"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.**

***Index Terms*—Deep Reinforcement Learning, Safety, constraints, Q-Learning**

I. Introduction

Reinforcement Learning (RL) (Sutton and Barto, 2018) is the main machine learning answer to address sequential decision-making problems under uncertainty. Its genericity allows application to a variety of domains such as Robotics (Gu et al., 2017; Levine et al., 2016), Resource Management (Mao et al., 2016), Chemical reaction (Zhou et al., 2017), Traffic-light Management (El-Tantawy et al., 2013), Spoken Dialogue Systems (Lemon and Pietquin, 2012) and sparked interest in other industrial applications. Bringing reinforcement learning to critical systems such as Surgery (Hashimoto et al., 2018), Dam Management Wang and Xu (2012) or Autonomous Driving (Leurent and Mercat, 2019; Sallab et al., 2017) remains one of the most interesting open challenges in machine learning. Unexpected behavior, inability to handle uncertainty and low sampling efficiency are the Achilles heels of real-world RL (Dulac-Arnold et al., 2019).

The major issue is that designing RL agents able to measure uncertainty about their internal estimates (*e.g.* their state, the outcome of their actions) is difficult (Geist and Pietquin, 2011) and especially when associated with recent Deep Learning methods (O'Donoghue et al., 2018). Even if available, using a measure of uncertainty for safety is not straightforward either (Bellemare et al., 2017; Daubigney et al., 2011). Uncertainty is especially harmful when agents are meant to control a physical system, where a wrong action can lead to catastrophic consequences (*e.g.* damaging material or endangering people).

Fortunately, many real-world systems are equipped with contingency measures, in the form of *forbidden actions* or external controller taking over when the system is misbehaving. For example, over-temperature monitoring regulates servo-motors present in robots, preventing the motors from reaching their heat limit. Cleaning robots also automatically u-turn in the presence of an obstacle, preventing any damage to the machine and its environment.

Beyond critical systems, many areas could benefit from *forbidden actions*. In Natural Language Generation Reiter and Dale (1997) or Dialogue systems Chandramohan et al. (2010); Chen et al. (2017), syntax parser or auto-correct mechanism can act as an external rejection signal. Indicating which word does not fit a generated sentence or pointing out grammar mistakes could improve language generation by greatly reducing word's space, leveraging language learning and generation.

These examples show a potential misalignment between the standard RL frameworks and the potential real-world applications. Of course, designing constraints to avoid critical situations requires expert knowledge about the system to be controlled. But it is often the case that environments already implement such contingency measures (obstacle avoidance, circuit breaker, etc.).

In this paper, we consider a simple type of external constraints, prevalent in many real-world problems: when the agent is about to perform a hazardous action, the system rejects it and thus prevents the agent from doing so. The agent then follows the natural dynamics of the environment (Alshiekh et al., 2018). We aim at building an algorithm that learns from these rejected actions.

In the general Markov Decision Process (MDP) framework (Puterman, 2014), a rejected action, from the agent's point of view, would be seen as a transition to the same state. Everything happens like if the action had no effect. This misrepresents the potential harmfulness of the action and prevents the agent from learning anything useful from the rejected action (See subsection III-A for a detailed explanation). We want a model-free reinforcement learning to benefit from those constraints so as to learn faster to avoid hazardous states and alleviate exploration problems.

In the coming sections, after introducing the RL paradigm, we propose a constrained version of Deep Q-learning (DQN) (Mnih et al., 2015) by adding a classification loss that maintains Q-values of forbidden actions below valid ones. We then validate our method empirically, showing that vanilla DQN struggles at solving tasks with rejected actions while our algorithm reduces the number of calls to forbidden actions. It accelerates convergence to near-optimal policies compared to standard DQN. Experiments are conducted on two tasks: A maze navigation using visual features and a text-based game.

II. Context: Reinforcement Learning

In the reinforcement learning (RL) paradigm (Sutton and Barto, 2018), an agent learns to interact with its environment so as to maximize a cumulative function of rewards. The environment to be controlled is modeled as a Markov Decision Process (MDP) that is a tuple $\{S, A, \mathcal{P}, \mathcal{R}, \gamma\}$ which elements are defined as follows. 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 to its policy $\pi : \mathcal{S} \to \mathcal{A}$. It moves to state s_{t+1} according to a transition kernel $\mathcal{P}$ and receives a reward $r_t = r(s_t, a_t)$ drawn from the environment's reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. 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.

$$\pi^*(s) \in \arg\max_a Q^*(s,a)$$

We here use the Deep Q-learning (DQN) algorithm (Mnih et al., 2015) 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 (Lin, 1992).

III. Method

A. Feedback Signal and MDP-F

In this section, we present a way to integrate forbidden actions into the MDP framework. We augment the MDP model with a *Feedback Signal* (Alshiekh et al., 2018), a Boolean indicating whether an action was accepted by the environment or rejected. A MDP-F is then defined as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{F} >$ 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.

Vanilla Q-learning struggles to differentiate between actions flagged as forbidden and valid ones. Consider the following example: an agent in a state s takes action a flagged as forbidden ($\mathcal{F}(a,s) = 1$). 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). DQN-like algorithm will require lots of transitions to make the Q-function of forbidden actions smaller, thus loosing time to explore and collect useful samples. We emphasize that an invalid action indicates an action that could be harmful, so rapidly identifying and avoiding those potentially dangerous situations is crucial.

B. Frontier loss

We take inspiration from the learning from demonstrations paradigm where one wants to use expert demonstrations to induce the usage of preferred actions in RL agents.

a) Expert loss and Imitation learning: In Imitation learning, few expert demonstrations are available and extracting as much information from those is essential. For example (Hester et al., 2018; Piot et al., 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 most 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.

b) Frontier loss: The optimal policy π^* (derived from Q^*) will never take a forbidden action as it keeps the agent in the same state. Based on this assumption we can derive the following rule: for every state encountered during training, the Q-values of all forbidden actions should be below the one of each valid actions, within a certain margin m. This defines a new loss that we call *frontier loss* :

$$\text{Min } J_{\text{Frontier}}(Q) = Q(s, a_{bad}) - \min_{a \in \mathbb{A}_{\text{s,v}}} [Q(s,a) - m]$$

The margin is a hyperparameter that depends mostly on the Q-values magnitude. In our experiments, since the rewards are bounded between 0 and 1, the margin is small ($m = 0.1$).

c) Frontier loss and classification: The main problem regarding this objective function is the need to know which actions are valid for every state. In most tasks, the agent encounters the majority of states only once, specially in visual domain. Thus, we need 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**. For every action taken, we store the corresponding feedback, creating a dataset of (s, a, f). The network, taking the state as input, predicts a binary value for every action (1 for valid, 0 for invalid). For each state in our transition dataset, since only one action is labelled, we need to adapt the training regime. We can achieve this by masking the gradients from untaken action, only backpropagating for the

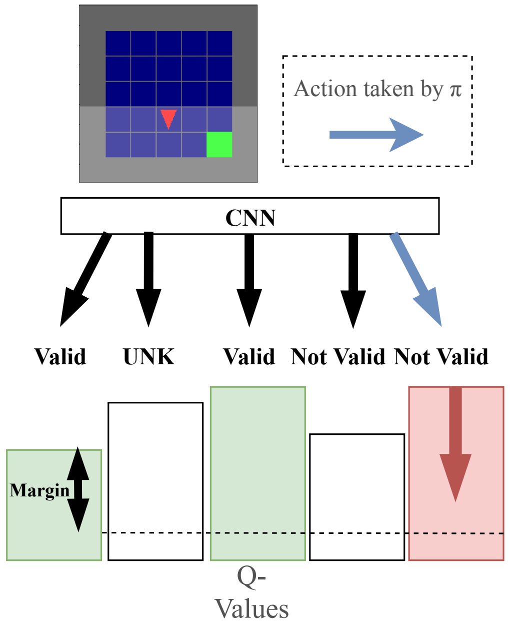


Figure 1. Illustration of frontier loss.

action the policy π took. The training procedure is illustrated in Figure 2.

To consider an action as valid and to avoid early misclassifications, we put a threshold after the sigmoid function. The action is considered to be valid if its activation is above the threshold.

d) 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 to balance the two losses. 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.

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

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         $a_{bad}$ = a                              ▷ Renaming for clarity
5         actions$_{valid}$ = C(s); ▷ Estimated valid action set
6         if min[Q(s, actions$_{valid}$)] < Q(s, $a_{bad}$) - m then
7             loss = loss + || min[Q(s, actions$_{valid}$)]
8             - Q(s, $a_{bad}$) -m ||$^2$
9 return loss;
```

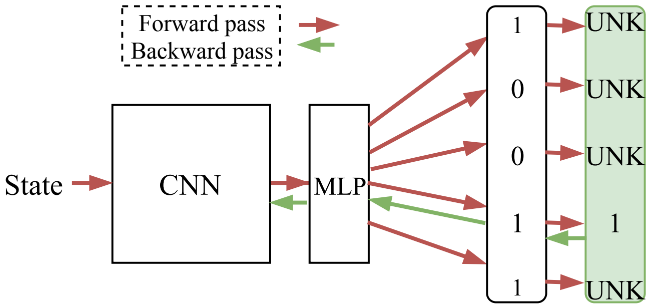


Figure 2. Training procedure: the model predicts the validity for each action, and we only backpropagate for the action the agent took.

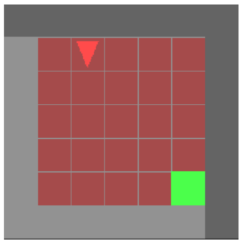


Figure 3. An instance of the MiniGrid problem. The state is a partial view of the maze (point of view of the agent) to avoid problem regarding partial observability, we stacked the last 3 frames.

IV. Experiments

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

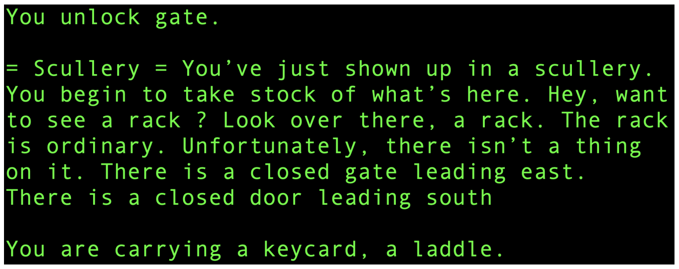
A. *MiniGrid Enviroment*

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 an encoded representation of the agent's point of view represented as different features maps allowing the use of convolution layer (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 do not use frame-skipping. An episode ends when the agent reaches the green zone or after 200 environment steps. An illustration of the environment can be found Figure 3

B. *TextWorld Environment*

TextWorld (Côté et al., 2018) is a text-based game where the agent interacts with the environment using short sentences.



```
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 laddle.
```

Figure 4. An example of interaction in TextWorld. The agent has access to: what happened after its latest action ("You open the door, it's very dark in here, [...]"), a room description ("Attic, an empty room, maybe you should head back. You can go North, East") and its inventory content ("Keycard, Mask").

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. An illustration of the game can be found Figure 4.

C. Model and architecture

During all experiments, we use Double Deep Q-Network (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)

V. Results

In Figure 5, Figure 6, Figure 7, Figure 8, we compare DQN and DQN equipped with 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 does not encounter the reward and as a result, can not solve the game. This could be mitigated by having a better exploration strategy, but it shows that shaping Q-values is enough to reduce the sample complexity and guide exploration. We want to emphasize that in early stage of the training, the classification network performs poorly due to low

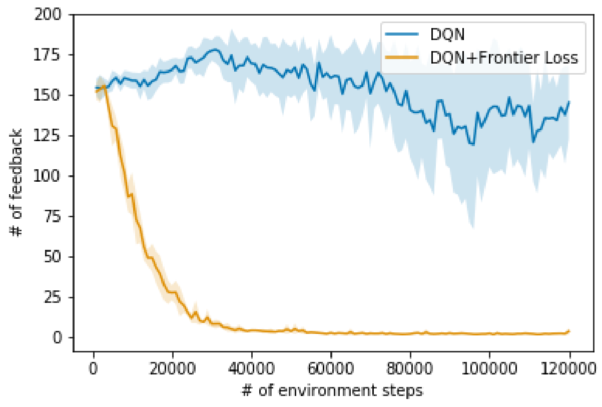


Figure 5. **Minigrid results: Number of times a forbidden action is taken.** DQN+Frontier (yellow) DQN (blue). Results are averaged over five random seeds. The shaded area represents one standard deviation.

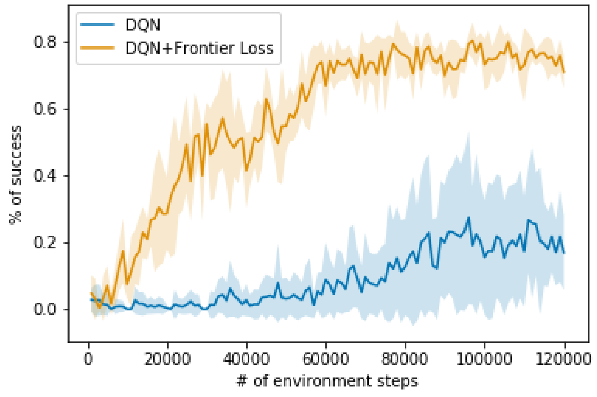


Figure 6. **Minigrid results: Percentage of success over time.** DQN+Frontier (yellow) DQN (blue). Results are averaged over 5 random seeds. The shaded area represents one standard deviation.

quantity of samples but it does not hurt the performances of DQN, it's able to quickly learn to avoid forbidden actions as shown. Visualization of Q-values at different stages of training can be found on Figure 9,Figure 10. They clearly illustrates the benefit of using the frontier loss in those setups. Even in the early training stage, the separation between valid actions and invalid is clear, alleviating the difficulty of finding the optimal policy. Where as DQN Q-values are really hard to distinguish from each other.

VI. Related Works

a) 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 (Lai and Robbins, 1985; Lattimore and Szepesvári, 2018; 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

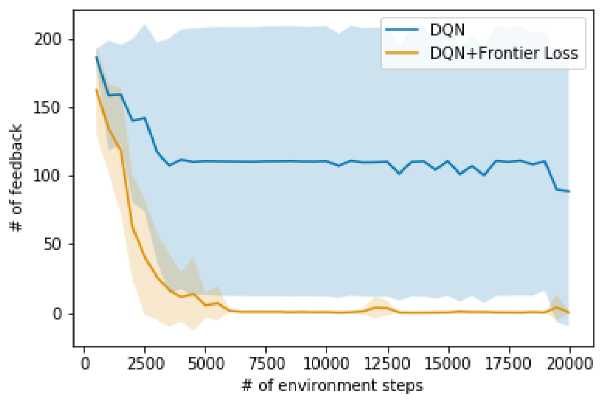


Figure 7. **Textworld results: Number of invalid action taken by the agent.** DQN+Frontier (yellow) DQN (blue). Results are averaged over nine random seeds. The shaded area represents one standard deviation.

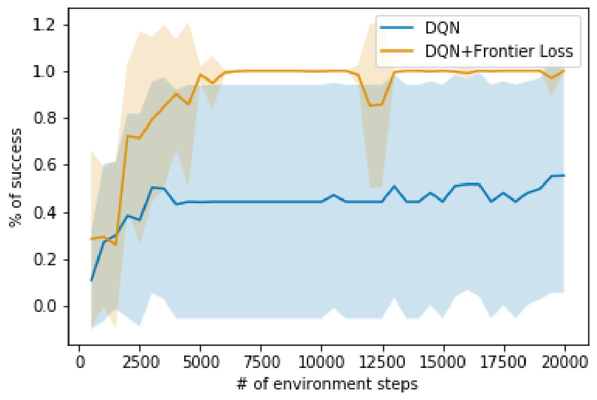


Figure 8. **Textworld results: Percentage of success over time.** DQN+Frontier (yellow) DQN (blue). Results are averaged over 9 random seeds. The shaded area represents one standard deviation.

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. They need extra care because removing a valid action can irreversibly damage the policy where as in our case, we only nudge values.

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.

b) 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

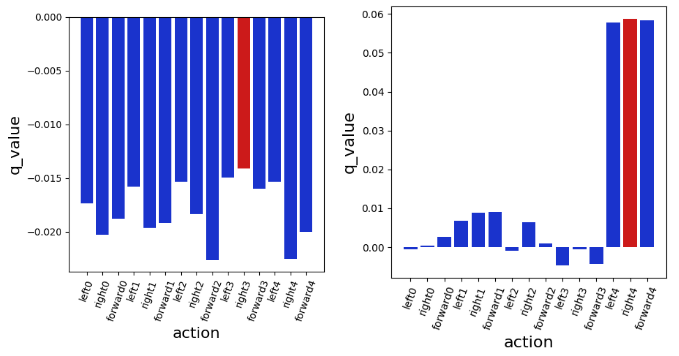


Figure 9. **Q-values, after 150 episodes of training (30.000 env steps) for certain state**. Left: DQN; Right: DQN with frontier loss. We can clearly see how the frontier-loss shapes Q-values, clearly separating forbidden action from good ones.

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 our hypothesis that Q-learning is not integrating the feedback signal.

c) Reward Shaping: Lipton et al. (2016) introduced the notion of Intrinsic Fear (IF) to deal with dangerous states. Deep agents tend to periodically revisit these states upon forgetting their existence under a new policy. IF is shaping the reward to guard agents against periodic catastrophes. To achieve that they train a model to predict the probability of imminent dangerous state and it penalizes the true reward function R with an intrinsic reward F

d) 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 Adolphs and Hofmann (2019); Chandak et al. (2019); Chen et al. (2019); Tennenholtz and Mannor (2019) Another body of literature explores how to reduce combinatorial action space by using an encoding dictionary (Dulac-Arnold et al., 2012) rendering problems with large action sets tractable. (He et al., 2016a,b) are dealing with variable action space, meaning that at every time step, the action set size is different. To cope with this challenge, they compute a state embedding and action embedding for every action available then perform a dot product between the two. This allows them to deal with variable sized action set.

VII. Conclusion

Critical real-world systems are constrained by safety measures that prevent hazardous action. We hypothesize that Q-learning, a model-free reinforcement learning method struggles with those constraints. In this paper, we proposed a frontier loss, combined with a classification network, to help DQN.

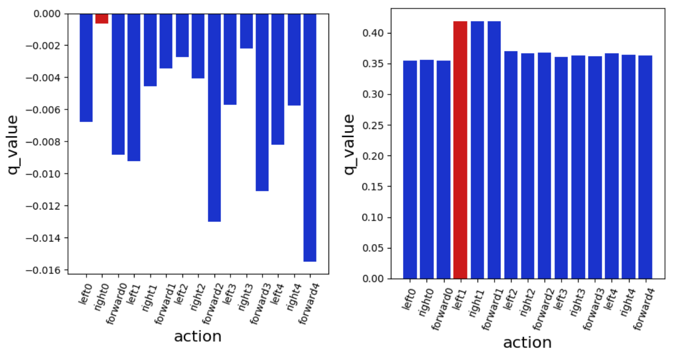


Figure 10. **Q-values, after 100 000 training steps** Left: DQN; Right: DQN with frontier loss. Even after 100 000 network updates, DQN still struggles to differentiate good action from bad ones.

This algorithm nudges rejected actions Q-values below Q-values of valid actions. We showed empirically that the frontier loss reduces the number of calls to rejected actions and guides early exploration, helping Deep Q-learning achieving higher performances. We demonstrate its effectiveness on two benchmarks, a visual grid world and a TextWorld domain.

a) Limitations and Future Work: At the moment, applying the frontier to continuous action space is non-trivial but it would be a key to use this type of algorithm in robotics and more realistic settings. Another future improvement would be to combine 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. *In NeurIPS Deep Reinforcement Learning Workshop*.

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

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org.

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

Chandramohan, S., Geist, M., and Pietquin, O. (2010). Optimizing spoken dialogue management with fitted value iteration. In *Proc. of ISCA*.

Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Chen, Y., Chen, Y., Yang, Y., Li, Y., Yin, J., and Fan, C. (2019). Learning action-transferable policy with action embedding. In *Proc. of AAAI*.

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.

Daubigney, L., Gašić, M., Chandramohan, S., Geist, M., Pietquin, O., and Young, S. (2011). Uncertainty management for on-line optimisation of a pomdp-based large-scale spoken dialogue system.

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*. 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*.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. (2019). Challenges of real-world reinforcement learning. In *Real-world Sequential Decision Making Workshop @ ICML 2019*.

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.

Geist, M. and Pietquin, O. (2011). Managing uncertainty within the KTD framework. In *Active Learning and Experimental Design workshop @ AISTATS 2010*, pages 157–168.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proc. of ICRA*. IEEE.

Hashimoto, D. A., Rosman, G., Rus, D., and Meireles, O. R. (2018). Artificial intelligence in surgery: promises and perils. *Annals of surgery*, 268(1):70.

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*.

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*.

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.

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

Lemon, O. and Pietquin, O. (2012). *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. Springer Science & Business Media.

Leurent, E. and Mercat, J. (2019). Social attention for autonomous decision-making in dense traffic. In *Machine Learning for Autonomous Driving Workshop at NeurIPS 2019*.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

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

Lipton, Z. C., Azizzadenesheli, K., Kumar, A., Li, L., Gao, J., and Deng, L. (2016). Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*.

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.

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

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. (2018). The uncertainty bellman equation and exploration. In *International Conference on Machine Learning*, pages 3836–3845.

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.

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

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

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*.

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*.

Wang, R. and Xu, L. (2012). Multi-agent dam management model based on improved reinforcement learning technology. *Applied Mechanics and Materials*, 198.

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.

APPENDIX

Convolution Layer	16, 32, 64
Kernel size	2,2,2
Pooling	2 on the first layer
Fully Connected hidden size	64
Optimizer	Rmsprop
Learning rate	1×10^{-5} decayed to 1×10^{-7}
Weight Decay	1×10^{-4}
Replay buffer size	10 000
Target update every	2 000
Action classifier learning rate	1×10^{-4}

Table I
MINIGRID NETWORK AND TRAININIG.

Word embedding size	128
Inventory RNN size	256
Description RNN size	256
Obs RNN size	256
Fully Connected hidden size	400
Optimizer	Rmsprop
Learning rate	1×10^{-5} decayed to 1×10^{-7}
Weight Decay	1×10^{-4}
Replay buffer size	10 000
Target update every	2 000
Action classifier learning rate	1×10^{-4}

Table II
TEXTWORLD NETWORK AND TRAINING.