Reinforcement Report on "Leave no trace"

David-Stephane Belemkoagba et Quentin Puyrazat

February 2020

Contents

1 Introduction

In deep reinforcement learning, we are able nowadays to build algorithms that can learn complex tasks : autonomous cars, walking robots, etc. Learning those systems requires repeating numerously a series of attempts until the agent is able to perform correctly the task : we create our agent, we build the environment, states and actions spaces, and the reward system; in an attempt, we let the agent try to perform the task, we reward him or punish him regarding its performance; in the first attempts, it will fail. But we keep repeating the episodes until, our algorithm learns.

This way of learning have its limits. Let us imagine a real world task, where you teach a real (not simulated) robot to walk. We will need a human intervention every time the robot fails in order to put it back in its standing position; this can be very costly since our agent needs a numerous number of attempts before succeeding the task. So, how can we build systems where we can drastically reduce the number of resets? How can we build systems that can perform this additional task : detect when they are going to fail and avoid actions that will lead to this failure.

In this paper, the authors propose an algorithm where the agent can learn to detect failing actions. This is done by learning two policies : a forward and a reset policy. In the forward policy, the agent learns to complete the original task. And in the reset policy, it learns a Q-function that can, in one hand, evaluate an action and check if this action will lead to an irreversible state (like our falling robot), and then avoid it; and in another hand, learns a policy where the agent knows how to go back to some initial states or safe states (like standing position for our walking robot) when it is likely to fail.

The experiments illustrate that proper use of the reset policy can greatly reduce the number of manual resets required to learn a task, can reduce the number of unsafe actions that lead to non-reversible states.

2 Context of the paper

The method presented in the paper is inspired by previous work on safe exploration, multiple policies and automatic curriculum generation.

For safe exploration, paper [3] proposes a method limited to Small Discrete Markov Processes where they consider a state to be safe if they can reach any other state from this current state.

The idea of multiple policies for safety and for learning difficult task is present on different other papers. The paper [2] for example, uses multiple policy as a tool to achieve complex tasks. Richter Roy whose method is more similar to the method in the paper uses a heuristic deterministic hand-engineered reset policy while the method proposed here learns simultaneously the forward and the reset policy.

Also, the method described in the paper can be viewed with the point of view of the reverse curriculum generation. In the reverse curriculum generation, the agent learns to complete a task gradually by starting from different initial states : there is a special policy which will decide the initial state from which to start given the performances of the agent; initially the agent will be bad and so, the state selector will choose states not so far from the goal and as the agents get better, the state selector will choose more and more difficult initial states. By this way, the agent learns well and gradually.
Coming back to "our" method : the forward policy can be seen as an initial state selector; our agent complete a set of actions until it gets in a critical state where the method triggers the reset policy; and then the reset agent can be viewed then as the agent which learns to complete a task (find equilibrium from critical state and then avoid failure). We can clearly see that, with this point view,the main goal of the agent becomes what the reset policy learns to achieve.This is why the authors insist in their paper that their method is a reverse curriculum learning but in reverse (the reset agent is the main agent and the forward agent is the initial state selector).

In Reinforcement Learning, there is always some concepts which will interact with each others :

- The Agent, which we want to teach a specific task and which will take an action depending on its current state.
- The environment which represents the spatial framework of the agent
- The notion of State space which is a set of possible set in which the agent can be, represented as “S”.
- The Action space which is a set of all possible action that the agent can take to move from a state to another, represented as “A”
- The reward function $r(s, a)$ which will give a reward to the agent or a penalty, depending on its choices for the next state.

At the beginning, the agent will be placed in an initial position in state with probability $P_0(s)$ and will move with the probability $P(s'|s, a)$ to the next state thanks to a policy. Our main goal is to find the best policy which will give to the agent the maximum of expected discounted returns γ.

$$E_\pi[\sum_{t=0}^{T} \gamma^t r(s_t, a_t]$$

where r is the immediate reward that an agent gets when performing action a_t being in state s_t, γ^t is the discounting factor : the further we run into the episodes steps, the less valuable is an action in the expected return from the current state;

This learning task is made through a lot of episodes iterations. An episode is made of a succession of actions taken by the agent to go to a given state and rewards given to the agent by the environment. In fact, the agent is rewarded if it takes a good action given its current state and penalized otherwise. This is done repeatedly until the agent is able to perform the task and cummulate the maximum of rewards possible through an episode.

As we mentioned previously, the issue here is that, when an episode terminates (failure), we have to reset manually (in real world applications for example), the initial state for the next episode. And the goal of this paper, is to find a way to learn a policy that detects when the agent is likely to fail, and starts a series of actions in order to avoid failure and continue learning (this can be seen as a safe mode in which our agent enters when it is likely to fail).

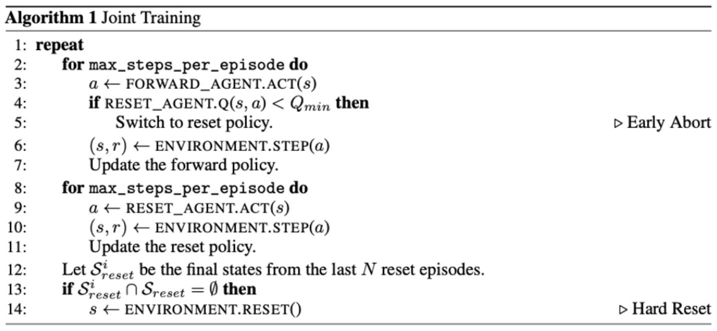
Yet we have to specify that, in simulated environments the resets are not necessarily costly (a couple of intructions in the program can handle it); this is more problematic when we are in real world situations like teaching a robot to walk or an autonomous vehicle.

3 Method

The authors proposed a new Framework which can be used with Q-learning algorithms. This is a necessary condition since the reset agent needs to evaluate (by computing a Q-value) each action taken by the agent in order to measure how safe it is. For example, Actor Critic methods are suitable for this framework; we can use it for both forward and reset agents.

The authors proposed to introduce a new agent called “reset agent” with the main agent which now called forward agent.

Alternately, we train the forward agent to achieve the initial task with a reward function $r_f(s, a)$, and a reset agent to learn how to reach one of the set of “safe states” from any state, with a reward function $r_s(a)$.This method is suitable for continual learning, when we need our agent to run as long as possible while learning to achieve the task.



```
Algorithm 1 Joint Training
repeat
    for max_steps_per_episode do
        $a \leftarrow$ FORWARD_AGENT.ACT($s$)
        if RESET_AGENT.Q($s, a$) $< Q_{min}$ then
            Switch to reset policy.                              ▷ Early Abort
        $(s, r) \leftarrow$ ENVIRONMENT.STEP($a$)
        Update the forward policy.
    for max_steps_per_episode do
        $a \leftarrow$ RESET_AGENT.ACT($s$)
        $(s, r) \leftarrow$ ENVIRONMENT.STEP($a$)
        Update the reset policy.
    Let $\mathcal{S}^i_{reset}$ be the final states from the last $N$ reset episodes.
    if $\mathcal{S}^i_{reset} \cap \mathcal{S}_{reset} = \emptyset$ then
        $s \leftarrow$ ENVIRONMENT.RESET()                       ▷ Hard Reset
```

Figure 1: source : the paper

Let S be the set of states , A the set of possible actions, $\epsilon \subseteq S \times A$ the set of all possible transitions. We define also Q_{reset}, the Q_{value} of our reset policy. We define also $\epsilon^* := \{(s,a) \in \epsilon | Q_{reset}(s,a) > Q_{min}\}$ which is the set of all reversible or safe transitions.
Following this definition, we define the set of all the safe state as $S^* := \{s|(s,a) \in \epsilon^* foratleastonea \in A\}$

The algorithm runs as follow: We alternate two loops for training the forward agent and the reset agent. In the first loop, the forward agent takes an action. This action is evaluated by the reset agent which outputs a Q_{value}. If the Q_{value} is less than a given threshold (Q_{min}), we consider that the action will lead to an irreversible state and we avoid it. We switch to the reset agent, and takes an action with its policy. Given that the reset policy is trained to take actions that lead to safe states, this switch will allow us to stabilize our agent and then come back to the forward policy to continue the training. Those are called early aborts. The second loop is where we train our reset agent.

It is important to specify that the reset policy doesn't learn to do the reverse path done by the agent; we can think that the reset agent learns to do the exact reverse path of all the actions taken by the forward policy, this is not the case. Rather, given a state, it learns a set of actions that will bring the agent to one of its initial states (or set of safe states), to be in safe position (standing position for a walking robot which is falling for example).

Since the reset agent is not perfect, we cannot avoid throughout an entire episode some irreversible states. Then we have to reset the agent "manually". The authors decided that if the agent fails to recover a safe state after N consecutive early aborts, then it must have failed and we have to reset the episode. Those resets are called hard resets.

We can see that two more parameters must be taken into account : the Q_{min} which is the threshold for an action to be considered safe and N which is the number of successive early aborts we take before considering an eventual failure.

Finally, the authors experimented Value Function Ensembles. Since the reset agent estimates the Q-values, those estimations are noisy and sometimes not accurate at all. The idea here is to learn many reset agents that will learn to estimate some Q-values and then take one of the three strategies:

- Optimistic strategy : We consider an early abort if all the estimations are less than Q_{min}
- Realistic strategy : We consider an early abort if the average of the Q-values is less than Q_{min}
- Pessimistic strategy : We consider an early abort when at least one of the Q-values estimations is less than Q_{min}.

We will detail in the results section which strategy works the best.

4 Authors' Experiments

In this section, we are going to discuss all the experiments that have been led by the authors. In order to experiment the new framework, they decided to use off-policy actor critic methods for both the forward and the reset agent.

The authors experimented the framework on many games environments. The simplest of them all is the small-scale didactic game. In this example, the goal of our agent is to succeed to reach two cells in a grid with one of them being a trap: once in it (the one with the trap), the agent cannot get out.To succeed the agent needs then to first enter the cell with no trap and then enter the one containing the trap . Each time the agent will go into a red bounded box (a trap), the algorithm will shut down and will need a human intervention to launch it again. The forward policy will not immediatly distinguish the two goal cells because the agent will receive the same amount of reward but, with the reset policy, the agent will be able to prevent the forward policy to go first in the wrong goal cell. We can visualize in the grid the zones where early aborts occured (blue cells). We can clearly see that those cells are near the dangerous cells, meaning that our agent has been able, thanks to the reset agent, to detect much of the time when it is likely to take the wrong action and then avoid it.

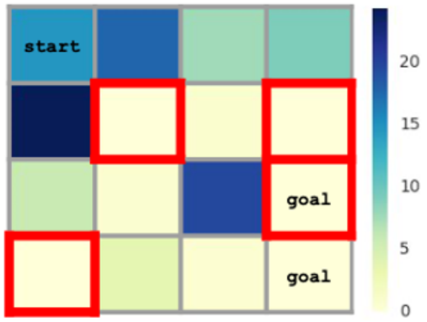
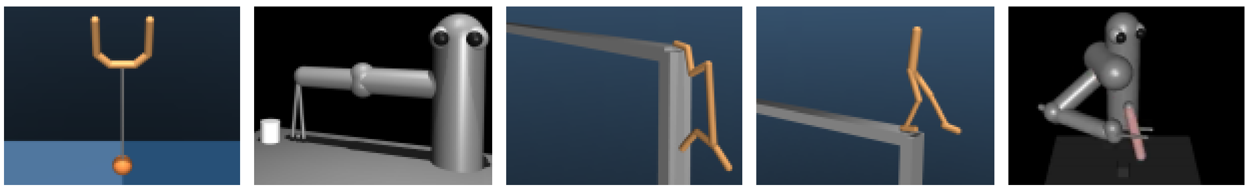


Figure 2: Early aborts with an absorbing goal.

The previous example was a quite simple problem in a discrete environment. The authors also experimented the framework on continuous cases like the ones below. For example :

- Ball in cup : The goal here is to find how to balance the cup in order to ball in it. The overall elements (ball, cup, rope) are under the physics laws like gravity etc.
- the cliff walker : The goal here is to learn our humanoid to walk without falling.



5 Authors' Results

The first result shown in the paper is with the Small-Scale Didactic. In fact, as explained before, this new algorithm allows the agent to reduce the number of failure. However, this reduce of the number of hard reset has a cost : the agent will not go into dangerous state but will need more steps to learn efficiently. Indeed, if we increase the Q_{min} (which is the threshold used by the reset policy to able or not the forward policy to go into the state it wants), the reset policy will impact more the forward policy and will force it to be safer and safer. This configuration is an issue for the exploration and will require more number of iterations to reach the same information than with a lower Q_{min}.

Furthermore, it has been shown in the paper that this algorithm is able to solve issues that only forward policy can't. For that, the algorithm has been compared to two others : the forward-only policy and the status quo policy (which correspond to have hard reset). As we can observe in the figure, the forward-only policy doesn't succeed to learn because the reward policy is inversely proportional to the number of steps, contrary to the two other algorithms.

The status quo method can't be used outside the simulation because it will need a lot of hard reset to be able to learn.

As explained before, the number of hard reset will be reduce and this has been shown with experiments in the paper with comparison with the two other algorithm presented before. In the figure 2, we can see that the conventional approach has the ability to learn faster (in term of reward depending on the number of steps) than the Leave No Trace algorithm but will require twice more hard reset. Thus, the number of hard reset has been reduced with this new algorithm.

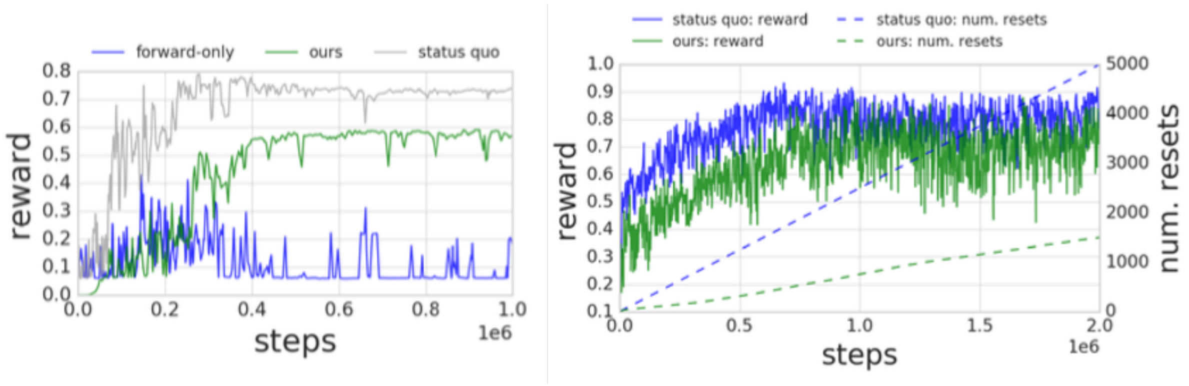


Figure 2: Use of Reset Controller

Then, the paper brings forward the fact that increasing the value of the threshold (Q_{min}) will avoid hard resets, which shows that early aborts helps to no going on irreversible state.
To demonstrate this, different values of Q_{min} has been taken and compared with the number of hard resets. As we can see, when the threshold increase, the number of hard decrease while the rewards will be nearly the same. The fact that the reward stay constant when the Q_{min} increase is not always true : another example an been displayed in the paper showing that the reward will be reduced when the Q_{min} is too large. This can be explained by the fact that the agent will not enough explore to found good states to obtain reward. It can be seen as slot machine : the agent hasn't find the good one because it has not enough explored all of them and will continue on the slot machine chosen near the first one (start state).

However, even if this new method reduce the number of hard resets, some of them are required to improve the learning of the agent. To do so, a number N has been chosen to be the number of steps after which the algorithm will come back to the starting point. This number have to be chosen carefully because it will have an impact on the reward and on the number of hard reset. Indeed, increasing the number N reduces hard reset but reduce reward too, as shown in the figure.

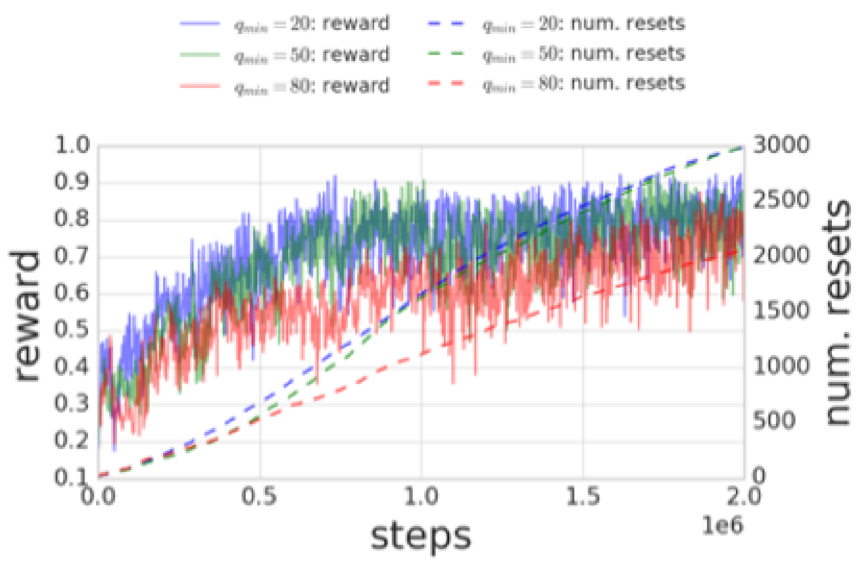


Figure 3: Reduction of hard resets

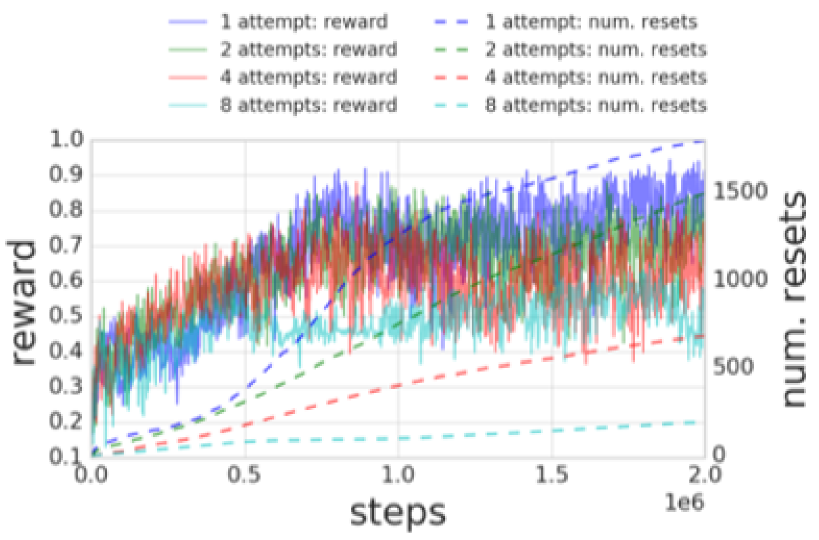


Figure 4: Impact on the choice of N

Another improvement of this method brings forward in the paper is to use ensembles for the Q function. Indeed, as explained because, instead of taking one Q_{value}, we take an ensemble of Q_{value} and use them with an optimistic approach. This will reduce the bias introduced by the fact to have only 1 value to take a decision (reset policy or forward policy).

Moreover, the paper shows that the reset policy will create an automatic curriculum learning. Indeed, an experiment has been made with a peg insertion task (random exploration). This experiment compares an "insert only" method which will will only try every time to learn how to insert the peg in the hole. The result is that only the algorithm presented in the paper succeed to solve this difficult task, after 1 million iterations, by learning from simpler task (trying in random positions) and transfer this knowledge to learn to solve this hard task

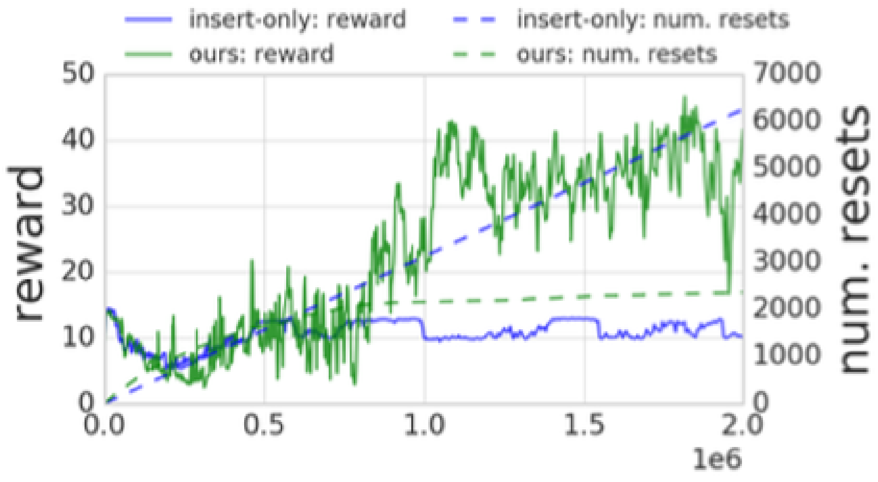


Figure 5: Automatic Curriculum Learning

6 Our Experiments

For our experiments, we decided to train an agent to walk in an environment.

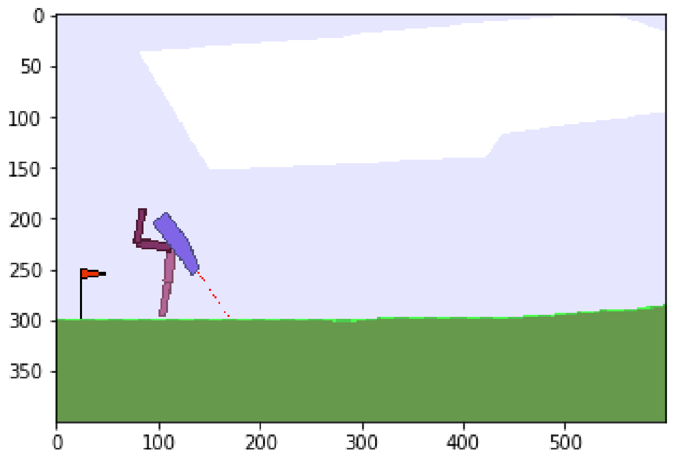


Figure 6: Our agent in its environment

The goal of this agent is to learn to go forward as much as it can. For that matter we used soft actor critic (SAC) methods for both the forward and reset agent. First, we will run our benchmark model which is a simple Soft actor Critic method and collect some informations that we will study later in the notebook. Secondly, we will run our Leave-No-Trace Framework still using (SAC).

Our goal is to study:

- how good is the SAC only method.
- the performance of the main agent when helped by the reset agent
- Comparative study of both methods

- the impact of the reset agent
- the impact of the reset reward function

For the experiments, we will look essentially at two performance "metrics":

The maximum number of steps achieved by the agent per episode

The total number of rewards collected throughout each episode

In our study we were interested in the influence of the reset reward function on the main agent. We then studied two reset reward functions.

- In the first reward function, we penalize the agent when it takes actions that puts him in a state that is very far away from the sate where the hull is horizontal and one leg is straight and touches the ground. We consider that position to be an equilibrium state (not the best necessarily).
- In the second reward function, we consider the equilibrium state to be the one where the hull is horizontal and the body is higher than a certain threshold.

In this work, we tried to prove the point of this paper. We managed to prove that introducing a reset agent allows the main agent to run longer episodes and then reduce the number of hard resets and the cost that it implies. But more than that, we have seen that our agent learns to accomplish its task by taking "safe" actions. It is learning the main goal, while also learning how to be cautious about the actions it takes. For example, in our experiments, we can see that the agent while learning to move forward also learns how to stay in balance and not fall.

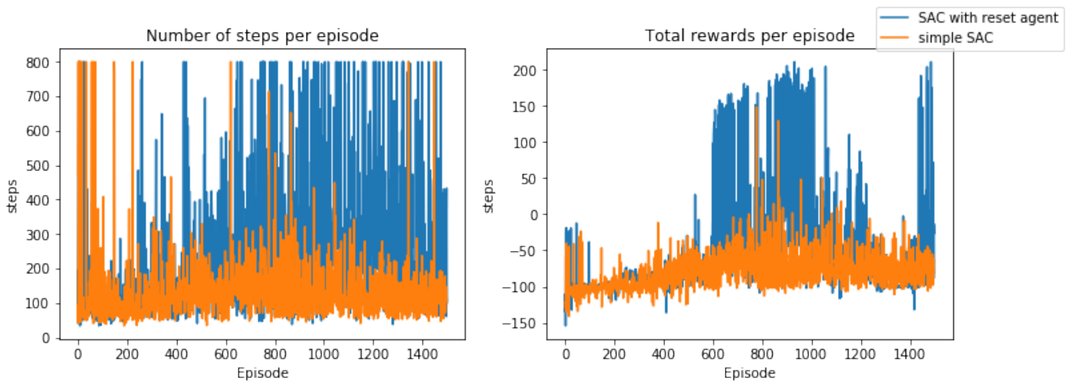


Figure 7: Experiment with the first reward function

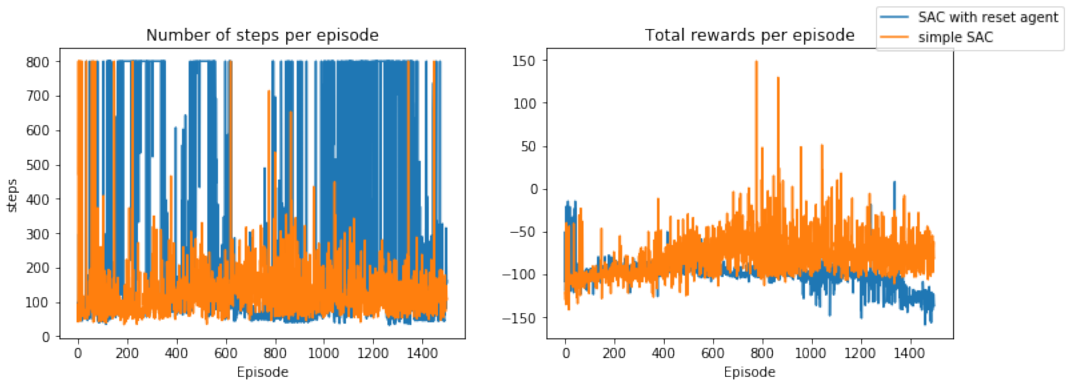


Figure 8: Experiment with the second reward function

As you can see in the figures below, the the reset agent alows the main agent to run much more steps in an episode than it would been able to do alone. Also, in some cases it is able to collect much more rewards (like when we are using the first reward function, which seem to work better than the second).

We also found in our experiments that the choice of the reset reward function is also very important and by the way the set of safe states. In fact, the reset agent seem to interfere a lot with the forward agent. We have observed that sometimes the agent seem to focus more on the task of the reset agent (stay safe) than doing the actual 'job'. This is why it is important in this framework to correctly tune the hyper-parameters like the Q_{min}, and define a correct set of safe states and a suitable reset reward function.

You can visualize those results in some videos in the projects. Also, you may find more details on the results in the project notebook.

7 Conclusion

The authors of [1] introduces a new framework for safe learning while allowing the agent to run much more steps in continual learning than they would have in an ordinary Reinforcement Learning Framework. They introduced a new agent, the reset agent which is a helper for the main agent (called now forward agent); its role is to ensure that, when the forward agent is likely to fail, the latter enters in a safe mode, have recourse to the reset policy, avoid the actions that will lead to a failure, take a series of actions in order to stabilize itself, and then continue the learning. The authors managed to prove the efficiency of this new framework, and throughout our experiments we had the same conclusions.

References

[1] Julian Ibarz Sergey Levine Benjamin Eysenbach, Shixiang Gu. *Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning.* (Submitted on 18 Nov 2017).

[2] Fernando Fernandez Javier Garcia. *A Comprehensive Survey on Safe Reinforcement Learning.* 2015.

[3] Teodor Mihai Moldovan and Pieter Abbeel. *Safe exploration in markov decision processes.* 2012b.