"I'm sorry Dave, I'm afraid I can't do that" Deep Q-Learning From Forbidden Actions

Mathieu Seurin[*]
Univ. Lille, CNRS, Inria, UMR 9189 CRIStAL
mathieu.seurin@inria.fr

Philippe Preux
Univ. Lille, CNRS, Inria, UMR 9189 CRIStAL
philippe.preux@inria.fr

Olivier Pietquin
Google Research - Brain Team
pietquin@google.com

Abstract

The use of Reinforcement Learning (RL) is still restricted to simulation or to enhance human-operated systems through recommendations. Real-world environments (*e.g.* industrial robots or power grids) are generally designed with safety constraints in mind implemented in the shape of valid actions masks or contingency controllers. For example, the range of motion and the angles of the motors of a robot can be limited to physical boundaries. Violating constraints thus results in rejected actions or entering in a safe mode driven by an external controller, making RL agents incapable of learning from their mistakes. In this paper, we propose a simple modification of a state-of-the-art deep RL algorithm (DQN), enabling learning from forbidden actions. To do so, the standard Q-learning update is enhanced with an extra safety loss inspired by structured classification. We empirically show that it reduces the number of hit constraints during the learning phase and accelerates convergence to near-optimal policies compared to using standard DQN. Experiments are done on a Visual Grid World Environment and Text-World domain.

1 Introduction

Despite the success of Reinforcement Learning (RL) (Sutton and Barto, 2018) in different domains: Games (Silver et al., 2018), Ressources Management (Mao et al., 2016), Chemical reaction (Zhou et al., 2017), many reasons are deterring industrial from using reinforcement learning in the real world. One of them is the agent's unpredictability in unknown situations. Those problems are especially harmful when the algorithm is embedded in a physical system such as traffic-light management (El-Tantawy et al., 2013), or data-cooling center (Lazic et al., 2018) where a bad action can lead to catastrophic consequences (Damaging material, putting people in danger).

On the other hand, many real-world systems are equipped with security locks, in the form of *forbidden actions* or external controller taking over when the system is behaving wrongly. For example, most servo-motors present in robots are equipped with over-temperature monitoring, and over-voltage detection features or cleaning robots automatically u-turn when an obstacle is present.

[*]Contact Author

Accepted at NeurIPS-2019 Workshop on Safety and Robustness in Decision Making

Another motivation, taking inspirations from Natural Language Processing and Dialogue, is the integration of external rejection signal such as syntax parser or autocorrect mechanism. If integrated correctly, those tools could simplify language acquisition by removing unnecessary or wrong sentences.

However, model-free reinforcement learning is not designed to take this type of information into account. In the current Markov Decision Process (MDP) framework (Puterman, 2014), a rejected action, from the agent point of view, is seen as a transition to the same state. The state-action value is only decreased by a γ factor (more details in subsection 3.1) misrepresenting the fact that the action is potentially harmful. The only way to modify the agent behavior is to tweak the reward function, giving a negative reward when a forbidden action is taken. This technique is a form of *reward shaping* (Ng et al., 1999), which is known to be hard and potentially changes the optimal policy in unpredictable ways.

In this paper, we propose a better integration of forbidden actions into a Q-learning-like algorithm by adding a classification loss that maintains Q-values of forbidden actions below valid ones. We show empirically that it reduces the number of calls to forbidden actions by the agent, and it accelerates the convergence to near-optimal policies compared to standard DQN. Those experiments are conducted on two tasks: a visual grid world and a textual environment.

2 Context: Reinforcement Learning

In reinforcement learning (Sutton and Barto, 2018), an agent learns to interact with an environment so as to maximize a cumulative function of rewards. At each time step t, the agent is in a state $s_t \in \mathcal{S}$, where it selects an action $a_t \in \mathcal{A}$ according its policy $\pi : \mathcal{S} \to \mathcal{A}$. It move to the next state s_{t+1} according to a transition kernel $\mathcal{P}$ and receives a reward r_t drawn from the environment's reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The quality of the policy is assessed by the Q-function defined by $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_t \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\right]$ for all (s,a) where $\gamma \in [0,1]$ is the discount factor. The optimal Q-value is defined as $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, from which the optimal policy π^* is derived.

We here use Deep Q-learning (DQN) to approximate the optimal Q-function with neural networks and perform off-policy updates by sampling transitions (s_t, a_t, r_t, s_{t+1}) from a replay buffer (Mnih et al., 2015).

3 Method

3.1 Feedback Signal and MDP-F

We augment the MDP model with a *Feedback Signal*, a Boolean indicating whether an action was accepted by the environment or rejected. A MDP-F is then defined as a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{F} >$ where $\mathcal{F}$ is a function mapping a state s_t and action a_t to a binary value $\mathcal{F} : S \times \mathcal{A} \to (0,1)$ with 0 meaning the action is valid and 1 meaning unsafe/rejected action.

Why Q-learning algorithm should integrate this information Vanilla Q-learning struggles to differentiate between actions flagged as forbidden and valid ones. Consider the following example: An agent is in a state s takes action a flagged as forbidden, $\mathcal{F}(a) = 1$. When applying the Q-learning update ($Q(s, a_{feed}) = r(s, a_{feed}) + \gamma max Q(s', a')$), since the action was rejected, $r(s,a) = 0$ and $s = s'$. Thus the update becomes $Q(s, a_{feed}) = \gamma max Q(s, a')$. In current Deep Reinforcement Learning setup γ is usually set between 0.99 (Mnih et al., 2015) and 0.999 (Pohlen et al., 2018). So a DQN-like algorithm will require a lot of transitions to get the Q-function of forbidden actions to get smaller. As a result, it will try the forbidden action many time. We are emphasizing that an invalid action indicates an action that could be harmful, so rapidly identifying and avoiding those potentially dangerous situations is essential.

3.2 Frontier loss

Link to Imitation Learning In Imitation learning, few expert demonstrations are available and extracting as much information from those is the key. For example Hester et al. (2018); Piot et al.

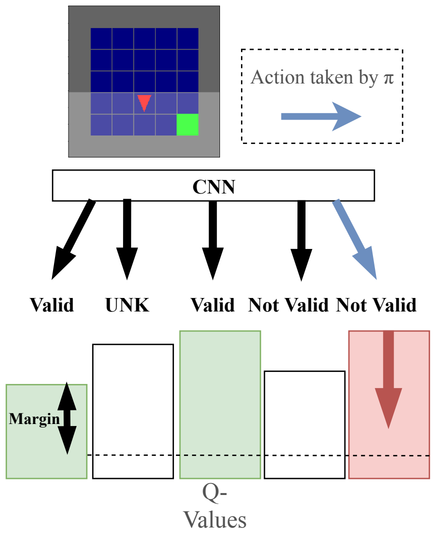


Figure 1: Illustration of frontier loss

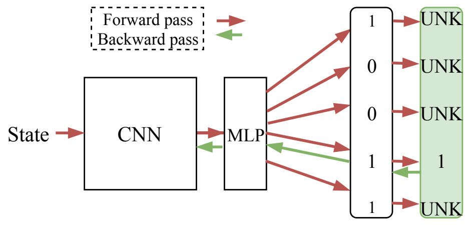


Figure 2: The training procedure is simple, the model predicts the validity for each action, and we only backprogate for the action the agent took

(2014) slightly modify the Q-learning update to nudge expert actions-value above other actions. This is done by adding a secondary loss inspired by structured classification:

$$\text{Minimize } J_{\text{Expert}}(Q) = \max_{a \in A}[Q(s,a) + l(a_{\text{expert}}, a)] - Q(s, aE)$$

where $l(a_{\text{expert}}, a) = 0$ when $a_{\text{expert}} = a$ and m otherwise. This nudges the Q-value of actions taken by the expert above the Q-value of other actions by at least a certain margin m.

Similarly, we want to derive a loss that penalizes the Q-function when a *forbidden action's* value excesses the value of a valid one.

Frontier loss Ideally, for every state during training, we would like the Q-value of all forbidden actions to be below each valid actions, with a certain margin m. We call this loss *frontier loss* :

$$\text{Minimize } J_{\text{Frontier}}(Q) = Q(s, a_{bad}) - \min_{\forall a \text{ where} F(s,a)=0} [Q(s,a) - m]$$

In our experiments, $m = 0.1$.

Frontier loss and classification The main problem regarding this objective function is the need to know for every state which actions are valid. In most tasks, the agent encounters the majority of states only once. It's needed to rely on function approximation to estimate which actions are valid in a given state. To achieve this, we train a neural network to predict in each state which action will be valid. The training procedure is illustrated in Figure 2. To consider an action as valid and to avoid early mis-classifications, we put a threshold after the sigmoid function. The action is considered to be valid if its value is above the threshold.

Frontier loss and DQN Combining the frontier loss and Deep Q-learning is simple as it only requires to sum the two losses. We use a weighting factor λ. For all the experiments described below, we use $\lambda_{\text{q\_learning}} = 1$ and $\lambda_{\text{frontier\_loss}} = 0.5$. Not much tweaking is required regarding this hyper-parameter.

$$\text{final loss} = \lambda_{\text{q\_learning}} + \lambda_{\text{frontier\_loss}}$$

4 Experiments

We designed two experiments on two different domains to assess the quality of our method.

```
Algorithm 1: Frontier loss and classification network
Data: minibatch $b$ from replay buffer $\mathcal{R}$, Q-network $\mathcal{Q}$, classification network $\mathcal{C}$
Result: Frontier loss
1 loss = 0;
2 for <state $s$, action $a$, feedback $f$ > in minibatch $b$ do
3     if $f$ == 1 then
4         valid_actions = C(s);
5         if min[Q(valid_actions)] < Q(s, a) - m then
6             loss = loss + || min[Q(valid_actions)] - Q(s, a) -m ||$^2$
7         ;
8 return loss;
```

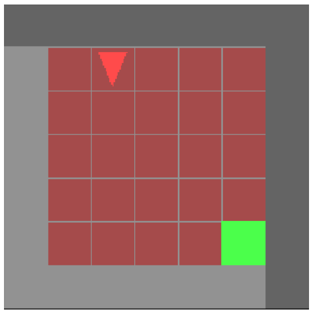
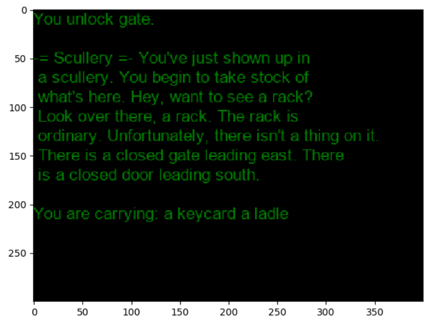


Figure 3: An instance of the MiniGrid problem. The state is a partial view of the maze (point of view of the agent) to avoid problem regarding partial observability, we stacked the last 3 frames


You unlock gate.

= Scullery =- You've just shown up in a scullery. You begin to take stock of what's here. Hey, want to see a rack? Look over there, a rack. The rack is ordinary. Unfortunately, there isn't a thing on it. There is a closed gate leading east. There is a closed door leading south.

You are carrying: a keycard a ladle

Figure 4: An example of interaction in TextWorld. The agent knows : what happened after his last action, a room description and its inventory content

4.1 MiniGrid Enviroment

The first environment is a simple visual gridworld presented in (Chevalier-Boisvert et al., 2019). The goal is to reach the green zone starting from a random point. Since we want to study how the agent can integrate feedback about action's validity, we increase the action space size. The primary action space is composed of 3 actions (Turning Left, Turning Right, Going Forward), and each action is duplicated k times. The action space size becomes $3 \times k$. Then, we create k different rooms in which the agent has to navigate, the color of the background indicating *which set of actions is valid*. For example, in the red room, only actions 11, 12, 13 are valid, and all the others are returning a **not valid** feedback. In our setup, we use $k = 5$ making a total of 15 actions.

The state space is a compact encoded representation of the agent's point of view (more details in (Chevalier-Boisvert et al., 2019)). Since the environment is partially observable, we stack the last three frames as done is Mnih et al. (2015) but we don't use frame-skipping. An episode ends when the agent reaches the green zone or after 200 environment steps.

4.2 TextWorld Environment

TextWorld (Côté et al., 2018) is a text-based game where the agent interacts with the environment using short sentences. We generated a game composed of 3 rooms, 7 objects, and quest length of size 4. An example is shown on Figure 4. In this context, we modified the environment to fit our needs. The action space is composed of all <action> <object> pairs, creating a total of 46 actions. Most of the actions created will be rejected by the simulator since they will not fit the situation the agent is facing. For example, the action "take sword" will be rejected if no sword is available.

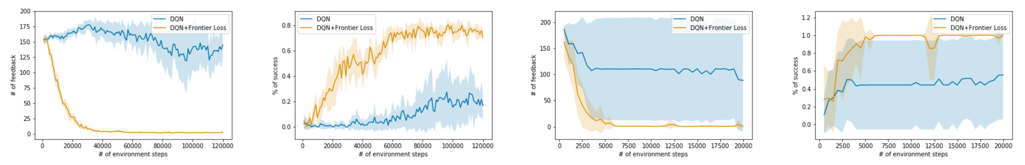


Figure 5: DQN+Frontier (yellow) DQN (blue). Results are averaged over five random seeds for Minigrid and nine random seeds for TextWorld. The shaded area represents one standard deviation. **Left** Performance on Minigrid. The first plot represents the number of times a forbidden action is taken, and the second one represents the percentage of success over time. **Right** Results on TextWorld, the high standard deviation indicates that DQN struggles to reach the level's end consistently.

4.3 Model and architecture

During all experiments, we use Double DQN (Hasselt et al., 2016) with uniform Experience Replay and ϵ-greedy exploration. In the Minigrid environment, we use a Convolution Neural Network (LeCun et al., 1995) with a fully-connected layer on top. In TextWorld, inventory, observation, and room descriptions are each encoded by an LSTM (Hochreiter and Schmidhuber, 1997), concatenated, processed by a fully-connected layer on top.

The classification network matches exactly the architecture used by DQN, *i.e.* ConvNet for Minigrid and LSTM's for TextWorld, the only difference resides in training (explained Figure 2)

5 Results

In Figure 5 we compare DQN and DQN plus the frontier loss. In the Minigrid domain, DQN struggles to find the optimal policy and reaches only 20% of the time the exit. Most of the time, DQN is able to solve one room but fails to find the set of actions for each room, performing forbidden actions over and over. On the contrary, the frontier loss is guiding DQN, reducing the number of feedback signals from the environment, and helping to find the optimal policy. Those results are echoed in the TextWorld experiment. DQN solves the game half of the time, and the other half doesn't encounter the reward and as a result, can't solve the game. This could be mitigated by having a better exploration strategy. Visualization of Q-values can be found in Appendix B

6 Related Works

Action Elimination Closely related to our work is the notion of *action elimination* which was introduced in (Even-Dar et al., 2006). The main idea developed in that work, applied to Multi-Arm Bandits (Lattimore and Szepesvári, 2018; Lai and Robbins, 1985; Robbins, 1952) is to get rid of a sub-optimal action as soon as the value of this action is out of some confidence interval.

A similar idea was applied in Deep Reinforcement Learning by Zahavy et al. (2018). This article shares similarities with ours as the authors are trying to eliminate actions based on a signal given by the environment, indicating if the action is valid or not. They are using a contextual bandit to assess the elimination signal's certainty, and remove actions from the action set A when the confidence is above a certain threshold. The main difference is in the way the elimination signal acts on Q-learning. In their case, the elimination signal doesn't change Q-values but modifies the action set directly.

Alshiekh et al. (2018) define the term *shielding* similar to our notion of feedback. The simulator rejects potentially harmful actions. To learn from this process, the agent outputs a set of actions, ordered by preferences, and the simulator picks the best-allowed action.

Learning when the environment takes over Orseau and Armstrong (2016) design agent to *not* take into account feedback from the environment. For example, for an agent operating in real-time, it may be necessary for a human operator to prevent executing a harmful sequence and lead the agent into a safer situation. However, if the learning agent expects to receive rewards from this sequence, it may learn in the long run to avoid such interruptions, for example, by disabling the *off-button*. Under this setup, they showed that Q-learning could be interrupted safely, supporting the hypothesis that Q-learning is not integrating the feedback signal.

Large Discrete Action Space A benefit of our method is to simplify policy space search when using bigger action space. Dealing with large discrete action space in Deep Reinforcement Learning was studied by Dulac-Arnold et al. (2015) where they use an action embedding in continuous space and map it to a discrete action. Recent methods build upon this by also learning the action embedding instead of using a pre-computed one Chandak et al. (2019); Tennenholtz and Mannor (2019); Chen et al. (2019); Adolphs and Hofmann (2019) Another body of literature explores how to reduce combinatorial action space (He et al., 2016b,a; Dulac-Arnold et al., 2012).

7 Conclusion

In this paper, we proposed a frontier loss, combined with a classification network, which nudges Q-values of rejected actions below Q-values of valid actions. We demonstrate its effectiveness on two simple benchmarks, a visual grid world, and a TextWorld domain. The frontier loss reduces the number of calls to rejected actions and guides early exploration, helping Deep Q-learning collecting more rewards.

Future Work Generalizing the frontier loss in continuous action space would be a key to use this type of algorithm in robotics and more realistic settings. Combining the loss with action's embedding could allow generalization to unseen actions. For example, learning that "Take sword" is rejected "Grab sword" shouldn't be considered by the algorithm.

Acknowledgments

The authors would like to acknowledge the stimulating research environment of the SequeL Team. Special thanks to Florian Strub and Edouard Leurent for fruitful discussions We acknowledge the following agencies for research funding and computing support: Project BabyRobot (H2020-ICT-24-2015, grant agreement no.687831), and CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020

References

Adolphs, L. and Hofmann, T. (2019). Ledeepchef: Deep reinforcement learning agent for families of text-based games.

Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U. (2018). Safe reinforcement learning via shielding. In *Proc. of AAAI*.

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. (2019). Learning action representations for reinforcement learning. In *Proc. of ICML*.

Chen, Y., Chen, Y., Yang, Y., Li, Y., Yin, J., and Fan, C. (2019). Learning action-transferable policy with action embedding. *arXiv preprint arXiv:1909.02291*.

Chevalier-Boisvert, M., Bahdanau, D., Lahlou, S., Willems, L., Saharia, C., Nguyen, T. H., and Bengio, Y. (2019). BabyAI: First steps towards grounded language learning with a human in the loop. In *Proc. of ICLR*.

Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., Asri, L. E., Adada, M., Tay, W., and Trischler, A. (2018). Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532.

Dulac-Arnold, G., Denoyer, L., Preux, P., and Gallinari, P. (2012). Fast reinforcement learning with large action sets using error-correcting output codes for mdp factorization. In *Proc. of ECML and PKDD*, pages 180–194. Springer.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. (2015). Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*.

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *Proc. of TITS*.

Even-Dar, E., Mannor, S., and Mansour, Y. (2006). Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105.

Hasselt, H. v., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proc. of AAAI*.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. (2016a). Deep reinforcement learning with a natural language action space. In *Proc. of ACL*, pages 1621–1630.

He, J., Ostendorf, M., He, X., Chen, J., Gao, J., Li, L., and Deng, L. (2016b). Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *Proc. of EMNLP*, pages 1838–1848.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al. (2018). Deep q-learning from demonstrations. In *Proc. of AAAI*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Lattimore, T. and Szepesvári, C. (2018). Bandit algorithms.

Lazic, N., Boutilier, C., Lu, T., Wong, E., Roy, B., Ryu, M., and Imwalle, G. (2018). Data center cooling using model-predictive control. In *Proc. of NeurIPS*.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*.

Mao, H., Alizadeh, M., Menache, I., and Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proc. of ACM Workshop on Hot Topics in Networks*.

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.

Ng, A. Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of ICML*.

Orseau, L. and Armstrong, S. (2016). Safely interruptible agents. In *Proc. of UAI*.

Piot, B., Geist, M., and Pietquin, O. (2014). Boosted Bellman residual minimization handling expert demonstrations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8725 LNAI(PART 2):549–564.

Pohlen, T., Piot, B., Hester, T., Azar, M. G., Horgan, D., Budden, D., Barth-Maron, G., Van Hasselt, H., Quan, J., Večerík, M., et al. (2018). Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593*.

Puterman, M. L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

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

Tennenholtz, G. and Mannor, S. (2019). The natural language of actions. In *Proc. of ICML*.

Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D. J., and Mannor, S. (2018). Learn what not to learn: Action elimination with deep reinforcement learning. In *Proc. of NeurIPS*.

Zhou, Z., Li, X., and Zare, R. N. (2017). Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344.

A Appendix A: Training details

A.1 Minigrid Network and traininig

Convolution net, 3 layer, 16, 32, 64
Kernel size : 2,2,2
pooling : 2 on the first layer
FC 64 -> actions
rmsprop
1e-5 learning rate, decayed to 1e-7
weight decay 1e-4
replay buffer size 10000
target update every 10'000 steps
classif learning rate : 1e-4
sigmoid ceil classif : 0.9 (don't consider action as valid if the sigmoid doesn't excess this value)

A.2 TextWorld Network and training

word embedding size : 128
inventory rnn size : 256
description rnn size : 256
obs rnn size : 256
fc hidden : 350
rmsprop
1e-3 learning rate
weight decay 1e-4
replay buffer size 10000
target update every 2'000 steps
classif learning rate : 1e-4
sigmoid ceil classif : 0.9 (don't consider action as valid if the sigmoid doesn't excess this value)

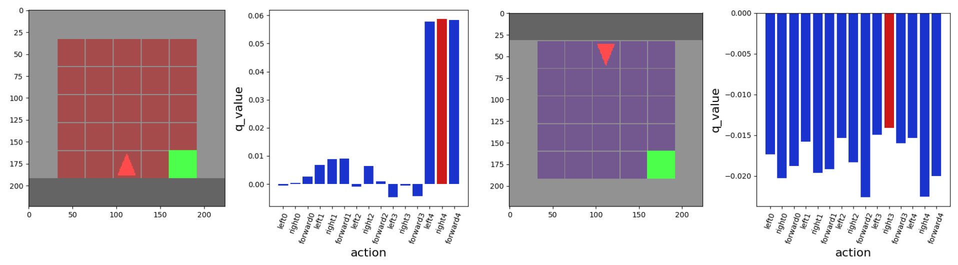


Figure 6: Q-value, after 150 episodes of training (30'000 env steps) **Left** : Frontier loss **Right** : DQN

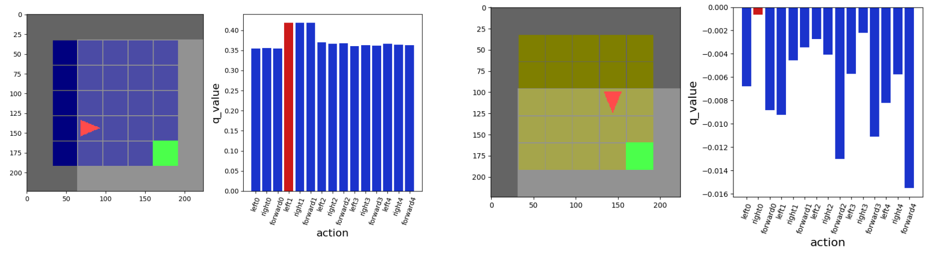


Figure 7: Q-value, after 100'000 env steps **Left** : Frontier loss **Right** : DQN

B Appendix B : Visualisation of Q-values