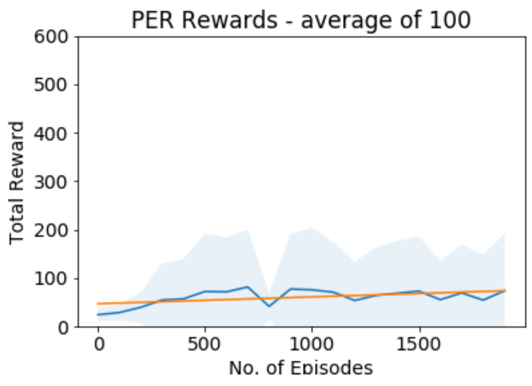
Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay.

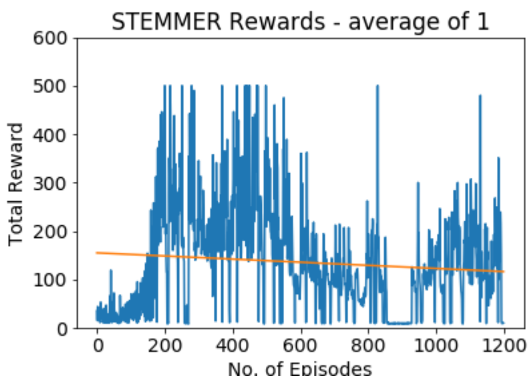
9 Appendix

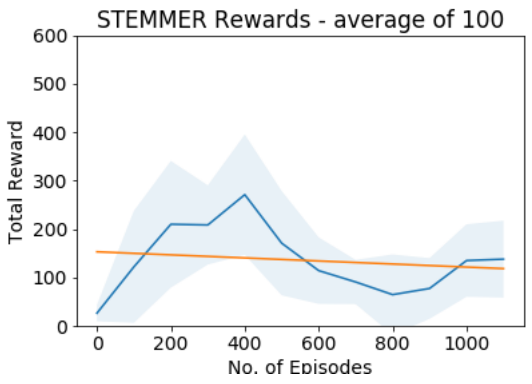


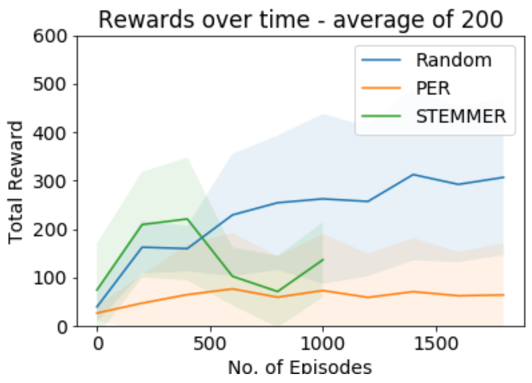







STEMMER Rewards - average of 1
Total Reward
No. of Episodes
0
100
200
300
400
500
600
0
200
400
600
800
1000
1200


STEMMER Rewards - average of 100
Total Reward
No. of Episodes
0
100
200
300
400
500
600
0
200
400
600
800
1000


Rewards over time - average of 200
Total Reward
No. of Episodes
Random
PER
STEMMER
0
100
200
300
400
500
600
0
500
1000
1500