Extending DQN with Double Q-learning and Prioritized Experience Replay

Merckx, Yannick
Vrije Universiteit Brussel

ABSTRACT

Prioritized Experience Replay and Double Q-learning are two known improvements to the performance of Deep Q-networks (DQN). In this paper, we implemented these two improvements and evaluated them emprirically on two Atari Games, namely Krull and Boxing. As a conclusion, we can confirm the improvement of Double Q-learning. Prioritized Experience Replay left use with an unstable version of DQN and failed to learn the games. Several tests where conducted to test the working of the implementation and find a possible bug, but none could be found.

1. INTRODUCTION

Prioritized Experience Replay [Schaul et al., 2015] and Double Q-learning [Van Hasselt et al., 2016] are two adjustements to Deep Q-networks (DQN), who are known to sometimes improve the performance. A Deep Q-network is a reinforcement learning algorithm that achieves human-level performance across many Atari Games [Mnih et al., 2015]. The goal of reinforcement learning [Sutton and Barto, 1998] is to learn good policies for sequential decision problems, by optimizing a cumulative future reward signal. One of the most popular reinforcement learning algorithms is Q-learning [Watkins and Dayan, 1992]. One of the drawbacks of Q-learning is that it is known for occasionally learning extremely high action values. This is caused by the maximization step over the estimated action values, which tends to prefer overestimated to underestimated values. DQN is using the Q-learning algorithm and also tends to be overoptimistic. Double Q-learning is a reinforcement learning algorithm that can replace the Q-learning algorithm in the DQN and reduces the overoptimistic behaviour [Van Hasselt et al., 2016]. Prioritized experience replay improves the performance in a total different way. With online reinforcement learning, the agent updates incrementally his parameters, while it is observing the stream of experiences. Reinforcement learning is known to be unstable or even to diverge when nonlinear function approximations. And since DQN is using a neural network to represent the action-value function, DQN was suffering instability. The two main reasons of this instability were the close correlation between the updates and inability to remember rare and interesting experiences. Mnih et al. resolved this issues [Mnih et al., 2015] by adding *Experience replay* [Lin, 1992] to the DQN. Prioritized replay makes this experience replay more efficient and effective by prioritizeing the selection of transitions. Where with the original DQN, the experiences were uniformly selected from the memory. Now what we will try in this paper, is to implement these extensions and empirically evaluate them on two Atari Games. The games we have selected are Krull and Boxing. We will first discus the implementation details, before we continue with the experimental results.

2. BACKGROUND

When we want to solve a sequential decision problem, we can learn to estimate the optimal value for each action. This optimal value can be described as the maximum discounted future reward. Formally we describe the true value of an action a in state s, when using a certain policy π by:

$$\mathbb{Q}_\pi(a,b) \equiv \mathbb{E}[R_1 + \gamma R_2 + ... \ |S_0 = s, A_0 = \alpha, \pi] \quad (1)$$

where $\gamma \in [0,1]$ is the discount factor, which allows us to express more importance to immediate rewards than later rewards. Then, the optimal value is $Q_*(s,a) = max_\pi Q_\phi(s,a)$ The optimal policy is a greedy policy, which selects the highest action value for each state.
Q-learning [Watkins and Dayan, 1992] allows use to learn these optimal action values. Because it is for most interesting problems too large to calculate all action values, we can try to approximate the optimal values by a parametrized value function $Q(s,a;\theta_t)$. The update function of the parameters follows the standard Q-learning update:

$$\theta_{t+1} = \theta_t + \alpha(Y_t^Q - Q(S_t, A_t; \theta_t))\nabla_{\theta_t} Q(S_t, A_t; \theta_t) \quad (2)$$

with R_{t+1} the immediate observing reward, after taken action A_t in state S_t. α is here the learning rate and Y_t^Q the target, which is defined as:

$$Y_t^Q \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t) \quad (3)$$

Note that, there is a big resembles in the update rule of θ and stochastic gradient descent. Mnih et al. saw this resembles too during the development of the Deep Q-network and used this to implement Q-learning together with neural networks [Mnih et al., 2015].

3. DEEP Q-NETWORKS

A Deep Q-network (DQN) is a multilayer neural network that uses end-to-end reinforcement learning [Mnih et al., 2015]. As earlier mentioned, we see a lot of similarities between equation 2 and stochastic gradient descent. In DQN is

the parameter θ from equation 2 represented by the weights of the network. Note that, Reinforcement learning is known to be unstable for non linear approximations of the action values. This is also the case for DQN, since it is a neural network. [Mnih et al., 2015] came up with two smart ingredients, which resolved the issue and dramatically improved the performance. The first ingredient is experience replay, which will be discussed in section 6. The second one is the use of a target network. The target network is just the same as the online network, only with older parameter values θ^-. The parameter values θ^- are periodically updated with the parameter values θ of the online network and are kept fixed for a certain amount of steps. This causes, together with the experience replay, for a decorrelation for the experiences and resolves the instability. Because of the target network is the target of DQN slightly different:

$$Y_t^{DQN} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t^-) \tag{4}$$

4. DOUBLE Q-LEARNING

Double Q-learning [Hasselt, 2010] decouples the evaluation and selection of an action. Standard Q-learning sometimes result in extremely high action value. Together with the max operation in equation 4 and 2 are the overestimated action values selected and tends the Standard Q-learning to be overoptimistic.[Van Hasselt et al., 2016]

The decoupling of the selection happens by introducing two parameters θ_t and $\theta_t^{'}$. The target for Double Q-learning can be defined as:

$$Y_t^{DQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_t); \theta_t^{'}) \tag{5}$$

The first weights θ_t are the online weights, which are used to select the action. The second weights $\theta_t^{'}$ are used to evaluate the selected action more fairly. Note that, θ_t and $\theta_t^{'}$ can be updated symmetrically by switching the roles of the two weights after a certain amount of time.

5. DOUBLE DQN

Double DQN solves the issues of overoptimistic behaviour by using DQN with Double Q learning. In section 4, we saw that decoupling the selection and evaluation of the action value requires two separate weights. Further did we see in section 3, that DQN is already in the possession of a second set of weights, namely the target network. This makes is very easy to integrate double Q-learning in DQN. We define the target $Y_t^{DoubleDQN}$ as:

$$Y_t^{DoubleDQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \theta_t); \theta^-) \tag{6}$$

The second weights $\theta_t^{'}$ of equation 5 are replaced by the weights of the target network θ^-. Note that, θ^- still gets periodically updated by the online weights (as in the original DQN) and not symmetrically, as suggested in section 4.

6. PRIORITIZED EXPERIENCE REPLAY

Prioritized Experience Replay is a different approach on fetching the experiences from the replay memory. DQN uses experience replay to break the correlation between the experiences. This is realised by saving all transitions and fetching them uniformly from the replay memory. The problem with this approach is that rare experience are forgotten and a lot of non-informative transitions are replayed.With Prioritized Experience replay we prioritize the transitions and replay them based on their prior. [Schaul et al., 2015] discovered that prioritizing the transitions, makes the experience replay most of the time more efficient and effective.

6.1 Prioritizing with TD-error

The measure for prioritizing every transition is essential to let this method work well. As a measurement to calculate the priors, we choice the temporal difference error (TD-error). The idea behind taking this measurement is the fact that the experiences with the biggest TD-error are to most 'interesting' to replay. The TD-error δ_j for the transition j of a vanilla DQN can be described as:

$$\delta_j = R_j + \gamma_j \max_a Q(S_t, a) - Q(S_{t-1}, A_{t-1}) \tag{7}$$

As we can see is this a very convenient measure, since we already need to calculate it for the DQN. Although, there are some issues when using the TD-error. First of all updating the TD-errors. To update the TD-error, we need to sweep the entire replay memory. As a solution, we suggest to update only the TD-error, when the experience is replayed. Note that, as a consequence experiences with a low TD-error may take a long time before they get replayed. Another issue is slow convergence and the lack of diversity , when taking a greedy replay selection.
As a solution for these issues, we use Stochastic Prioritization.

6.2 Stochastic Prioritization

Stochastic Prioritization is an alternative selection method for the greedy selection method. This is also the sampling method, we will use during our experiments. Based on a certain probability, each sample will be picked from the replay memory. We define the probalibity for sample i by:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{8}$$

with p_i the priority of sample i and α determining the importance of the priority.

The priority can be represented in many ways. We represent it rank-based:

$$p_i = \frac{1}{rank(i)} \tag{9}$$

where $rank(i)$ is the rank of instance i in the sorted replay memory, according to $|\delta|$.

Important to note is that these stochastic updates introduce bias. We correct this by introducing importance-sampling (IS) weights. [Schaul et al., 2015]

$$w_i = (\frac{1}{N} \cdot \frac{1}{P(i)})^\beta \tag{10}$$

This correction is applied in the Q-learning update rule, where we use weighted deltas $w_i \delta_i$ instead of regular delta's δ_i

Now, we know how Prioritized Experience Replay and Double Q learning are implemented in a DQN structure, we can continue with the experiments.

7. ATARI EXPERIMENTS

The main goal of this paper is to see how these adjustments perform with DQN. We will answer this question by empirically evaluating the adjustments separately and combined on two Atari Games. We take as a baseline the performance of the vanilla DQN on the Atari games.

7.1 Experimental Setup

The two Atari games, that will be learned are Boxing and Krull. All experiments are ran 5 times on the Hydra cluster with 16gb for each run. The network architecture for all experiments is the same and is identical to the network used in [Mnih et al., 2015]. The source code of this experiment can be found on: https://github.com/cmaixen/DQN-tensorflow

7.1.1 Hyper-parameters

In all the experiments were the same hyper-parameters used. The parameter values were based mainly on the hyper-parameters of the tuned DQN in [Mnih et al., 2015]. The discount was set to $\gamma = 0.99$, and the learning rate to $\alpha = 0.00025$. The number of steps between target network updates was $\tau = 10000$. The training steps was set to 20 million. Although, we had not the chance to observe the results for all the steps, due the slow calculation speed and the deadline of this project. The size of the experience replay memory will be 100000 transitions. The minibatch size is 32 and the history length and action step are both 4 steps. At last we have ϵ, which is at the start 1 and is linearly decreasing to 0.1.

7.2 Empirical Results of Double DQN

The first emperical results are from Double DQN learning agent on the two Atari games Krull and Boxing. We've ran for both environments also the Vanilla DQN to have a baseline. We ran all experiments with paired seeds.

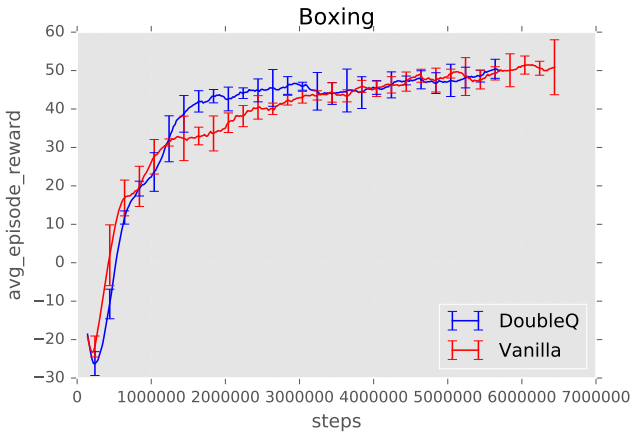


Figure 1: Average Episode reward with Double DQN for Boxing

For the Atari Boxing game, it seems that the Double DQN tends to perform equally as the Vanilla DQN, but for Krull, we can clearly see that the average episode reward of double DQN lies higher than the baseline. Also do we see very clear the point, where decoupling of the selection and evaluation comes into its own. Both methods learn at an equal

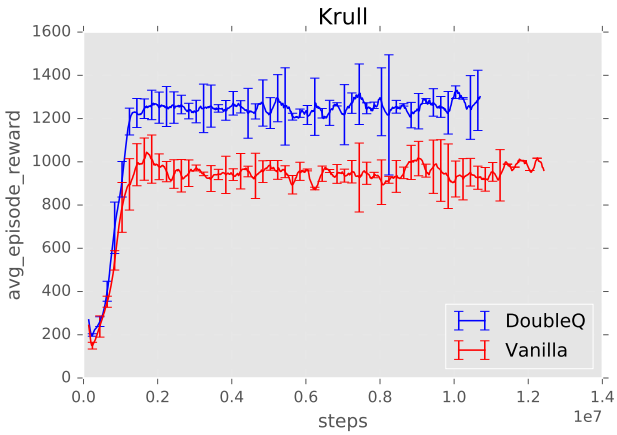


Figure 2: Average Episode reward with Double DQN for Krull

rate until a certain point, where Double DQN continues the slope of the learning rate and where the Vanilla DQN falls back in performance due it overoptimistic behaviour. The overoptimistic behaviour is visible in the average Q. We see that Vanilla DQN has a much higher Average Q value and Double DQN is able to estimate the Q value fairer. The distance between the two Average Q graphs, shows itself also in the graphs with the average episode rewards. For boxing is the distance rather small and do we see also a small distance between the average episode rewards, where this is the other way around for Krull.

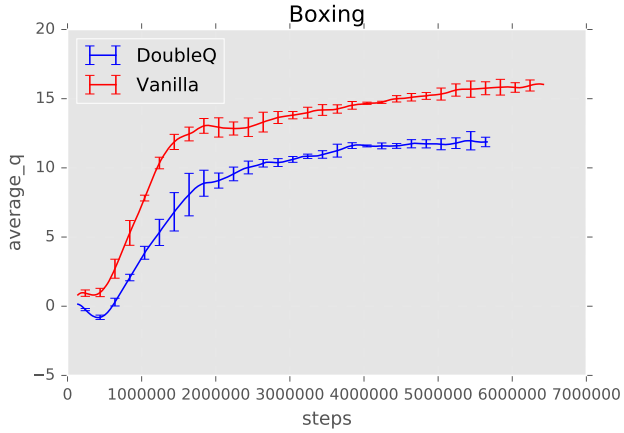


Figure 3: Average Episode reward with Double DQN for Boxing

All other results of this experiment can be found in appendix A

7.3 DQN with Prioritized Experience Replay

Secondly, we ran the DQN with Priortized Experience Replay to learn to play the two games. We ran it with the defined hyper-parameters in [Schaul et al., 2015]: $\alpha = 0.7$ and $\beta = 0.3$

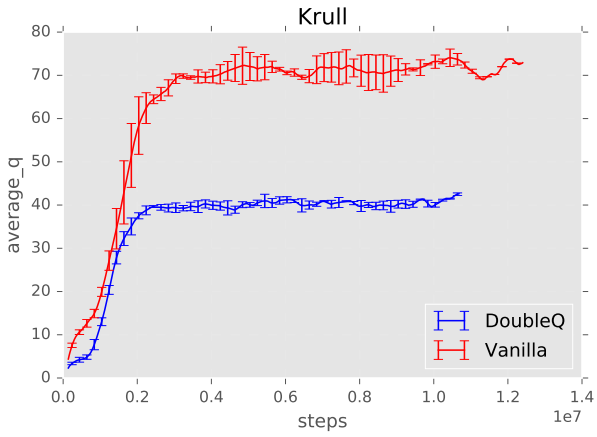


Figure 4: Average Episode reward with Double DQN for Krull

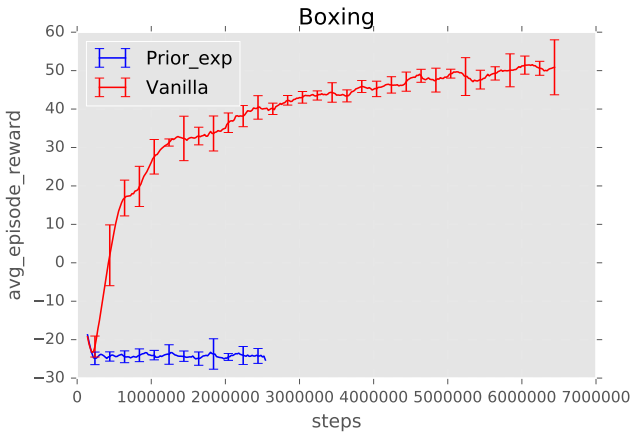


Figure 5: Average Episode reward with Prioritized Experience Replay for Boxing

As we can see in figure 7.3 and 7.3, is there a drastically difference with the results from Double DQN. It seems simple to learn nothing.Also the average Q shows dramatic bad results. In figure 7.3, we see that the average q value of prior experience goes in the opposite direction, compared to the baseline. The bad results are not environment dependant, since we see the same bad results for both games.

Our thoughts after seeing these results were a bug. We must have implemented something wrong. Although, after further research nothing seems really to be wrong. The Prioritized Experience Replay is in the same way implemented as the normal experience replay. The only difference is the implemented memory and how it selects is transitions. Our hypothesis of the bug was a bad selection distribution, which causes instability. Surprisingly, tests showed that the implemented distribution was just fine and behaves exactly as described. Figure 7.3 shows one of these results, where a histogram shows the requested indexes to the replay memory.

The complete results can be found in Appendix B

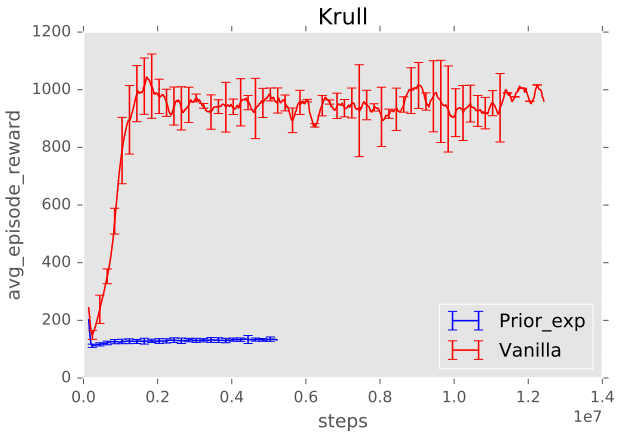


Figure 6: Average Episode reward with Prioritized Experience Replay for Krull

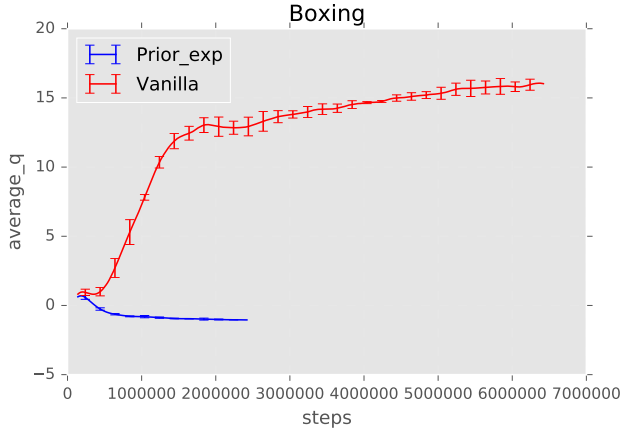


Figure 7: Average Q with Prioritized Experience Repla yfor Boxing

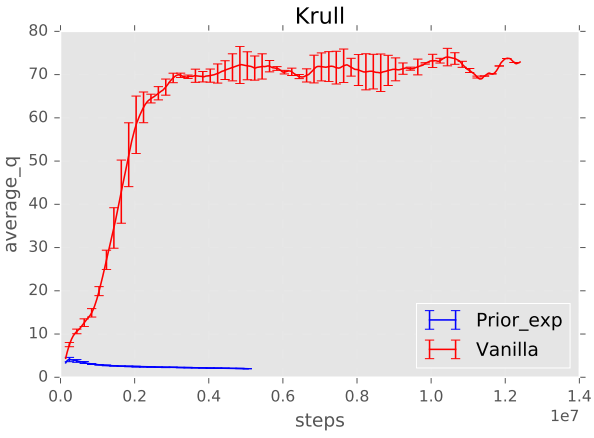


Figure 8: Average Episode reward with Prioritized Experience Replay for Krull

```
[4631443 4572162 4541147 4528613 4502237 4487485 4459655 4427915 4421711
 3970091      23      11      25      12      19      21      22      20
      15      16      13      16      18      11      13      11      17
      15       9      15       9      16       9      13      14      22
      18      22      11      13      12      14      23      15      16
      19      19      16      18      10      19      11      16      11
      18      14      17      22      14      13      15      15      17
      16      17      14      13      12      17      17      16      15
      15       9      15      14      13       9       7      17      15
      14      16       9      16      16      12      14      14      15
      14      13      19      10      12      15      22      15      19]
```

Figure 9: A Histogram of 100 bins with all the requested ranks of the heap. As we can see are all the requested indexes mostly for the value who ranked the higestes in the heap. Exactly the behaviour we want too.

7.4 Implementation Details

To speed up the working of Prioritized Experience Replay, we implemented a binary heap. This makes finding the maximum priority $O(1)$ and updating the priority values $O(log(n))$. The transitions get sorted in the binary heap based on their priority. The first transition in the heap gets initialised with one. When another transition is added to the memory is, it gets assigned the value of maximum priority currently in the heap, such that it will be replayed at least once. With an update, are the priorities of the replayed transitions incremented with the TD-error.

7.5 Empirical Results of Prioritized Double DQN

The last is experiment we conducted, is the combination of both adjustments, namely a Double DQN with Prioritized Experience Replay.

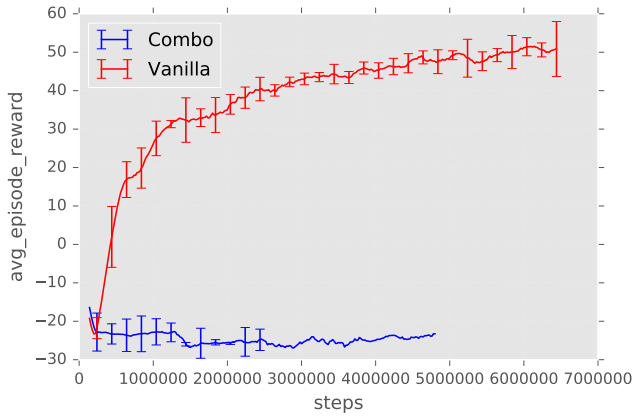


Figure 10: Average Episode reward with Double DQN and Prioritized Experience Replay for Boxing

We see the same behaviour as with Priortized Experience Replay alone. The positive effect of Double DQN, seems not to have an impact on the terrible performance of the Prioritized experience replay. It shows exactly the same behaviour.

The complete results can be found in appendix C

7.6 Side Note

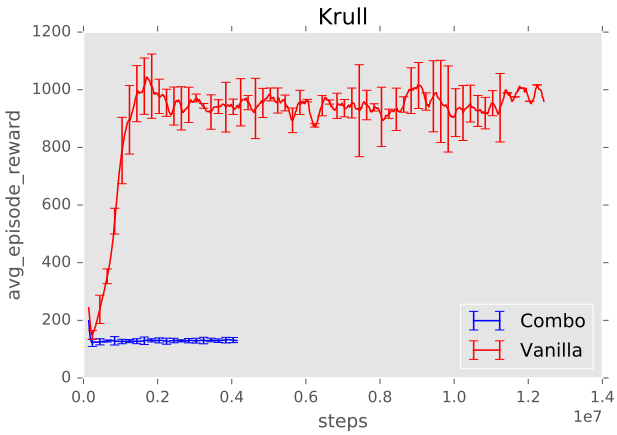


Figure 11: Average Episode reward with Double DQN and Prioritized Experience Replay for Krull

It was a real struggle to realise all the experiments. We know that the experimental setup is not ideal. 5 runs for each experiment and a low amount of total steps makes the results less accurate. The reason is the limited time frame, which caused a lot of trouble. The initial idea was to conduct the experiments with the hyper-parameters of the original paper by [Mnih et al., 2015]. Soon it became clear that an experiment with a memory-size of 1 million and 50 million steps, would not finish in time. As suggested, we tried to reduce the frame size to 42x42, but this resulted in almost no learning for several Atari games. Eventually after a lot of side-experiments, we ended up with the configuration suggested in 7.1.1 and the two Atari games Krull and Boxing. Both of these games are selected based on their fast learning rate and good performance in a limited configuration. The limited time frame for this project had also an impact on the amount of runs. Ideally we liked to conduct 10 runs or more for each experiment. This meant that we had to run 80 runs at the same time on the hydracluster. Unfortunately, this seemed to be not possible, because or the UserRAMLimit. Also balancing the runs and only assign 16gb to experiments that needed it, did not help. This resulted in several runs getting randomly killed after a few days. Eventually we succeed to create a stable experimental environment by doing only 5 runs per experiment.

8. DISCUSSION

Overall we can say that we are not completely satisfied with our results. Especially, the bad results of Prioritized Experience Replay disappoints us. Extensive research towards the behaviour of our implementation shows that it works as described in the original paper. This led to a big frustration. Especially since it is shown by [Schaul et al., 2015] to be possible to achieve better results. Of course do we also need to nuance our results, since we have evaluated them over a limited stepsize. Although, that the two games were specifically chosen because of their fast learning rate and significant improvement with these two techniques. To conclude we say that we implemented a prioritized experience replay that needs to be improved and at last we also successfully implemented and Double DQN network, which

showed successful results.

9. REFERENCES

[Hasselt, 2010] Hasselt, H. V. (2010). Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621.

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

[Schaul et al., 2015] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

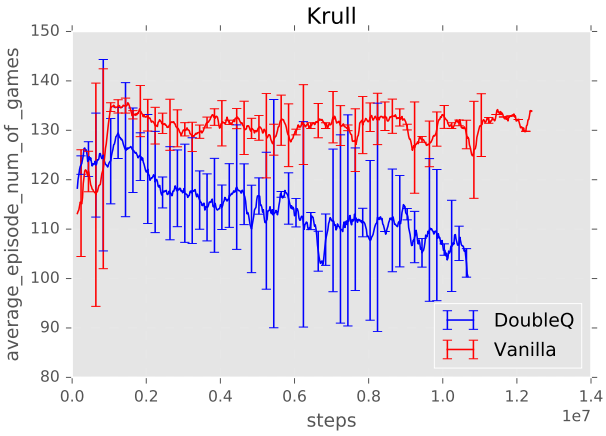
[Van Hasselt et al., 2016] Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100.

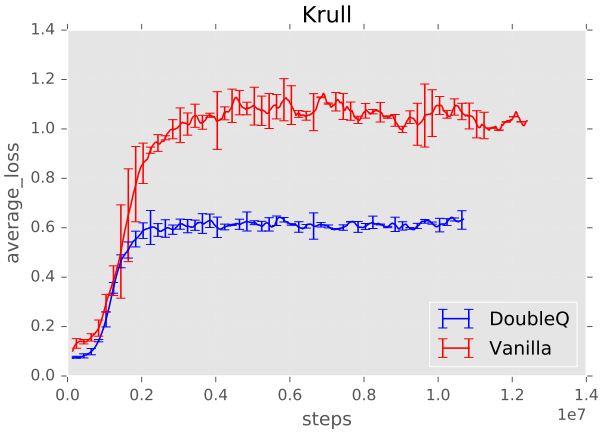
[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

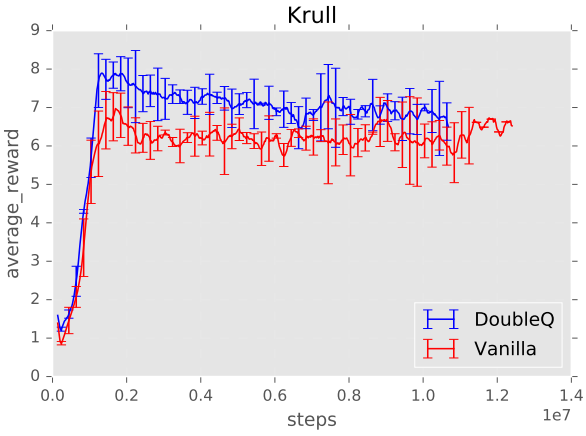
APPENDIX

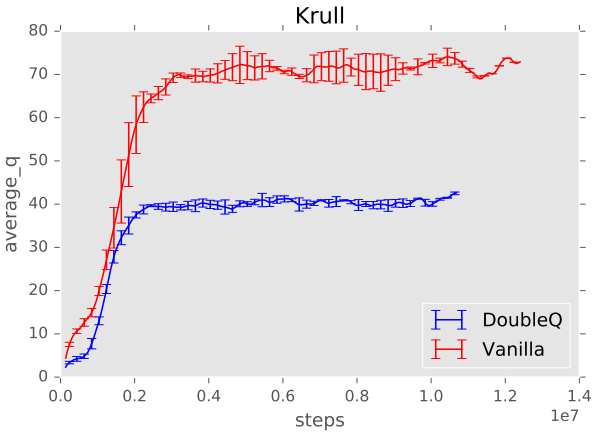
A. DOUBLE DQN

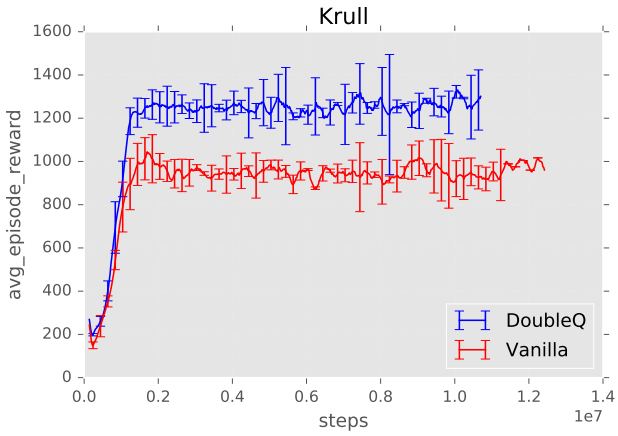
A.1 Krull

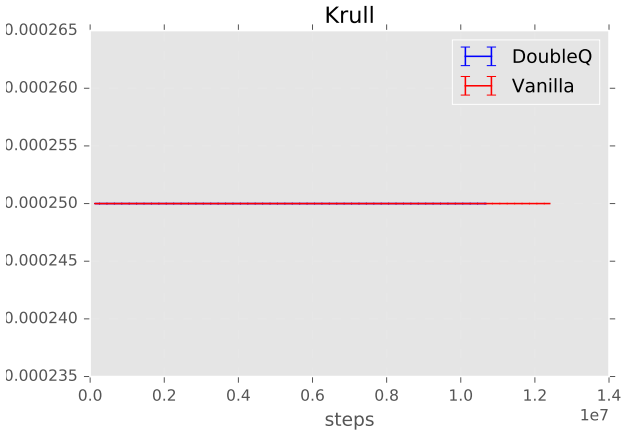


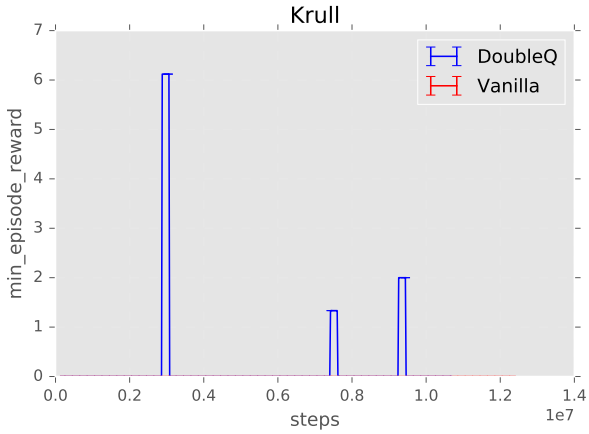


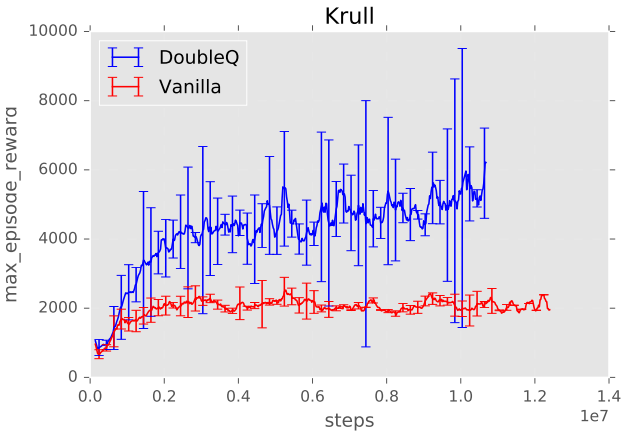




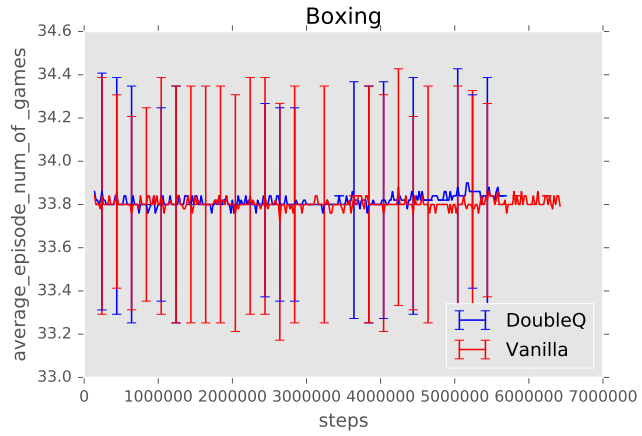

Krull
avg_episode_reward
steps
1e7
DoubleQ
Vanilla

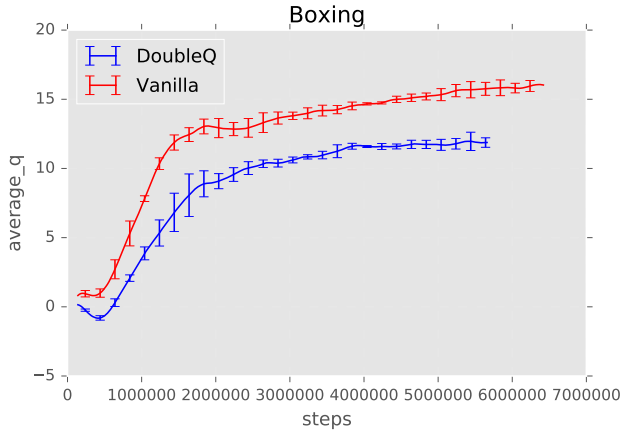

Krull
steps
1e7
DoubleQ
Vanilla

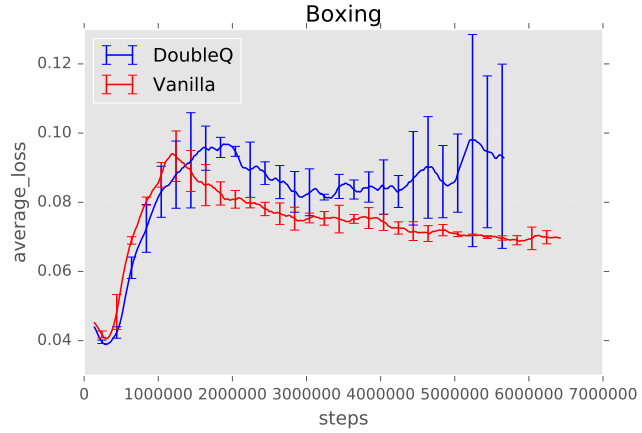

Krull
min_episode_reward
steps
1e7
DoubleQ
Vanilla

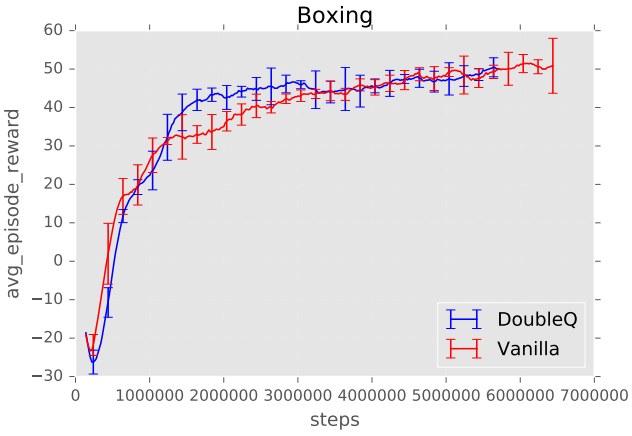

Krull
max_episode_reward
steps
1e7
DoubleQ
Vanilla

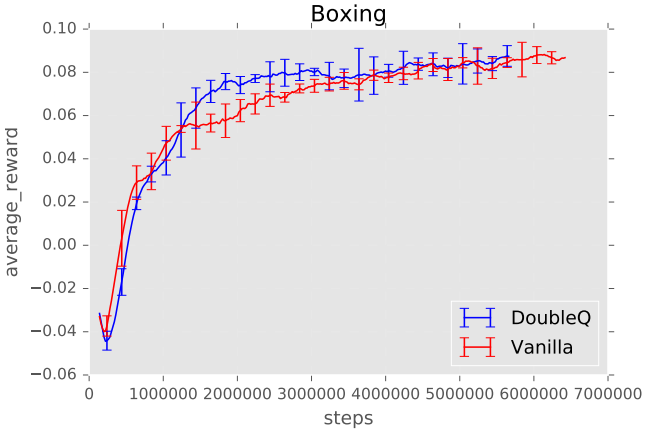
A.2 Boxing

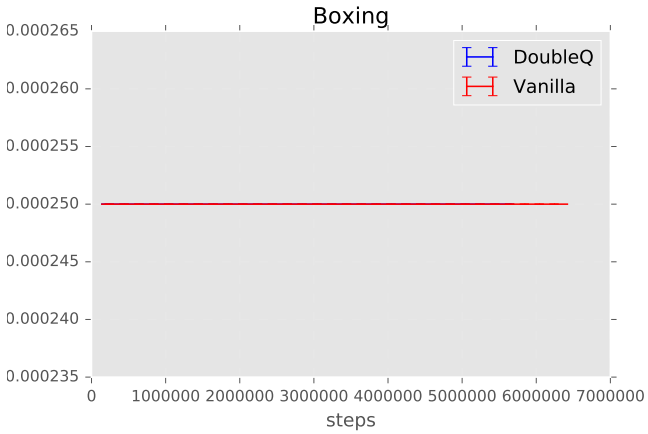

Boxing
average_episode_num_of_games
steps
DoubleQ
Vanilla

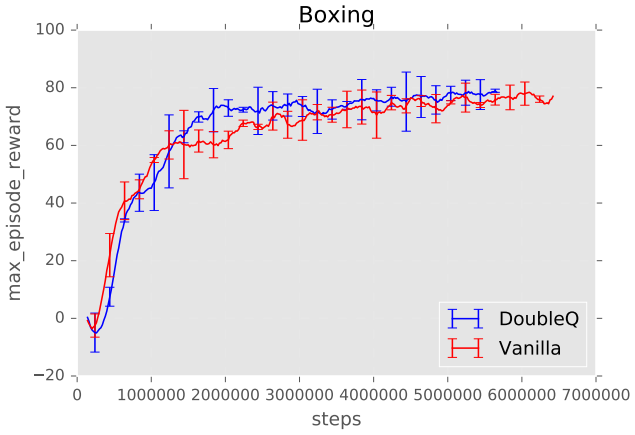

Boxing
average_q
steps
DoubleQ
Vanilla

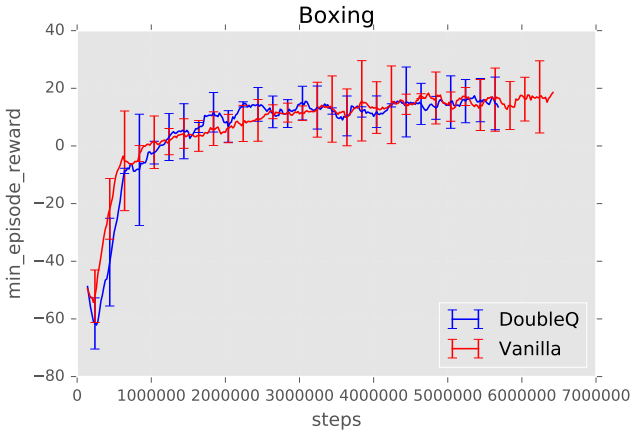

Boxing
average_loss
steps
DoubleQ
Vanilla

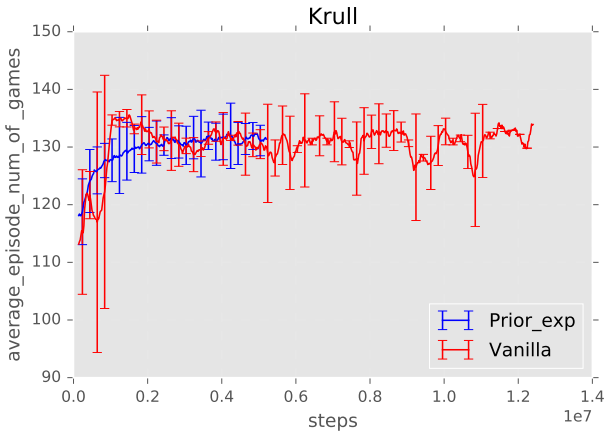

Boxing
avg_episode_reward
steps
DoubleQ
Vanilla

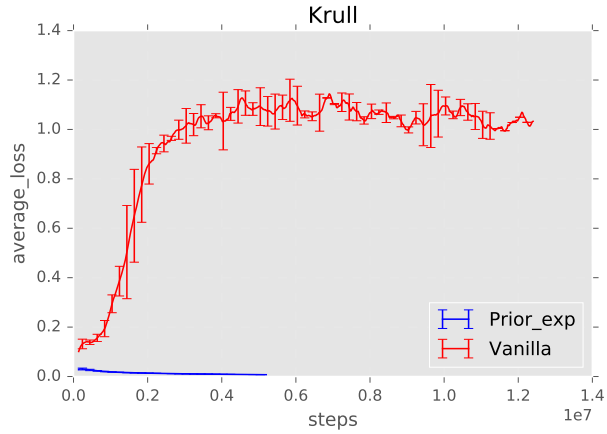

Boxing
average_reward
steps
DoubleQ
Vanilla

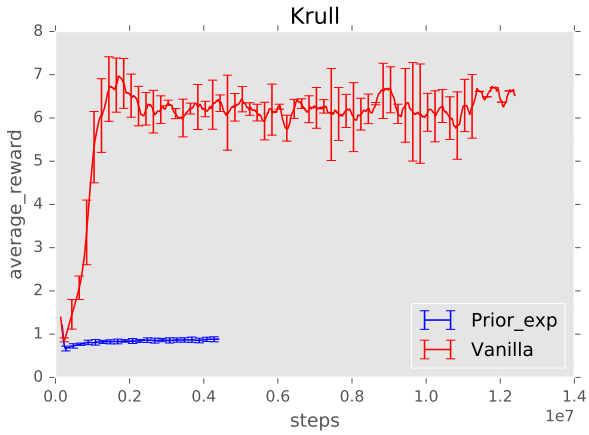

Boxing
steps
DoubleQ
Vanilla





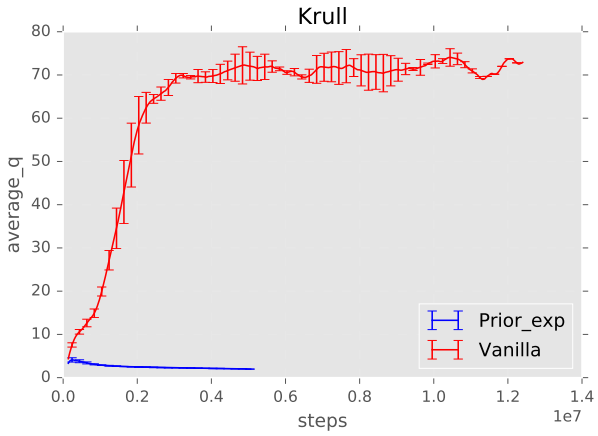


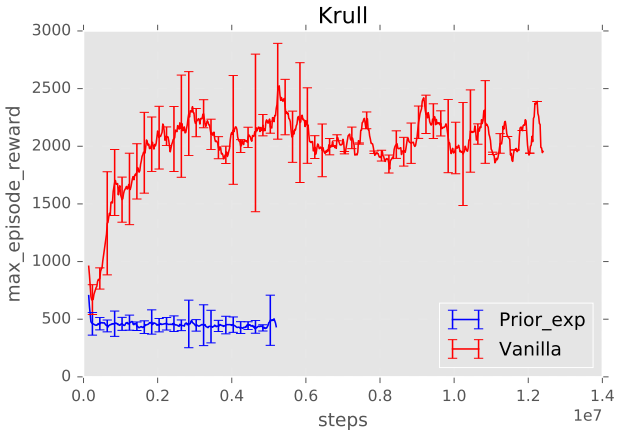


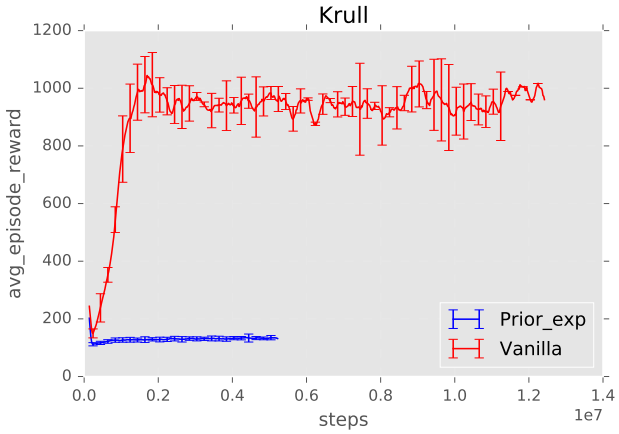


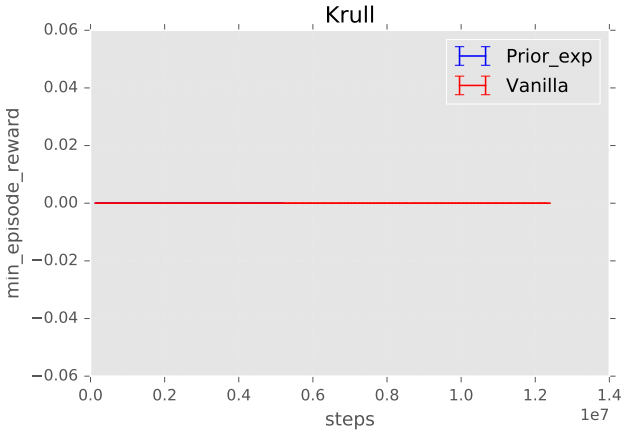
B. PRIORITY EXPERIENCE REPLAY

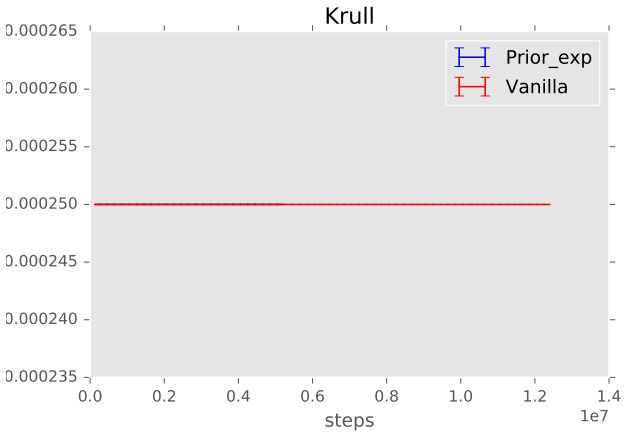
B.1 Krull


Krull
average_q
steps
1e7
Prior_exp
Vanilla

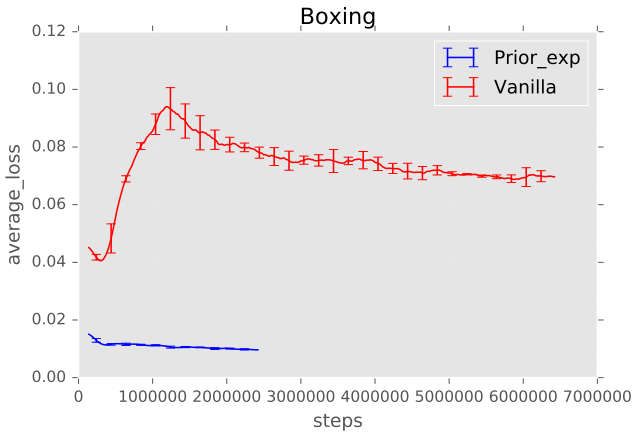

Krull
max_episode_reward
steps
1e7
Prior_exp
Vanilla

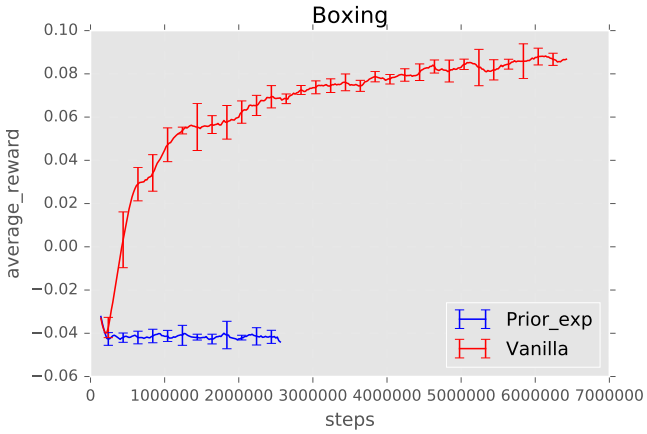

Krull
avg_episode_reward
steps
1e7
Prior_exp
Vanilla

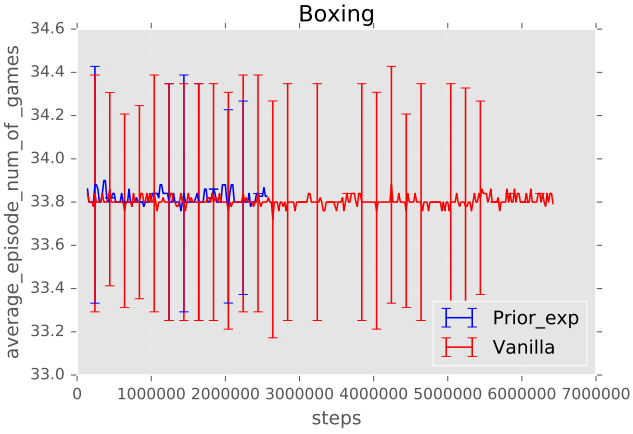

Krull
min_episode_reward
steps
1e7
Prior_exp
Vanilla

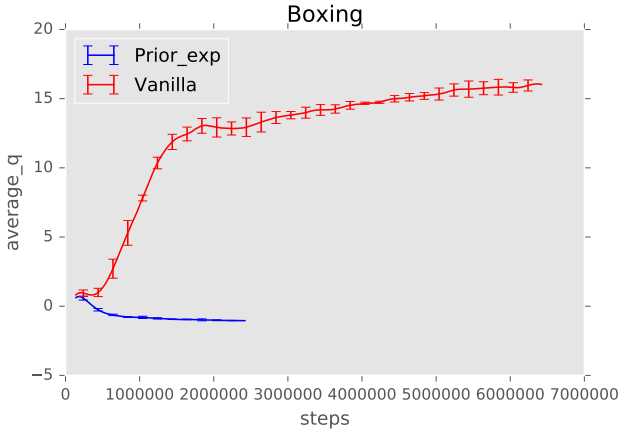

Krull
steps
1e7
Prior_exp
Vanilla

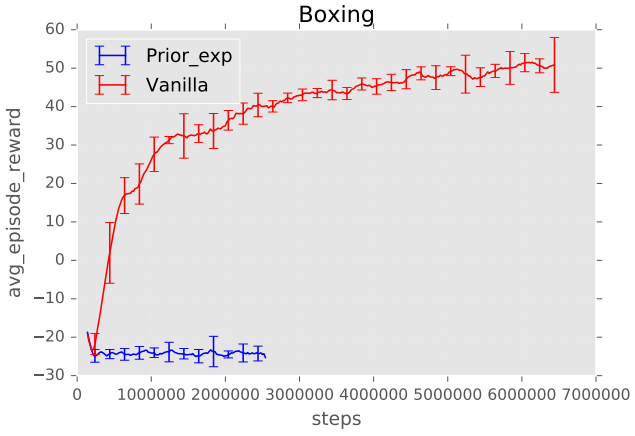
B.2 Boxing

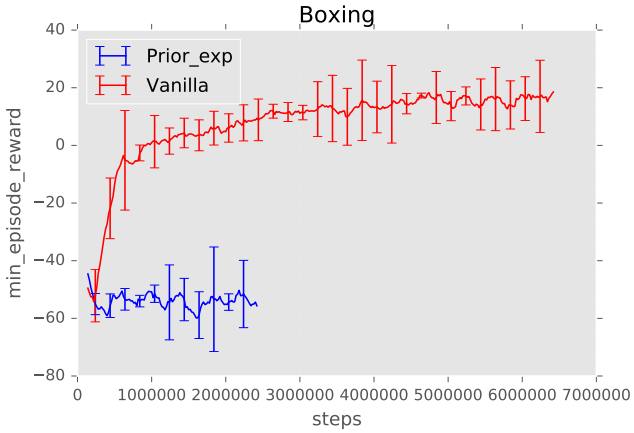

Boxing
Prior_exp
Vanilla
average_loss
steps
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


Boxing
Prior_exp
Vanilla
average_reward
steps
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


Boxing
Prior_exp
Vanilla
average_episode_num_of_games
steps
34.6
34.4
34.2
34.0
33.8
33.6
33.4
33.2
33.0
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000

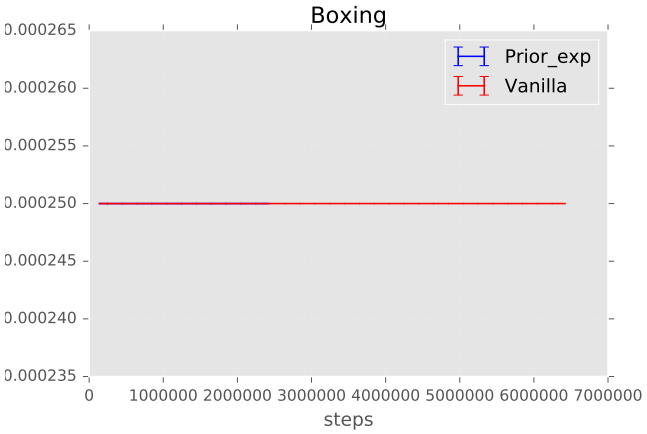

Boxing
Prior_exp
Vanilla
average_q
steps
20
15
10
5
0
−5
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000

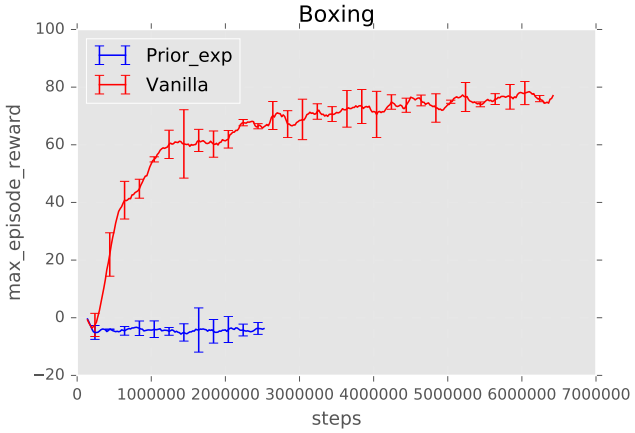


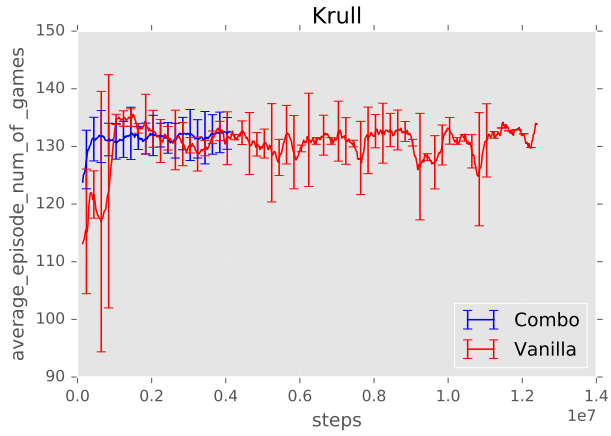


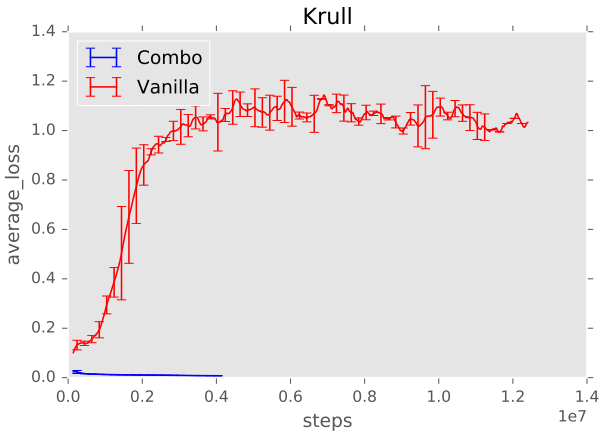
C. PRIORITY EXPERIENCE REPLAY WIHT DOUBLE DQN

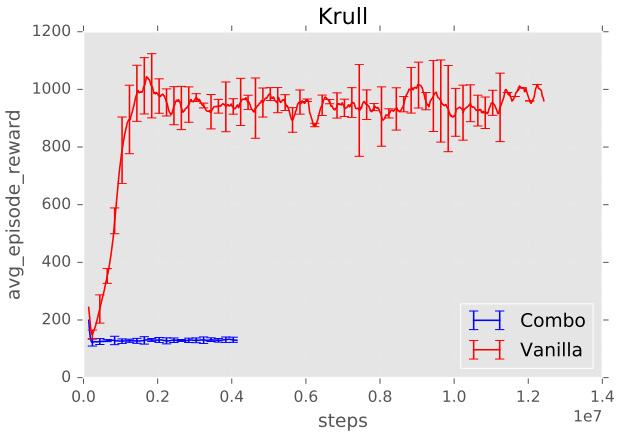
C.1 Krull

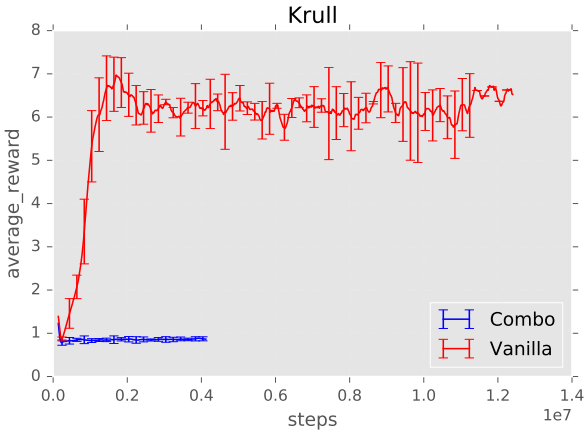


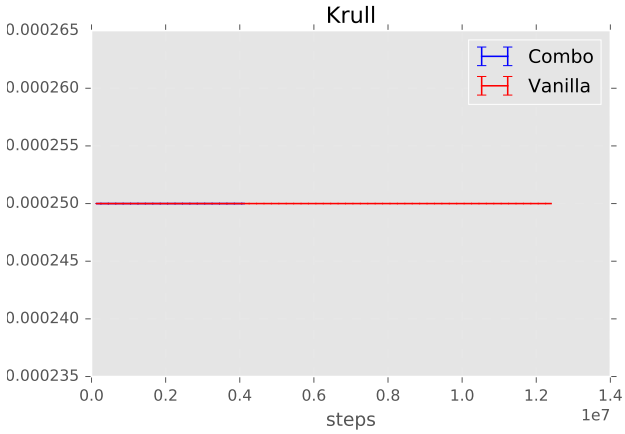


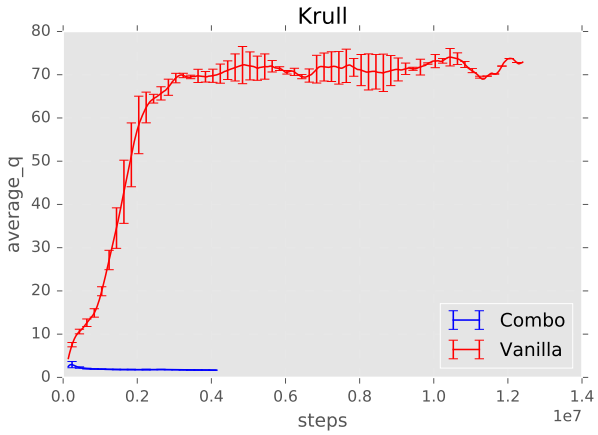


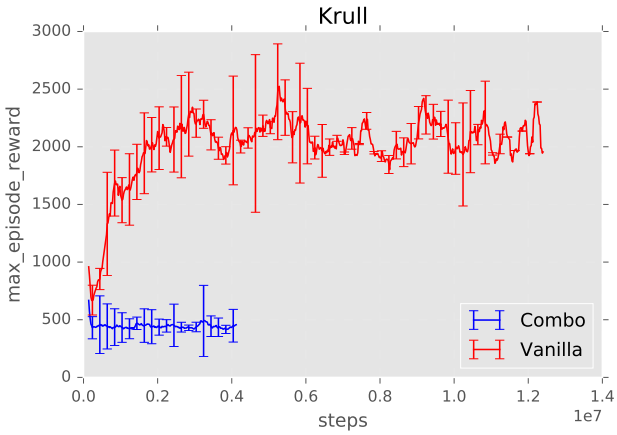

Krull
Combo
Vanilla
average_loss
steps
1e7

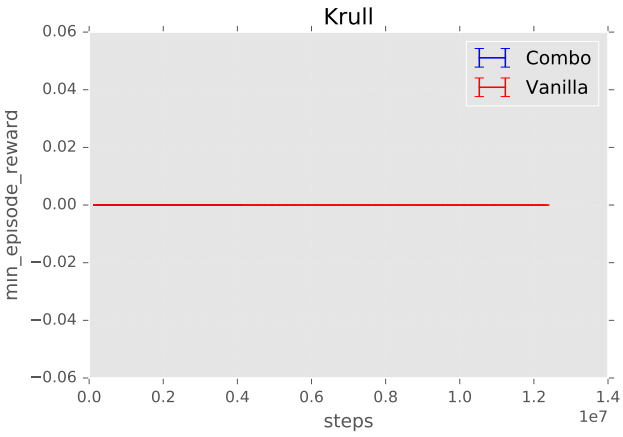

Krull
Combo
Vanilla
avg_episode_reward
steps
1e7

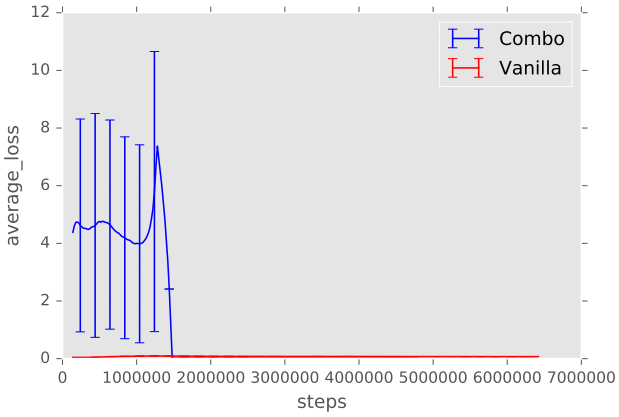

Krull
Combo
Vanilla
average_reward
steps
1e7


Krull
Combo
Vanilla
steps
1e7

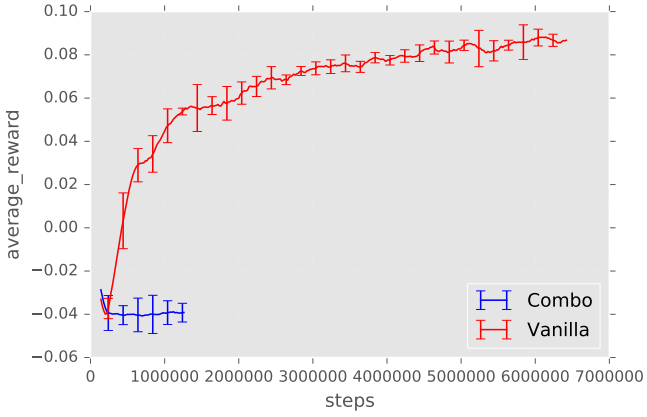

Krull
Combo
Vanilla
average_q
steps
1e7

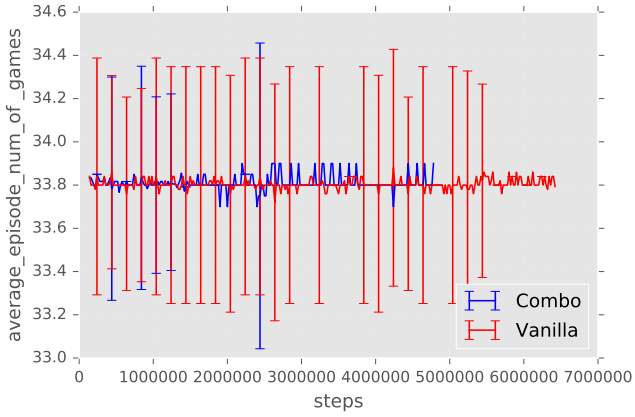

Krull
Combo
Vanilla
max_episode_reward
steps
1e7

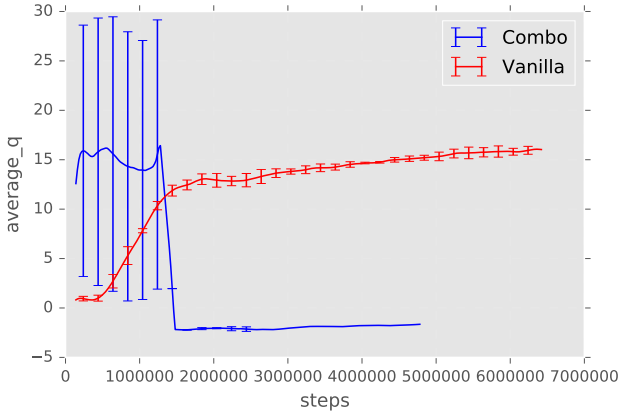

Krull
Combo
Vanilla
min_episode_reward
steps
1e7

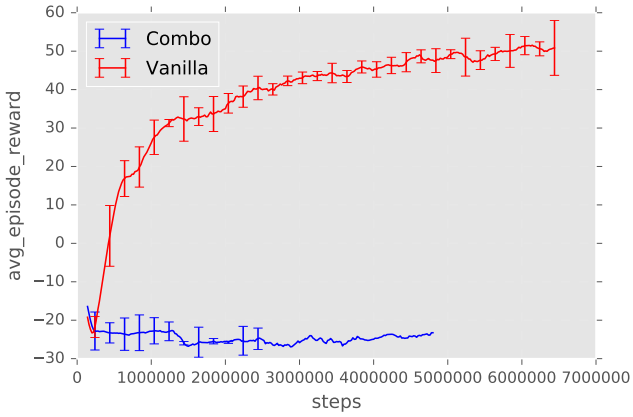

Combo
Vanilla
average_loss
steps

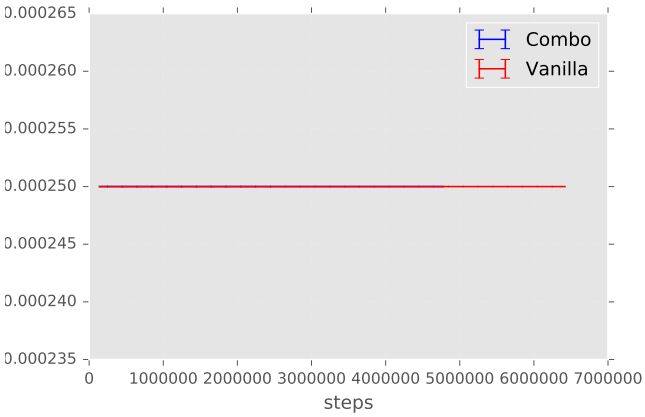
C.2 Boxing

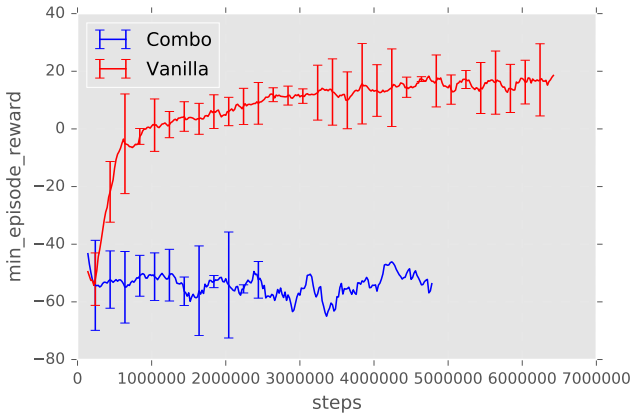

Combo
Vanilla
average_reward
steps

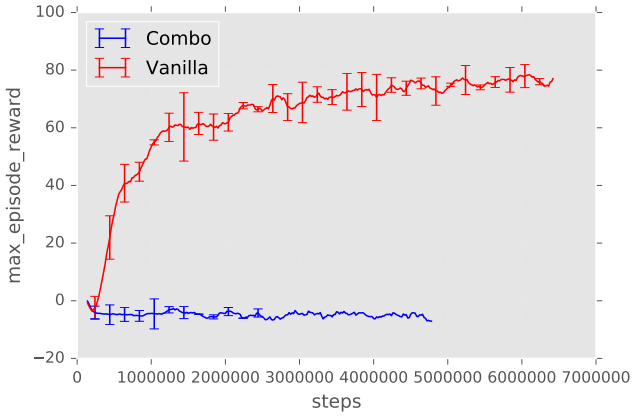

Combo
Vanilla
average_episode_num_of_games
steps


Combo
Vanilla
average_q
steps


Combo
Vanilla
avg_episode_reward
steps
60
50
40
30
20
10
0
−10
−20
−30
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


Combo
Vanilla
steps
0.000265
0.000260
0.000255
0.000250
0.000245
0.000240
0.000235
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


Combo
Vanilla
min_episode_reward
steps
40
20
0
−20
−40
−60
−80
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000


Combo
Vanilla
max_episode_reward
steps
100
80
60
40
20
0
−20
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000