DDQN: Metrics for measuring stability using the example of replay buffer size and minibatch size

Anja Witte anja.witte@haw-hamburg.de
Department Computer Science
Hamburg University of Applied Sciences
Hamburg, Germany

Abstract

The Reinforcement Learning algorithm Double Deep Q-Network (DDQN) is known to have an unstable training process (Halat and Ebadzadeh, 2021). In order to overcome instability, this paper aims to deepen the understanding of stability and measuring it. Therefore, numerical indicators are proposed to determine convergence and experiment stability. Additionally, the metrics are investigated for Cartpole by adapting the replay buffer size and minibatch size. Experimental results show that the minibatch size has a higher impact on stability. Best stabilities are achieved with the lowest minibatch size of 10. This setup leads to an up to 4.6 % higher stability.

Keywords: DDQN, stability metrics, convergence stability, experiment stability, replay buffer size, minibatch size

1. Introduction

Reinforcement Learning (RL) methods are increasingly used in various application areas, e. g. robotics (Kober et al., 2013), games or natural language processing (Li, 2018). Nevertheless, RL algorithms as Double Deep Q-Network (DDQN) have an unstable training process which leads to forgetting. As stability is a key factor for success, this issue has to be addressed (Halat and Ebadzadeh, 2021). Before solving the instability, it is necessary to fully understand the topic in order to recognize e. g. hyperparameter dependent instabilities. Therefore, this paper introduces numerical metrics for stability of DDQN that support the evaluation and comparability of different experiment setups. Additionally, the metrics will be determined and analyzed for different experimental setups to solve Cartpole with DDQN. The experiments investigate exemplary the adaption of the hyperparameters replay buffer size and minibatch size as Liu and Zou (2017) already showed their impact on the learning of an agent. Moreover, Zhang and Sutton (2017) found the replay buffer size to be an important but underestimated hyperparameter for performance.

The paper is structured as follows. Section 2 discusses related work in the context of the goal of this paper. Section 3 then explains the basics of DDQN that are needed as foundation for the paper and the metrics that are introduced in section 4. Here, a distinction is made between convergence and experiment stability. Using the metrics, section 5 analyzes the stability of different setups used to solve CartPole with DDQN. It follows a discussion of the results and the metrics in section 6. Finally, section 7 draws a conclusion.

2. Related Work

The reinforcement learning algorithm DDQN leads to instability in the training process of e. g. CartPole (Halat and Ebadzadeh, 2021). Some paper address this challenge that an agent suddenly forgets things already learned. Buşoniu et al. (2018) focus on reinforcement learning algorithms for control and find that stability is an open research topic. The authors differ between stability in terms of robustness against e. g. noise or disturbances and in terms of convergence of the training process. For the purpose of this paper, stability includes on the one hand the convergence process and on the other hand the similarity of the training processes between multiple runs with the same hyperparameter but different seeds for the neural networks. The two characteristics will be referred as convergence stability and experiment stability.

Moreover, Halat and Ebadzadeh (2021) propose some modifications in the target value function of DDQN in order to overcome the convergence instability. Experiment stability is investigated through averaging scores for each episode over multiple runs. As the final convergence comparisons are based on plotting average scores, it is a visual approach. In contrast, this paper proposes primarily numerical indicators for convergence and experiment stability. Both can be plotted as well.

In terms of benchmarking new or modified algorithms, other papers have a similar visual approach as mentioned above. In addition to the average return for each step Fujimoto et al. (2019) plot the standard deviation to compare different batch deep reinforcement learning algorithms. Kumar et al. (2019) and Anschel et al. (2017) do a similar comparison for evaluating their approaches to reduce instability. Instead of the average, Van Hasselt et al. (2016) and Fedus et al. (2020) analyze the median over different runs and the quantiles.

Furthermore, Explainable Reinforcement Learning (XRL) (Heuillet et al., 2020) is a related topic as this paper aims to deepen the understanding. The idea of XRL is to make RL algorithms more transparent and comprehensible in order to increase the understanding and the trust of users (Heuillet et al., 2020). Wang et al. (2019) approach this by proposing a dashboard to give insights into the processes of Deep Q-Networks (DQNs). Here, one part is the statistics view that aims to summarize the training statistics. With information as e. g. the average rewards per episode, the user should be able to understand stability among others. Numerical indicators for experiments and convergence stability could improve XRL through a better description than simply averaging the reward over multiple runs.

3. Double Deep Q-Networks

RL is a machine learning paradigm. In comparison to supervised and unsupervised learning, agents aim to learn goal-directed actions to maximize a numerical reward signal. For this purpose, agents explore the environment through interaction. This means that in a current state s, an agent chooses an action $a \epsilon \mathcal{A}(s)$ according to a policy π[1] that results in a following state s'. Moreover, the agent receives a reward r. In order to maximize the reward signal, an agent has to balance *exploration* and *exploitation*. Meaning that the selection of not yet

1. A distinction is made between deterministic and stochastic policies. A deterministic policy $\pi(s)$ chooses an action based on a given state s. In comparison, a stochastic policy $\pi(a|s)$ returns a probability for an action to be taken given state s.

taken actions (exploration) as well as actions experienced as useful in the past (exploitation) are needed to achieve a goal (Sutton and Barto, 2018, P. 1-6).

The basic concept of DDQN is Q-Learning which is considered as an off-policy method. Meaning that the goal is to learn a *target policy* π while the agent is following another policy b called *behavior policy*. The policy π is typically deterministic while b has to be stochastic (e. g. ϵ-greedy). Expected returns sampled from one policy can be estimated for another policy by *importance sampling*. It weights returns according to the relative probability (Sutton and Barto, 2018, P. 103-104).

Q-Learning enables this by learning the Q-value which represents the expected discounted rewards for an agent taking action a in state s and acting optimally afterwards. Action-values are determined by

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma * max_{a'}Q(s',a') - Q(s,a)] \tag{1}$$

where γ is the discount factor which limits future rewards and α is the step size. With this, the optimal action-value function q_* can be approximated (Russell and Norvig, 2021).

DQN extends the concept of Q-Learning by the integration of deep neural networks as a nonlinear function approximator. This expands the applicability of RL to domains where the state spaces are high-dimensional as deep convolutional neural networks approximate the optimal action value. Including the weights θ, the network is called a *Q-network*. Training process therefore comprises the adaption of θ_i in iteration i. The network weights impact the approximate target values which are determined based on the parameters θ_i^- from the previous iteration:

$$Y_t^{DQN} \equiv R_{t+1} + \gamma * max_a Q(S_{t+1}, a; \theta_i^-) \tag{2}$$

In order to increase stability of the algorithm, the trained Q-network Q is used for multiple steps as a *target network* $\hat{Q}$ to generate targets and is updated by Q every C steps. Moreover, experience replay (Lin, 1992) is applied. The idea is to store all experiences of an agent over multiple episodes in a replay memory $D_t = \{e_1, ..., e_t\}$. At each step, an experience is presented as $e_t = (s_t, a_t, r_t, s_{t+1})$. In general, D has a fixed size and stores the last N steps. For each time-step, Q is trained on a minibatch of the replay memory where a minibatch consists of random samples of the stored experiences. This lead to less correlation of the samples. As a result, DQN outperforms previous RL algorithms (Mnih et al., 2015).

Following the approach explained above, actions are selected and evaluated by the same DQN which leads to overestimation of action values. Therefore, Van Hasselt et al. (2016) propose to separate between both by using different networks. The target network with the weights θ_t^- is used to evaluate the current policy while action selection is performed based on the network with weights θ_t. Equation 3 presents the resulting target. Similar to DQN, the target network is updated regularly.

$$Y_t^{DoubleDQN} \equiv R_{t+1} + \gamma * Q(S_{t+1}, argmax_a Q(S_{t+1}, a; \theta_i), \theta_i^-) \tag{3}$$

4. Stability Metrics

For the purpose of this paper, a distinction is made between convergence and experiment stability. Both are explained in the following and numerical indicators are introduced.

4.1 Convergence Stability

Convergence stability refers to the learning curve over multiple episodes. The optimal learning process has an upward trend meaning that an agent is increasing the reward signal continuously and converging the maximum. In case the learning process is monotonically increasing, it is considered as optimal convergence stability. Every decrease in the reward signal contradicts the monotonicity condition and is counted as *instability*. To measure the strength of instability, the $\Delta return$ of all decreases is summed up and called *actual instability* i_{actual}. It is defined in equation 4 where n is the number of episodes. The convergence stability $s_{convergence}$ is then defined as the actual instability relatively to the *maximum instability* i_{max} that can be reached in an experiment. The maximum instability depends on the use case. E. g. for Cartpole it can be calculated supposing that reaching the maximum return R_{max} and minimum return R_{min} alternate for episodes. The calculation follows equation 5 where $n_{episodes}$ is the number of episodes.

$$i_{actual} = \sum_{k=2}^{n} (R_{k-1} - R_k), \qquad (R_k < R_{k-1}) \tag{4}$$

$$i_{max} \approx \lfloor \frac{n_{episodes}}{2} \rfloor * (R_{max} - R_{min}) \tag{5}$$

Based on the actual instability i_{actual} and the maximum instability i_{max}, the convergence stability can be determined with

$$s_{convergence} = 1 - \frac{i_{actual}}{i_{max}}. \tag{6}$$

where $s_{convergence} \in [0, 1]$. Here, a monotonically increasing learning curve (optimal convergence stability) is reached with $s_{convergence} = 1$ whereas $s_{convergence} = 0$ represents the minimal stability.

Figure 1 shows a learning curve exemplary for a CartPole experiment. Blue lines highlight an upward trend that meet the monotonicity condition. In comparison to that, orange lines are decreases in the return which represent instability. For $n_{episodes} = 50$ the maximum instability is $i_{max} = (50/2) * (200 - 0) = 5000$. With i_{actual} and i_{max} convergence stability can be calculated. For this, all instabilities of an experiment i_{actual} are summed up and divided by the maximum instability i_{max}. It can be determined as in equation 6. For the example in figure 1 the instability is $i_{actual} = 485$. Calculating the convergence stability results in $s_{convergence} = 0.903$.

As convergence stability can differ for the same experimental setup but different seeds for the neural network, the convergence stability can be averaged for multiple runs.

4.2 Experiment Stability

In comparison to the convergence stability, the experiment stability aims to measure the similarity of the learning curves of multiple experiments with the same hyperparameters.

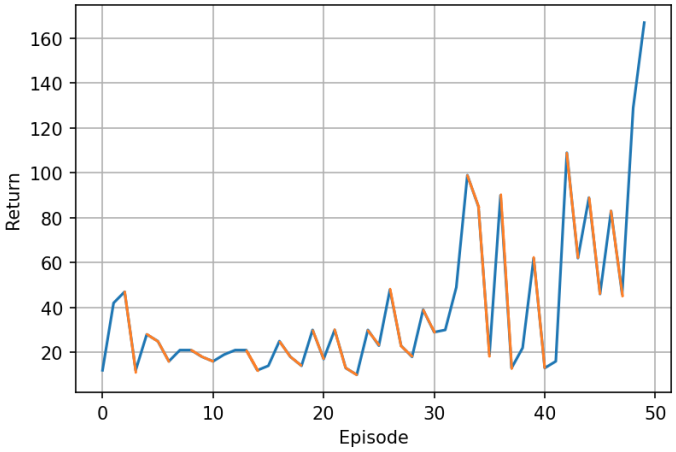

Figure 1: Learning curve for CartPole

Theoretically, learning curves are expected to be the same when running an experiment multiple times. Nevertheless, it differs in reality depending on the seed for the neural network. Figure 2 shows the learning curves for four experiments with the same setup for Cartpole. From a visual perspective, experiment 1 and 2 are more different than experiment 3 and 4. Experiment stability covers theses differences.

To measure the differences in the learning curves, each curve is taken as a set of points where each point represents the reached return in a specific episode. This enables the comparison of each episode over multiples experiments separately. Mathematically, similarity of two points x, y that are interval scaled can be determined by $1 - \frac{d - min_d}{max_d - min_d}$ where $d = |x - y|$ is the city block as the usecase has $n = 1$ dimensions (Tan et al., 2019, P. 95-96). This measure is used for a pairwise comparison of the experiments. For each episode e, the total distance $d_{total,e}$ comprises the distances of all pairwise comparisons. It can be determined with

$$d_{total,e} = \sum_{k=1}^{n} \sum_{l=2}^{n-1} |R_k - R_l|, \qquad (k < l). \tag{7}$$

Similar to the convergence stability, the experiment stability $s_{experiment}$ is measured relatively to the maximum distance d_{max} that can be reached between two experiments. The maximum distance is achieved when both experiments have a distance of $(r_{max} - r_{min})$ for all episodes $n_{episodes}$ and is therefore calculated as follows:

$$d_{max} = n_{episodes} * (R_{max} - R_{min}) \tag{8}$$

Finally, the experiment stability $s_{experiment}$ is based on the distances $d_{total,e}$ and d_{max} for all episodes n:

$$s_{experiment} = 1 - \frac{\sum_{e=1}^{n} d_{total,e}}{d_{max}} \tag{9}$$

where $s_{experiment} \in [0, 1]$. A result of $s_{experiment} = 1$ shows perfect repeatability of an algorithm as each experiment lead to the same result. In comparison, $s_{experiment} = 0$ means maximum different results.

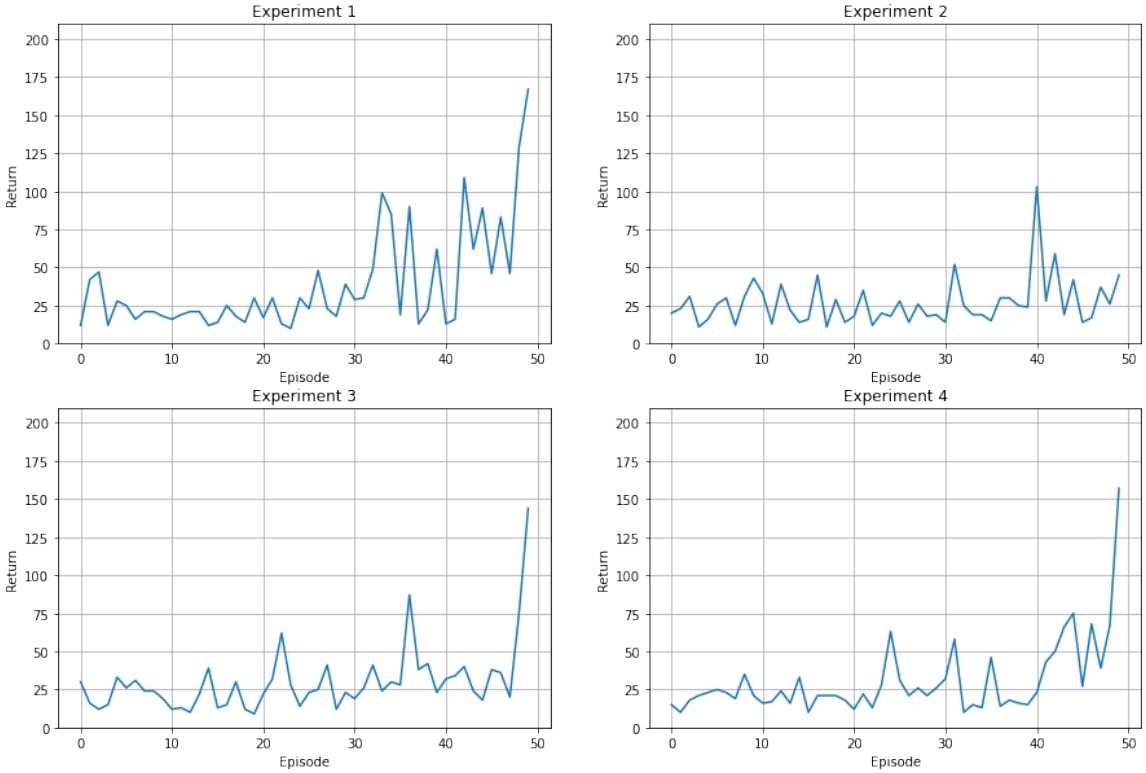

Figure 2: Learning curves for different experiments

For the example shown in figure 2 the experiment stability is calculated exemplary. Experiment 1 and 2 have a total distance of $d_{total,1\&2} = 1213$ whereas the total distance for experiment 3 and 4 is $d_{total,3\&4} = 754$. This leads to the experiment stabilities $s_{experiment,1\&2} = 1 - \frac{1213}{50*200} = 0.8787$ and $s_{experiment,3\&4} = 1 - \frac{754}{50*200} = 0.9246$. The numerical indicator for experiment stability verifies that experiments 3 and 4 lead to more similar learning curves and experiments.

5. Experiments

This section includes multiple experiments in order to showcase the convergence and experiment stability introduced in section 4. On the one hand, the experimental setup is explained and on the other hand, experimental results are shown in the following.

5.1 Experimental setup

For all experiments, the DDQN algorithm implemented with Python and Keras (Chollet et al., 2015) is used to solve the CartPole-v0 problem described by Barto et al. (1983) and implemented in OpenAI Gym (Brockman et al., 2016). Within CartPole, the goal is to keep a pendulum that is attached to a cart upright. An exemplary upright CartPole is shown in figure 3. The corresponding action space is $[0, 1]$ that pushes the cart to the left or right. Moreover, the observation space comprises cart position ($[-4.8, 4.8]$), cart velocity

($[-\infty, \infty]$), pole angle ($[-24°, 24°]$) and pole angular position ($[-\infty, \infty]$). The received rewards depend on the achieved upright time steps where one time steps corresponds to a reward of 1. For the termination of an episode, one of the following conditions has to be fulfilled: (1) pole angle $\pm 12°$, (2) cart position ± 2.4, (3) number of episodes > 200. Condition (3) leads to a maximum total return of 200 for each episode (Brockman et al., 2016).

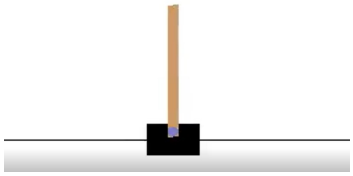

Figure 3: Upright CartPole System

As the goal of the paper is to measure the convergence and experiment stability, each of the following experimental setups is performed ten times with 50 episodes. An overview of all hyperparameters for the experiments is shown in table 1.

Table 1: Hyperparameter for CartPole experiments

Hyperparameter	Value
episodes	50
discount factor γ	0.99
experience replay buffer size	[50, 100,.., 950]
mini batch size t	[10, 50, 100, 150]
number of layer	3
layer size	16
activation function	relu, relu, linear
optimizer	Adam
learning rate	0.001
loss function	mean squared error

For DDQN, a DQN and a target-DQN are needed to predict Q-values. Both have the same architecture and hyperparameter. The target-DQN is initialized with the weights of DQN and every tenth episode updated accordingly. The used network architecture is shown in figure 4. It comprises an input layer which takes the observation space of CartPole (cart position, cart velocity, pole angle, pole angular position) as input. Moreover, two hidden fully connected layer with 16 neurons and an output layer with two neurons (according to the action space) are included. The hidden layer use relu as activation function whereas the output layer uses linear activation. In each step, the model is trained for one epoch where Adam is chosen as optimizer with the Keras default learning rate of 0.001. The used loss function is mean squared error.

As explained in section 3, the used DDQN implements a replay memory D_t where t is the fixed memory size and $t \in [50, 100, ..., 950]$. In the beginning, the replay memory

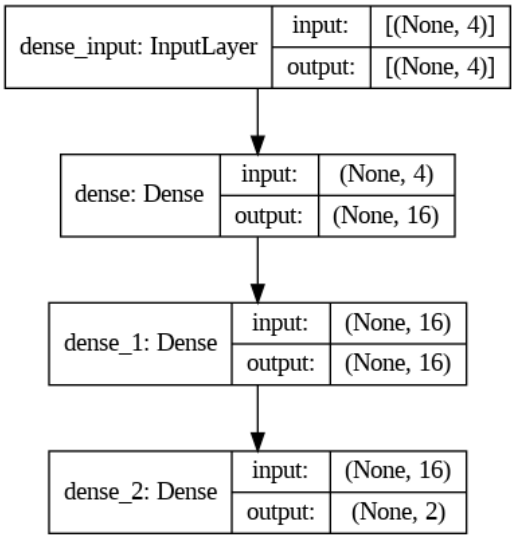

Figure 4: Network architecture used for DDQN

is initialized with experiences that followed the ϵ-greedy policy. After each time-step, a new experience is added while the oldest experience in the replay memory is dropped. Additionally, a random minibatch is taken from the replay memory to create training data. The size of the minibatch is besides the replay memory size the second hyperparameter which is tested with different values in the range of $[10, 50, 100, 150]$ in the experiments.

In order to ensure the balance of exploration and exploitation in the training process over episodes, the Boltzmann exploration policy (Cesa-Bianchi et al., 2017) is used. Here, action probabilities depend on Q-values and softmax is used to determine the distribution over the actions. The temperature $\theta \in [0.1, 10.0]$ controls the exploration of the policy.

5.2 Experimental results

As mentioned in the experimental setup, convergence and experiment stability are analyzed based on the hyperparameter minibatch size and replay buffer size. All of the following result figures show on the one hand the stability averaged over ten experiments and on the other hand the standard deviation.

5.2.1 Convergence Stability

Figure 5 shows the convergence stability curve depending on the minibatch size. It reaches its maximum $s_{convergence} = 0.9545$ with the lowest minibatch size of 10. With increasing minibatch size, the convergence stability is increasing to the global minimum of $s_{convergence} = 0.9211$ with a minibatch size of 150. Additionally, the standard deviation is increasing from 0.0141 for a minibatch size = 10 to 0.0269 for a minibatch size = 150. In average the standard deviation is 0.0183.

The results of analyzing the convergence stability for a changing replay buffer size is shown in figure 6. Here, two different values for the minibatch size are presented. On the left hand figure, the minibatch size = 10 leads to a convergence stability between 0.934

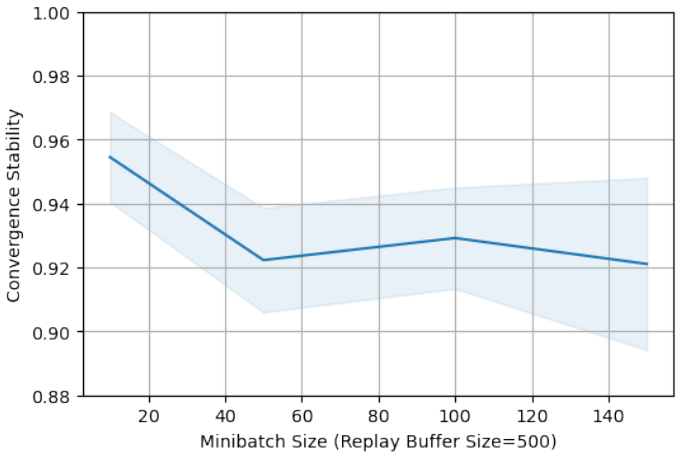

Figure 5: Convergence Stability for different minibatch sizes (The dark blue line represents the average whereas the light blue filled area represents the standard deviation)

and 0.9573. The highest stability is found with replay buffer size = 50 whereas the global minimum is reached with replay buffer size = 900. The standard deviation ranges from 0.0101 to 0.0236. In comparison to that, a minibatch size = 50 decreases the convergence stability curve for a changing replay buffer size. This curve is presented on the right hand image of figure 6. The convergence stability achieves a global maximum at replay buffer size = 200 with $s_{convergence} = 0.9373$. The global minimum is reached with $s_{convergence} = 0.9146$ at a replay buffer size of 650. The convergence stability has a standard deviation between 0.0109 and 0.0279. The average standard deviation is 0.0142 compared to 0.0177 for a minibatch size of 50.

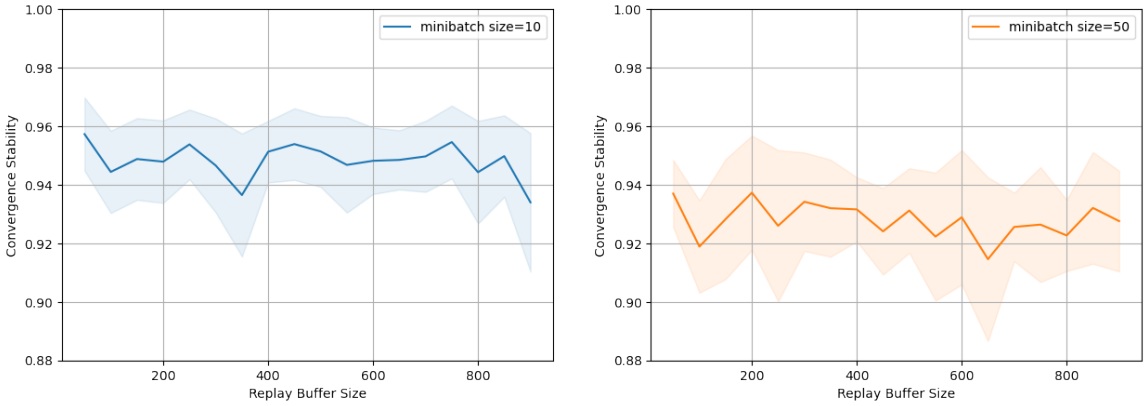

Figure 6: Convergence Stability for different replay buffer sizes and minibatch sizes (The dark blue/orange line represents the average whereas the light blue/orange filled area represents the standard deviation)

5.2.2 Experiment Stability

The experiment stability is analyzed similarly to the convergence stability. Figure 7 presents the results for an increasing minibatch size and a fixed replay buffer size = 500. The experiment stability is monotonously decreasing from the maximum $s_{experiment} = 0.949$ to the minimum $s_{experiment} = 0.9047$. The standard deviation reaches the highest value with 0.0205 at a minibatch size of 50, The lowest standard deviation is found at a minibatch size = 150 with 0.012. The average for the standard deviation for the convergence stability is 0.0162.

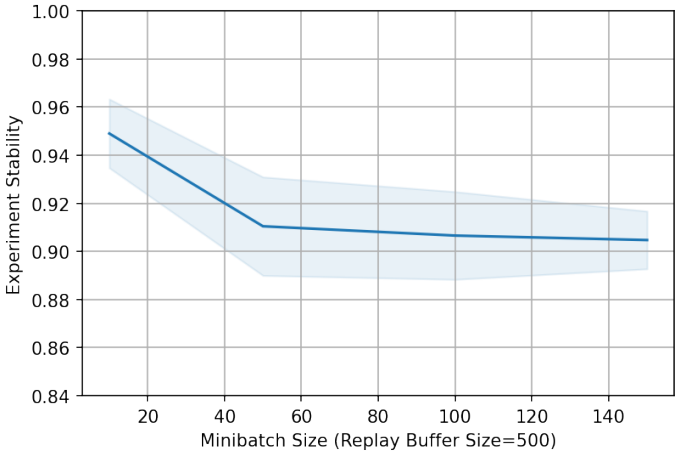

Figure 7: Experiment Stability for different minibatch sizes (The dark blue line represents the average whereas the light blue filled area represents the standard deviation)

In order to analyze the experiment stability for a changing replay buffer size, figure 8 shows the corresponding curves for a minibatch size of 10 on the left hand side. The right hand image presents the results for a fixed minibatch size of 50. The experiment stability curve for a minibatch size of 10 results in stabilities between $s_{experiment} = 0.9228$ and $s_{experiment} = 0.9511$. The experiment stabilities deviate with a standard deviation of miminum 0.0084 (replay buffer size = 600) and maximum of 0.0305 (replay buffer size = 900). By increasing the minibatch size to 50, the experiment stability curve is shifted downwards towards a lower stability. With this setup, a global maximum is reached with $s_{experiment} = 0.9176$ at a replay buffer size of 50. Setting the replay buffer size to 650 results in the lowest experiment stability with $s_{experiment} = 0.8801$. Furthermore, standard deviations range from 0.0142 to 0.0334. In average it is 0.0223 which is higher than for the setup with a minibatch size of 10 where the average standard deviation is 0.0143.

6. Discussion

The convergence and the experiment stability are analyzed through experiments with adapting the replay buffer size and minibatch size. The results show impacts on both stability metrics. First of all, increasing the minibatch size is the main factor for the reduction of

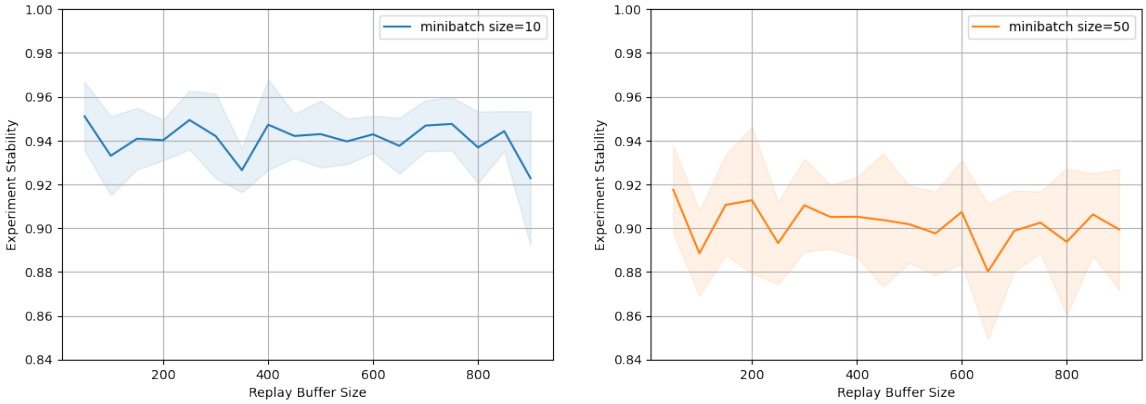

Figure 8: Experiment Stability for different replay buffer sizes and minibatch sizes (The dark blue/orange line represents the average whereas the light blue/orange filled area represents the standard deviation)

stability. Here, an increase from 10 to 150 leads to a 3.5 % lower convergence stability and a 4.6 % lower experiment stability. Additionally, the standard deviation increases with higher minibatch sizes. In comparison to that, the replay buffer size has a smaller impact. While increasing the hyperparameter, both stability measures fluctuate in a range of 3 %.

The introduced metrics focus on different aspects of stability. Especially with the help of the convergence stability, the goal is to minimize instabilities in the learning curve. This possibly leads to curves that on the one hand show minimum instabilities but on the other hand no learning process. Meaning that within the learning process, the total return remains almost the same. As this is not the intended goal, learnings could be included in the metrics or used in order to develop further metrics. Appendix A shows exemplary a comparison of instability and learnings. Here, learnings is defined corresponding to the instabilities in the convergence stability: it sums up all increases in a learning curve (blue lines in figure 1) relatively to the maximum reachable learning. This approach can be used for further research.

7. Conclusion

In this paper, new metrics for measuring stability of DDQN is proposed. A distinction is made between convergence stability and experiment stability. The first one refers to the monotonicity of the learning curve of an experiment whereas the second metric determines the similarity between the results of multiple experiments. Both metrics were determined and analyzed using the example of replay buffer size and minibatch size. The experiments lead to the conclusion that the deterioration of stability is stronger when the minibatch size is increased than when the replay buffer size is increased.

The paper is limited to measuring the stability of DDQN while changing the hyperparameters replay buffer size and minibatch size. In future, more hyperparameters could be analyzed to get a better understanding of the impacts. This possibly enables to determine

parameters that maximize stability. Moreover, the stability metrics do not take into account whether or how much an agent is learning within an experiment. Therefore, a further research direction is the integration of learnings into the introduced metrics.

Appendix A.

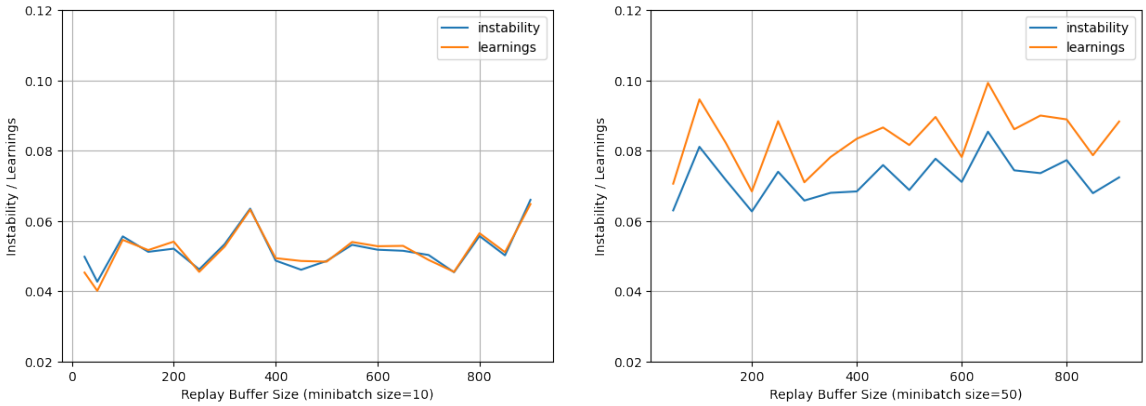

Figure 9: Comparison of Learnings and Instabilities for different replay buffer sizes and minibatch sizes

References

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 176–185. JMLR.org, 2017.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2018.09.005. URL `https://www.sciencedirect.com/science/article/pii/S1367578818301184`.

Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. Boltzmann exploration done right. *Advances in neural information processing systems*, 30, 2017.

François Chollet et al. Keras. `https://keras.io`, 2015.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019.

Shervin Halat and Mohammad Mehdi Ebadzadeh. Modified double dqn: addressing stability, 2021.

Alexandre Heuillet, Fabien Couthouis, and Natalia Díaz Rodríguez. Explainability in deep reinforcement learning. *CoRR*, abs/2008.06693, 2020. URL `https://arxiv.org/abs/2008.06693`.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. *Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Yuxi Li. Deep reinforcement learning. *CoRR*, abs/1810.06339, 2018. URL `http://arxiv.org/abs/1810.06339`.

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

Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. *CoRR*, abs/1710.06574, 2017. URL `http://arxiv.org/abs/1710.06574`.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529—533, February 2015. ISSN 0028-0836. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson series in artificial intelligence. Pearson, 2021. ISBN 9781292401133.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd edition, 2019. ISBN 0273769227.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Junpeng Wang, Liang Gou, Han-Wei Shen, and Hao Yang. Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):288–298, 2019. doi: 10.1109/TVCG.2018.2864504.

Shangtong Zhang and Richard S. Sutton. A deeper look at experience replay. *CoRR*, abs/1712.01275, 2017. URL `http://arxiv.org/abs/1712.01275`.

Declaration of Authorship

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

Hamburg	04/03/2022	A Witte
Place	Date	Signature